Technische Universität München

Fakultät für Mathematik
Lehrstuhl M2 - Numerische Mathematik (Prof. Dr. B. Wohlmuth)

Accelerating Isogeometric Analysis and Matrix-free Finite Element Methods Using the Surrogate Matrix Methodology

Daniel Peter Drzisga, M.Sc.

Vollständiger Abdruck der von der Fakultät für Mathematik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr. Massimo Fornasier

Prüfer der Dissertation:

1. Prof. Dr. Barbara Wohlmuth

2. Prof. Dr. Matthias Möller

3. Prof. Dr. Giancarlo Sangalli

Die Dissertation wurde am 13.08.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Mathematik am 17.11.2020 angenommen.

Zusammenfassung

In der vorliegenden Arbeit werden Methoden zur Beschleunigung der Matrix Assemblierung in der isogeometrischen Analyse und matrixfreier Finite-Elemente Methoden vorgestellt. Die grundlegende Idee dieser Methoden besteht darin, die Matrixeinträge der linearen Gleichungssysteme, welche aus Diskretisierungen von partiellen Differentialgleichungen stammen, zu approximieren. Auf diese Weise können unnötige Berechnungen vermieden werden, was letztendlich zu kürzeren Laufzeiten bei der Bestimmung von Lösungen dieser Systeme führt. Die Hauptziele dieser Arbeit sind die erste mathematische Analyse dieser Methoden und die Verifizierung deren Genauigkeit und Leistung, indem sie auf lineare und nichtlineare Probleme aus verschiedenen Bereichen der Wissenschaft und Technik angewandt werden.

Abstract

In this thesis, methods for accelerating matrix assembly in isogeometric analysis and stencil-based matrix-free finite element methods are presented. Their functional principle is the approximation of matrix entries in linear systems of equations originating from discretizations of partial differential equations. In this way, unnecessary over-computation may be avoided which ultimately results in a shorter run-time when determining solutions to these systems. The main objectives of this thesis are the first mathematical analysis of these methods and the verification of their accuracy and performance by applying them to linear and nonlinear problems from various fields of science and engineering.

Acknowledgements

The list of people who have, in any way, made valuable contributions to the completion of this work is long. First and foremost, I want to thank my supervisor Prof. Dr. Barbara Wohlmuth for her patience, good advice, and support that I have enjoyed since the beginning of my Bachelor studies. I am also thankful to Prof. Dr. Ulrich Rüde for his ongoing support and for assisting me in the high performance computing aspects of my research. I am very grateful to my reviewers Prof. Dr. Matthias Möller and Prof. Giancarlo Sangalli, Ph.D. I also want to thank Prof. Dr. Massimo Fornasier for agreeing to take the chair of the examining committee.

Moreover, I want to thank the past and present members of the M2 group: First, I want to thank Brendan Keith, Ph.D., for his support, the productive discussions, and his always encouraging words. Working together with you is always a pleasure. I also want to greatly thank my former long-time office mate Dr. Markus Huber for supporting me in the early stages of my Ph.D. studies. I want to thank Dr. Linus Wunderlich for being my mentor and I am grateful to my Bachelor's and Master's thesis advisor Dr. Tobias Köppl for supporting me in my first scientific steps. Furthermore, I am thankful to my fellow Ph.D. students, the postdocs, and the former members of the group for the countless chats, discussions, and entertaining coffee and lunch breaks. A special thanks goes to Jenny Radeck for her kind help in dealing with bureaucratic issues and to Prof. Dr. Rainer Callies for his pieces of advice and counsels. A big thank you goes to Dr. Klaus-Dieter Reinsch who always had an open ear for every issue. Thank you all for creating such a pleasant, productive, and friendly atmosphere at work.

During my research I had the possibility to work with and to learn a lot from many amazing people from different fields. From the Chair for System Simulation in Erlangen, I want to thank Dr. Dominik Bartuschat, Dominik Thönnes, and Nils Kohl for their helpfulness regarding scientific and high performance computing questions. From the Department of Earth and Environmental Sciences in Munich, I want to thank Dr. Simon Bauer, Prof. Dr. Hans-Peter Bunge, and Dr. Marcus Mohr for the fruitful discussions about geophysics and scientific computing. Further, I want to thank my fellow student and colleague Johannes Haubner for exchanging interesting ideas during breaks. I also want to thank my collaborators Prof. Robert Scheichl, Univ.-Prof. Dr. Walter Zulehner, Dr. Björn Gmeiner, Dr. Lorenz John, and Dr. Christian Waluga for their cooperativeness. A special thanks goes to Tzanio Kolev, Ph.D., and Veselin Dobrev, Ph.D., for hosting me at the Lawrence Livermore National Laboratory and allowing me to experience an exciting time in California.

I am very thankful to all the proofreaders of the present text. Moreover, I want to thank all my friends who have supported me in various ways over the last years. Furthermore, I want to express my greatest gratitude to my dear girlfriend Leah who always supported me in all these years. In the end, none of this would have been possible without the encouragement and support of my parents and my brother. Thank you.

List of contributed articles

This thesis is based on the following articles:

Core articles as principal author

I) Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth
The surrogate matrix methodology: a priori error estimation
SIAM Journal on Scientific Computing 41.6 (2019): A3806–A3838
(see also article [40] in the bibliography)

II) Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth
The surrogate matrix methodology: Low-cost assembly for isogeometric analysis
Computer Methods in Applied Mechanics and Engineering 361 (2020): 112776
(see also article [43] in the bibliography)

III) Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth
The surrogate matrix methodology: Accelerating isogeometric analysis of waves
Computer Methods in Applied Mechanics and Engineering 372 (2020): 113322
(see also article [42] in the bibliography)

Further articles

IV) Daniel Drzisga, Ulrich Rüde, and Barbara Wohlmuth
Stencil scaling for vector-valued PDEs on hybrid grids with applications to generalized Newtonian fluids
SIAM Journal on Scientific Computing 42.6 (2020): B1429–B1461
(see also article [45] in the bibliography)

I, Daniel Drzisga, am the principal author of the articles I, II, and III.

List of further contributed articles

The following selected articles include further contributions by the author which are not part of this thesis. They are included for the sake of completeness only. Note that the author of this thesis does not claim to be the principal author of the following articles.

Further articles which are not part of this thesis

- Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth
  The surrogate matrix methodology: A reference implementation for low-cost assembly in isogeometric analysis
  MethodsX (2020): 100813
  (see also article [41] in the bibliography)

- Nils Kohl, Dominik Thönnes, Daniel Drzisga, Dominik Bartuschat, and Ulrich Rüde
  The HyTeG finite-element software framework for scalable multigrid solvers
  International Journal of Parallel, Emergent and Distributed Systems 34.5 (2019): 477-496
  (see also article [69] in the bibliography)

- Simon Bauer, Daniel Drzisga, Marcus Mohr, Ulrich Rüde, Christian Waluga, and Barbara Wohlmuth
  A stencil scaling approach for accelerating matrix-free finite element implementations
  SIAM Journal on Scientific Computing 40.6 (2018): C748-C778
  (see also article [14] in the bibliography)

- Daniel Drzisga, Lorenz John, Ulrich Rüde, Barbara Wohlmuth, and Walter Zulehner
  On the analysis of block smoothers for saddle point problems
  SIAM Journal on Matrix Analysis and Applications 39.2 (2018): 932-960
  (see also article [39] in the bibliography)

- Daniel Drzisga, Björn Gmeiner, Ulrich Rüde, Robert Scheichl, and Barbara Wohlmuth
  Scheduling massively parallel multigrid for multilevel Monte Carlo methods
  SIAM Journal on Scientific Computing 39.5 (2017): S873-S897
  (see also article [38] in the bibliography)

- Daniel Drzisga, Tobias Köppl, Ulrich Pohl, Rainer Helmig, and Barbara Wohlmuth
  Numerical modeling of compensation mechanisms for peripheral arterial stenoses
  Computers in biology and medicine 70 (2016): 190-201
  (see also article [44] in the bibliography)

Contents

1. Introduction

In the field of science and engineering, one often requires to solve partial differential equations (PDEs) which model the underlying physical phenomena. In many scenarios, it can be shown that there exists a solution to a PDE, but it cannot be computed analytically. Since one is still interested in an approximation of the solution, the continuous formulation is discretized and the discrete problem is solved by computers. Typically, these discretizations result in large linear systems described by sparse matrices. The overarching topic of this thesis is accelerating the assembly of these sparse matrices as well as accelerating the matrix-free action of these linear operators while casting the techniques that make this possible into a mathematical framework for error analysis.

1.1. Open research issues

The cost of high performance computing (HPC) is expensive in terms of initial hardware investments but also in terms of running costs due to energy consumption, labor costs, and facility expenses. In particular, energy consumption is one of the leading cost factors as can be observed in emerging rankings like the GREEN500[1] list that ranks the supercomputers by energy efficiency instead of the absolute performance. These considerations gain additional relevance in current times since the first exascale computers which can perform at least a quintillion (10^{18}) floating point operations per second (FLOPS) are expected to be delivered by the end of the year 2021 [7]. It is predicted that the power consumption of these supercomputers will be around 20 MW [103]. To understand the enormous extent of this number, it can be explained by means of a vivid example. According to [101], the average annual electricity consumption for a U.S. residential utility customer was 10 972 kW h in 2018, yielding an average power consumption of about 1.253 kW. This means an exascale computer running at full load will probably require the same amount of energy as about 15 962 households. Assuming an average residential electricity price of \$0.1287/(kW h) [100] results in a total cost of about \$22 548 240 to power an exascale supercomputer for a year. Therefore, it is necessary to find measures for reducing the overall energy consumption by making better use of the available resources.

The required energy of a computer is directly related to the computational work its compute processors perform. This implies that reducing the absolute computing time of a program is not only a profitable goal for the end user, but also a way to reduce the energy consumption. In the past, users could rely on Moore's law which states that new developments in hardware yield computers running twice as fast every 12 to 24 months. However, Moore's law is slowing down and will soon come to an end [104]. Therefore, decreasing the computing time of an application without modifying its implementation is infeasible in the near future.

In order to reduce the computing time, implementations of existing and already established algorithms can be refactored and optimized for the underlying and

[1] https://www.top500.org/lists/green500/2020/06/

future hardware. This may include the adaption of codes for highly specialized accelerators. Among those are, for instance, graphics processing units (GPUs) and field programmable gate arrays (FPGAs). Moreover, it is possible to exploit fine-grained concurrency by using single instruction, multiple data (SIMD) instructions on modern central processing units (CPUs). However, these adaptions are not always feasible since the programs need to meet certain criteria which are often difficult to meet in practical applications. For instance, unavoidable data dependencies between computations may prevent the utilization of SIMD instructions and the execution falls back on utilizing sub-optimal scalar instructions. Likewise, data-intensive programs are prone to be limited by the memory bandwidth and cannot exploit the full potential of the CPUs and accelerators. In those cases where it is not possible to adapt the implementations directly, the underlying algorithms need to be revised from the ground up in order to obtain a better performance.

A large share of programs running on supercomputers are targeted for solving computationally intensive scientific problems in various fields. For example, these fields include weather forecasting, climate research, aerodynamics simulations, and geophysical applications used for earthquake forecasting and plate tectonics simulations. In order to make meaningful predictions in these fields, the simulations require computations with highly resolved data which is only achievable using the resources supercomputers provide. In these applications, innovations and improvements must increasingly rely on specialized algorithms for the problems of interest as well as better implementations suited for future supercomputers. Therefore, it is of utmost importance to carefully rethink and reconstruct long-established algorithms and to replace them by newly developed methods. The revision of methods in scientific computing takes up a large part of this thesis. In particular, this work is focusing on numerical methods for solving PDEs that model problems in science and engineering.

Practical simulations in this field of work are subject to a large number of possible sources of error. They are mainly categorizable into modeling error, numerical error, and data error. The total error of a solution is composed of contributions of all of these types of errors. One possible way of reducing the computing time of a simulation is the avoidance of performing too much unnecessary work which is also known as *over-computation*. Over-computation in this context means reducing a single source of error way below the total error. For instance, solving a linear system obtained from a discretization of a PDE down to the floating point precision of the CPU will not necessarily reduce the total error, since the discretization error may dominate all the other sources of error. This effect is also known as *over-solving* a problem and may be avoided in this case, by using iterative solvers which solve a problem only down to a given residual tolerance. Ideally, this tolerance is chosen as large as possible but still small enough that the discretization error dominates. However, obtaining an a priori estimate on the scale of the discretization error is often difficult or even impossible in the first place. In other situations, sources of over-computation are less obvious and avoiding them might require the development of fundamentally new algorithms.

This thesis deals with finding and analyzing new ways which reduce over-computation when solving PDEs. The idea being pursued here is developing methods which in-

tentionally introduce additional consistency errors in the solution. The ultimate goal is that these consistency errors are much smaller than the total error and using these methods requires a much shorter computing time. Under these premises, the following research questions arise:

- In which steps of the PDE solution process can over-computation be further reduced?
- Is it possible to quantify the errors and to control the accuracy when artificial consistency errors are introduced through a reduction of over-computation?
- What are the conditions which make an efficient reduction of over-computation possible?
- To which problems can these methods be applied to?
- What is the achievable speed-up?
- How can the performance of a method be quantified and compared?

In order to address these questions, we first make a few assumptions on the methods used here. In this work, we restrict ourselves to the Bubnov–Galerkin form of the finite element method (FEM) [97] and isogeometric analysis (IGA) [34, 65] for the discretization of PDEs. In these methods, one or more steps involve solving a linear system. Usually, these linear systems are written in matrix form in which the global matrix is assembled from contributions of smaller local matrices. Depending on the discretization parameters, the global matrices are usually large and sparse, whereas the local matrices are typically small and dense. Especially in IGA, it is a well-known fact that in traditional methods, assembling the matrices takes up a large part of the overall time required to obtain a solution. This fact may be highlighted with the following quote from the 2014 review article [37]:

> "[…] at the moment the assembly of the matrix is the most time-consuming part of isogeometric codes. The development of optimal assembly procedures is an important task required to render isogeometric methods a competitive technology."

Therefore, a large part of the present work targets the development of new assembly procedures for IGA.

Nevertheless, even if these matrices are stored in compressed formats, they require significantly more memory than what is needed to store the solution vectors. But not only the memory consumption presents a challenge, also the required data transfers and latency in loading the matrix indices and entries needs to be considered. In particular, data transport does not only include communication across compute nodes, but also the data transport within a compute node itself. Specifically, this includes data movement from the main memory to the CPU and within a single CPU between the different layers of caches and the registers [57]. An important characteristic for quantifying the efficiency of numerical algorithms on modern computers is the

arithmetic intensity, i.e., the ratio of floating point operations (FLOPs) performed per byte of memory access [57].

In order to improve on the memory consumption and data transfers, matrix-free methods present a possible remedy. Matrix-free methods only compute the results of matrix vector products (MVPs) without requiring an assembled global matrix which consumes a lot of memory and causes a lot of memory traffic when its data is accessed. The fact that most iterative solvers only require the results of MVPs and not the actual entries of the matrix, makes these methods even more attractive.

In the following subsection, we present the state of the art of accelerated matrix assembly methods for IGA and matrix-free approaches for the FEM.

1.2. State of the art

The Bubnov–Galerkin IGA method is a relatively new approach to the discretization of PDEs. Its idea is strongly related to isoparametric FEMs but with the main difference that the underlying basis functions are replaced by non-uniform rational B-splines (NURBS) [63]. It was shown by Hughes et al. [65] that the usage of such bases improves the interoperability between computer-aided design (CAD) and the analysis of PDEs. However, this is not the only benefit of the IGA approach, as it has been demonstrated in a vast amount of IGA literature. For instance, the arbitrary smoothness of NURBS bases may improve the accuracy per degree of freedom with respect to the FEM [34, 36, 37]. This property also facilitates the discretization of high-order PDEs [63, 67]. These features of IGA make it a successful method in modern computational science and engineering research. Nonetheless, assembling the matrices takes up a large part of the overall time required to obtain a solution [37, 91]. Therefore, a lot of work deals with the acceleration of matrix assembly in IGA.

One prominent approach is the utilization of reduced integration rules for the integration of the weak forms [8, 50, 60, 66, 93]. The objective in these approaches is to determine optimal integration strategies which yield a similar accuracy and stability behavior as the full integration rules, but require significantly less computational work. Another related approach is the integration by interpolation and lookup (IIL) proposed in [78, 79] and extended in [87]. In these works, an integrand term from the weak form composed of the PDE coefficients and the geometry mapping is approximated. A further approach is exploiting the tensor-product structure and constructing low rank approximations of the stiffness matrices [62, 80]. In these methods, multi-dimensional integration operations on tensor-product bases are reduced to inexpensive one-dimensional operations on univariate bases.

In this thesis, we present and analyze an alternative methodology which avoids over-computation during the assembly of matrices in IGA. It is based on an idea originally introduced by Bauer et al. in [13] for a low-order matrix-free FEM. The method requires performing integration for only a small fraction of the basis function interactions and the remaining matrix entries are approximated by, for instance, interpolation. Since our method is independent of the underlying integration rules, it can be used in conjunction with the cutting-edge techniques mentioned above and it should not be considered a substitute for them. Many techniques accelerating the

formation and assembly of matrices in IGA show their strength only when high-order approximations are used [5, 26, 29, 61, 92], but here, we also focus on accelerating the assembly of matrices for high-resolution grids.

As it was briefly mentioned in the previous section, matrix-free methods constitute a possible remedy in cases where the memory capacity is limited and data transfer is expensive. To this day, many different strategies to implement matrix-free FEMs have been proposed. The predominant candidate for finite element discretizations is the element-by-element approach [6, 21, 30, 53, 89]. Herein, the local dense stiffness matrices are multiplied by local vectors and the results are added to the global solution vector. However, storing the local stiffness matrices in memory actually requires more memory than storing the global sparse matrix. Another possibility is to compute only the action of the local stiffness matrix to a vector without assembling the matrix itself. Instead of using stored local element matrices, the weak form of a PDE may be integrated on-the-fly [28, 70, 75, 76, 82]. Similarly as in IGA, these approaches may be further accelerated by using reduced integration rules or, in case of tensor-product finite element spaces, by exploiting sum factorization. This is especially beneficial for higher-order FEMs where the on-the-fly integration is a well-suited strategy for future architectures, like GPU based supercomputers, because of its high arithmetic intensity [11, 52, 71, 72, 77, 88]. Moreover, it can be shown that using matrix-free methods instead of matrix-based approaches is already beneficial for the performance if second order discretizations on hexahedral elements are used [71].

In this thesis, however, we focus on a different matrix-free approach based on stencils which provides an alternative to the element-by-element approach. In a stencil-based approach, one does not iterate over the elements of the mesh, but over the rows of the stiffness matrix. The action of the global stiffness matrix is computed by iterating over its rows and computing scalar products with the solution vector, yielding a new value at a single degree of freedom. In this case, the non-zero entries of the rows in the stiffness matrix may be interpreted as so-called *stencils*. This approach is widespread in finite difference discretizations, but may be applied to finite element and IGA discretizations as well.

Stencil-based approaches work especially well when used in conjunction with locally-structured meshes like hierarchical hybrid grids (HHGs) [18, 20, 54], since the sparsity pattern of the matrices and thus the stencil structure remains constant over large parts of the computational domain. By using these grids, efficient matrix-free stencil-based methods have been successfully applied to Poisson's problem and Stokes flow [18, 19, 20, 47, 56].

If, for instance, the coefficient in the PDE is non-constant, the stencils need to be computed on-the-fly which may be similarly expensive as the element-by-element approach. In order to avoid over-computation during the matrix-free assembly of the stencils, a stencil scaling approach for low-order finite element discretizations of scalar elliptic PDEs on HHGs is presented in [14]. In this approach, the variable stencils are obtained by scaling the constant reference stencils corresponding to constant coefficient PDEs. There it is shown that the stencil scaling is able to reduce the computational cost significantly while maintaining the order of convergence of the

original method's error. In this thesis, we extend this approach by considering it for vector-valued PDEs.

Related to the concept of stencil-based approaches, in [13], a two-scale approach for efficient on-the-fly assembly of finite element stencils on HHGs is proposed. This methodology is based on approximating the matrix entries originating from a standard finite element discretization by piece-wise smooth functions. This modification is applied to Poisson's problem in [13] and is extended to Stokes flow with geophysical applications in [12, 15]. In the latter two articles, the idea is applied to surrogate element matrices which are then used to construct the stencils. Each of these articles focuses on the massively parallel high performance computing aspects of their respective methods and provides numerical indication for the error convergence rates.

In this thesis, we take up on this idea of approximating the matrix entries by casting it into a mathematically rigorous framework and providing the first error analysis. Furthermore, we demonstrate the efficiency of this approach in a stencil-based matrix-free finite element framework and advance it to accelerate matrix assembly in IGA.

1.3. Outline

The thesis is structured as follows. In the remainder of this section, we provide a short summary of the contributed articles which are part of this thesis. In Section 2, we bring forward the notation and concepts of the mathematical tools and theories which justify the development of our methods. In Section 3, we provide an overview of the surrogate matrix construction and the techniques we use for solving the emerging linear and nonlinear problems. In Section 4, we give a brief introduction to performance modeling and illustrate its main concepts by considering a specific model called the *roofline* model. Following this overview, we include the contributed core articles in Appendix A and the further articles in Appendix B. Each contributed article is attached alongside a summary and a description of the individual contributions of the author of this thesis. Moreover, we provide the permissions for the publication of the respective articles in this thesis.

1.4. Summary of results

The contributed articles address different methods to accelerate matrix assembly in IGA and stencil-based matrix-free FEMs. Moreover, these methods are analyzed theoretically, their performance is evaluated, and the scope of possible applications is investigated. In core article I (Appendix A.1), we provide a mathematical framework and a rigorous a priori error analysis with numerical verifications of the method originally introduced for low-order matrix-free finite element discretizations by Bauer et al. in [13]. In core article II (Appendix A.2), the ideas of the first core article are applied to accelerate matrix assembly in the IGA setting. In core article III (Appendix A.3), the ideas and considerations from core article II (Appendix A.2) are advanced to the analysis of waves. In article IV (Appendix B.1), the scalar stencil scaling approach for the FEM on hybrid grids introduced in [14] is advanced to

vector-valued PDEs. The following paragraphs provide brief summaries of the scope and subject matter of each contribution.

Core articles as principal author

- Core article I [40] in Appendix A.1:
  The surrogate matrix methodology: a priori error estimation

  In this work, we reconsider the classical Bubnov–Galerkin FEM and analyze a modification of it which is especially advantageous for stencil-based matrix-free computations. The methodology's main idea is approximating the matrix entries originating in a standard finite element discretization by piece-wise smooth functions. This modification is based upon a two-scale approach which was originally introduced in [13], where it was applied to Poisson's problem. In this work, however, we present the first mathematically rigorous analysis of this methodology by performing an a priori error analysis for the variable coefficient Poisson equation. In several numerical experiments, we demonstrate the efficiency of this method in a matrix-free framework using geometric multigrid solvers. The reported speed-ups for all tested examples, when evaluating the surrogate stencils on-the-fly instead of performing the numerical integration, range between a factor of 14 and 20.

- Core article II [43] in Appendix A.2:
  The surrogate matrix methodology: Low-cost assembly for isogeometric analysis

  Following the analysis and application of the surrogate matrix methodology for a low-order matrix-free FEM in core article I (Appendix A.1), we investigate its applicability to IGA in this work. It turns out that using uniform knot vectors in IGA has beneficial properties which can be exploited to efficiently construct surrogate matrices. In this article, we present a methodology to avoid over-assembling matrices in IGA. It is based on performing integration for only a small subset of the trial and test basis function interactions while the rest is approximated through interpolation. The majority of entries in the resulting sparse matrices are computed without using any integration at all. We consider the surrogate matrix methodology for Poisson's problem and provide a priori error estimates which are verified by numerical experiments. Furthermore, we consider the method for the analysis of transverse vibrations of an isotropic elastic membrane and provide an a priori analysis for the eigenvalue errors. Finally, we perform numerical experiments showing the applicability of the surrogate method for plate bending and Stokes' flow problems. At just over one million degrees of freedom, our experiments demonstrate assembly speed-ups beyond fifty times with a simple second-order NURBS basis, without any significant loss in the accuracy of the surrogate solution.

- Core article III [42] in Appendix A.3:
  The surrogate matrix methodology: Accelerating isogeometric analysis of waves

  Following the application of the surrogate matrix methodology to accelerate IGA matrix assembly in core article II (Appendix A.2), we extend the method to other applications targeting the analysis of waves. In this paper, the methodology is applied to several model problems in wave mechanics treated in the Bubnov–Galerkin isogeometric setting. In order to analyze the method theoretically, we consider the Helmholtz equation as a model problem. We present an a priori error analysis for this scenario and demonstrate that the additional consistency error introduced by the utilization of surrogate matrices is independent of the wave number. Additionally, we conduct a floating point complexity analysis which establishes that the computational complexity of the methodology compares favorably to other present fast assembly techniques for isogeometric methods. Furthermore, we consider a time-harmonic elastodynamic wave problem with perfectly matched layers (PML) absorbing boundary conditions and a transient nonlinear hyperelastic wave propagation example involving multiple patches. The numerical experiments demonstrate performance benefits in all experiments and we observe speed-ups of up to 3178%, when compared to the standard assembly algorithm, without losing any significant accuracy.

Further articles

- Article IV [45] in Appendix B.1:
  Stencil scaling for vector-valued PDEs on hybrid grids with applications to generalized Newtonian fluids

  The stencil scaling approach for low-order finite element discretizations presented in [14] marked itself successful for accelerating matrix-free methods on HHGs. However, only scalar elliptic PDEs were considered in that work. Therefore, in the present work, we investigate the applicability of this approach to vector-valued PDEs and observe that the idea of the scalar stencil scaling cannot be applied straightaway. We show that the simple scaling for vector-valued PDEs results in the discretization of a different PDE if the coefficient is not constant. In order to overcome this issue, we develop a new stencil scaling approach by adding a correction term to the discrete stencils. We present how to efficiently pre-compute these correction terms in 2D and 3D, respectively. Furthermore, we provide theoretical computational complexity estimates demonstrating the advantages of this new approach compared to the traditional on-the-fly integration and stored matrix approaches. The theoretical complexity analysis is verified by performing a roofline analysis for MVPs. Furthermore, we demonstrate the convergence rates and the run-time of this extended stencil scaling through a number of numerical experiments. By using this approach on the state of the art supercomputer SuperMUC-NG, we could observe speed-ups of 64% compared to the on-the-fly integration, without any significant loss in the accuracy of the solution.

2. Mathematical background

In this section, we bring forward the notation and concepts of the mathematical tools which justify the development of our methods. First, we introduce the function spaces we consider throughout this work and their corresponding norms and scalar products. Afterwards, we introduce the variable coefficient Poisson equation as our model problem. Moreover, we provide a brief introduction to the FEM, followed by a brief introduction to IGA, and summarize the *a priori* error estimates of the model problem for each approach. Finally, we describe the concept of variational crimes which justifies the development of the surrogate matrix methodology and we introduce further problems that go beyond the scope of the model problem.

2.1. Introduction of function spaces

First, we introduce the standard Sobolev function spaces widely used for the analysis of PDEs. For this purpose, we follow the definitions from [2, 23, 24, 48]. Let $\Omega \subseteq \mathbb{R}^d$ with $d \in \{2, 3\}$ be an open and bounded set with a piecewise smooth boundary. Let $p \in \mathbb{N}$ with $1 \leq p < \infty$. The space $L^p(\Omega)$ defined as

$$L^p(\Omega) = \left\{ v \colon \Omega \to \mathbb{R} : v \text{ measurable}, \int_\Omega |v|^p \, \mathrm{d}\boldsymbol{x} < \infty \right\},$$

is a Banach space when equipped with the norm

$$\|v\|_{0,p,\Omega} = \|v\|_{L^p(\Omega)} = \left(\int_\Omega |v|^p \, \mathrm{d}\boldsymbol{x} \right)^{\frac{1}{p}}.$$

For $p = \infty$, the Banach space $L^\infty(\Omega)$ is defined as

$$L^\infty(\Omega) = \left\{ v \colon \Omega \to \mathbb{R} : v \text{ measurable}, \|v\|_{0,\infty,\Omega} < \infty \right\},$$

with the corresponding norm

$$\|v\|_{0,\infty,\Omega} = \|v\|_{L^\infty(\Omega)} = \operatorname*{ess\,sup}_{\boldsymbol{x} \in \Omega} |v(\boldsymbol{x})|.$$

In the special case $p = 2$, the space $L^2(\Omega)$ is a Hilbert space with the scalar product

$$(u, v)_{0,\Omega} = (u, v)_{L^2(\Omega)} = \int_\Omega u \, v \, \mathrm{d}\boldsymbol{x} \quad \text{for all } u, v \in L^2(\Omega).$$

Let $\boldsymbol{\alpha} \in \mathbb{N}_0^d$ with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)^\intercal$ be a multi-index and let $D^{\boldsymbol{\alpha}} v$ denote the weak-derivative

$$D^{\boldsymbol{\alpha}} v = \frac{\partial^{|\boldsymbol{\alpha}|} v}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}} \quad \text{with} \quad |\boldsymbol{\alpha}| = \sum_{i=1}^d \alpha_i.$$

Let $s \in \mathbb{N}_0$ and $p \in \mathbb{N}$ with $s \geq 0$ and $1 \leq p \leq \infty$. Then the Sobolev space $W^{s,p}(\Omega)$ is defined as

$$W^{s,p}(\Omega) = \left\{ v \colon \Omega \to \mathbb{R} : D^{\boldsymbol{\alpha}} v \in L^p(\Omega), |\boldsymbol{\alpha}| \leq s \right\}.$$

These Sobolev spaces are Banach spaces when equipped with the norm

$$\|v\|_{s,p,\Omega}^2 = \sum_{|\boldsymbol{\alpha}|\leq s} \|D^{\boldsymbol{\alpha}} v\|_{0,p,\Omega}^2.$$

Moreover, we introduce the seminorm defined as

$$|v|_{s,p,\Omega}^2 = \sum_{|\boldsymbol{\alpha}|= s} \|D^{\boldsymbol{\alpha}} v\|_{0,p,\Omega}^2.$$

In the special case $p = 2$, the spaces $W^{s,2}(\Omega)$ have a Hilbert structure. Since we will use these spaces a lot, we denote them by $H^s(\Omega) = W^{s,2}(\Omega)$ throughout this work. The associated scalar product of the spaces $H^s(\Omega)$ is defined as

$$(u, v)_{s,\Omega} = (u, v)_{H^s(\Omega)} = \sum_{|\boldsymbol{\alpha}|\leq s} \int_\Omega (D^{\boldsymbol{\alpha}} u) \cdot (D^{\boldsymbol{\alpha}} v) \, \mathrm{d}\boldsymbol{x} \quad \text{for all } u, v \in H^s(\Omega).$$

Additionally, we introduce the short-hand notations

$$\|v\|_{s,\Omega} = \|v\|_{s,2,\Omega} \quad \text{and} \quad |v|_{s,\Omega} = |v|_{s,2,\Omega}.$$

For the incorporation of essential boundary conditions, we define the subspaces $H_0^s(\Omega) \subseteq H^s(\Omega)$ with $s \geq 1$ via

$$H_0^s(\Omega) = \left\{ v \in H^s(\Omega) : v|_{\partial\Omega} = 0 \right\},$$

where the restriction to the boundary $\partial\Omega$ is to be understood in the trace sense.

If it is clear from the context which domain Ω is meant, we drop the subscripted Ω from the norms and scalar products in order to simplify the notation.

2.2. Definition of the model problem

We consider the variable coefficient Poisson equation as a scalar elliptic model problem. This type of problem arises in many science and engineering applications. For instance, it is used to obtain the potentials of, e.g., electrostatic and gravitational fields. The problem is also of large interest since it often emerges as a part of larger and more complicated problems. For example, some numerical schemes for the approximate solution of the Navier-Stokes equations require to solve the Poisson equation in each time step [31]. Moreover, it is also used in many groundwater and porous media flow applications in which the variable coefficient may be used to model the permeability of the medium [10].

Let $K \colon \Omega \to \mathbb{R}^{d\times d}$ be a symmetric and positive definite tensor and $f \colon \Omega \to \mathbb{R}$ a function. We seek for a function $u \colon \Omega \to \mathbb{R}$ solving the following variable coefficient Poisson equation with homogeneous Dirichlet boundary conditions:

$$\begin{aligned} -\operatorname{div}(K\nabla u) &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{2.1}$$

We restrict ourselves to the case of homogeneous Dirichlet boundary conditions since the analysis of problems involving inhomogeneous and more general boundary conditions can often be reduced to a slightly modified problem with homogeneous boundary conditions [23, p. 40]. In order to analyze the problem theoretically, we narrow the spaces in which K, f, and u are in. Let $K \in [L^\infty(\Omega)]^{d\times d}$ be a symmetric and positive definite tensor such that its smallest eigenvalue is bounded away from zero almost everywhere. Let $f \in L^2(\Omega)$ and $V = H_0^1(\Omega)$. The corresponding weak form of (2.1) reads:

$$\text{Find } u \in V \text{ satisfying} \qquad a(u,v) = F(v) \qquad \text{for all } v \in V, \tag{2.2}$$

where $a\colon V \times V \to \mathbb{R}$ and $F\colon V \to \mathbb{R}$ are defined as

$$\begin{aligned} a(u,v) &= \int_\Omega \nabla u^\mathsf{T} K \nabla v \,\mathrm{d}\boldsymbol{x} && \text{for all } u,v \in V,\\ F(v) &= (f,v)_{0,\Omega} && \text{for all } v \in V. \end{aligned}$$

The bilinear form $a(\cdot,\cdot)$ is continuous, i.e., there exists a $C \in \mathbb{R}$ with $C > 0$ depending on K such that

$$a(u,v) \le C\|u\|_1\|v\|_1 \quad \text{for all } u,\, v \in V.$$

Additionally, $a(\cdot,\cdot)$ is coercive, i.e., there exists an $\alpha \in \mathbb{R}$ with $\alpha > 0$ depending on K and Ω such that

$$a(u,u) \ge \alpha\|u\|_1^2 \quad \text{for all } u \in V.$$

Moreover, $F \in V^*$ is a bounded linear functional where V^* denotes the dual space of V. Under these assumptions, the Lax-Milgram lemma asserts that problem (2.2) is well-posed in the sense that there exists a unique solution $u \in V$ which depends continuously on the data f; see, e.g., [48, Lemma 2.2].

The approach of constructing weak variational formulations for the PDEs of interest does not only yield the possibility to analyze them theoretically, but also to compute approximate solutions. The main idea is to restrict the continuous variational formulation (2.2) to a finite-dimensional subspace $V_h \subseteq V$ and to find a solution to the variational formulation in V_h. In particular, we employ the Bubnov–Galerkin method in which the trial and test spaces are chosen to be equal, as opposed to the Petrov–Galerkin method [23, 33, 48]. The discrete problem corresponding to (2.2) then reads:

$$\text{Find } u_h \in V_h \text{ satisfying} \qquad a(u_h,v_h) = F(v_h) \qquad \text{for all } v_h \in V_h\,. \tag{2.3}$$

In the next subsections, we present two popular ways of choosing the finite-dimensional subspace V_h. Additionally, we present the well-established a priori error estimates when using these discrete spaces for the approximation of $u \in V$ in (2.2).

2.3. Finite element method

A popular choice of the space $V_h \subseteq V$ for (2.3) may be constructed using the finite element method (FEM) [97]. Many different types of finite elements have been

proposed since its introduction, among them non-conforming variants which are used in the discontinuous Galerkin approach [90]. However, we restrict ourselves to conforming discretizations with local polynomial bases on simplicial meshes.

Let Ω be a polyhedral domain which is decomposed into triangles in 2D or tetrahedra in 3D. These decompositions are often called meshes or grids. Since we focus on conformal meshes only, we follow the definition of a conformal mesh from [23]. Assume $\mathcal{T}_h = \{T_1, T_2, \ldots, T_M\}$ to be a set of triangles or tetrahedra decomposing Ω. $\mathcal{T}_h$ is called a conforming mesh if the following properties are satisfied:

1. $\bar{\Omega} = \bigcup_{i=1}^{M} T_i$.
2. If $T_i \cap T_j$ contains exactly one element, then it is a corner vertex of T_i and T_j.
3. If $T_i \cap T_j$ with $i \neq j$ contains more than one element, then $T_i \cap T_j$ is an edge or face of T_i and T_j.
4. Each $T \in \mathcal{T}_h$ has a diameter of at most $2h$.

Additionally, a family of meshes $\{\mathcal{T}_h\}_{h>0}$ is called shape-regular if there exists a constant $\kappa > 0$, such that each element $T \in \mathcal{T}_h$ contains a ball of radius ρ_T with $\rho_T \geq h_T/\kappa$, where h_T is half the diameter of T.

Remark 1 *The HHGs employed in core article I (Appendix A.1) and article IV (Appendix B.1) satisfy the properties of a shape-regular conforming grid family provided that the initial input mesh satisfies them. This is because the uniform refinements used there do not introduce any new types of elements which violate these properties.*

The meshes are used to construct the discrete function spaces V_h by associating local basis functions to the elements in $\mathcal{T}_h$. Let $\mathcal{P}_p(T)$ be the set of polynomials up to degree p on $T \in \mathcal{T}_h$. Then the discrete finite element space $V_{h,p}^{\mathrm{FE}}$ may be defined as

$$V_{h,p}^{\mathrm{FE}} = \{v \in \mathcal{C}^0(\bar{\Omega}) : v|_T \in \mathcal{P}_p(T) \text{ for all } T \in \mathcal{T}_h\} \cap H_0^1(\Omega). \tag{2.4}$$

The following Theorem 1 provides a priori error estimates for the discrete solution $u_h \in V_{h,p}^{\mathrm{FE}}$ of (2.3) in the $H^1(\Omega)$- and $L^2(\Omega)$-norms [48, Theorems 3.16 and 3.18].

Theorem 1 (FEM: A priori error estimates for the model problem)
Let $\Omega \subseteq \mathbb{R}^d$ be a polyhedral domain and let $\{\mathcal{T}_h\}_{h>0}$ be a shape-regular family of geometrically conformal meshes of Ω. Let $u \in V$ be the solution of the continuous problem (2.2) and let $u_h \in V_{h,p}^{\mathrm{FE}}$ be the solution of the discrete problem (2.3) with $V_h = V_{h,p}^{\mathrm{FE}}$. If $u \in H^{s+1}(\Omega)$ with $0 \leq s \leq p$ and $f \in L^2(\Omega)$, the following error estimate holds true:

$$\|u - u_h\|_1 \leq C_1 h^s |u|_{s+1},$$

and if, additionally, Ω is convex and $K \in [\mathcal{C}^1(\bar{\Omega})]^{d\times d}$, we have

$$\|u - u_h\|_0 \leq C_0 h^{s+1} |u|_{s+1},$$

with constants $C_0 > 0$ and $C_1 > 0$ independent of h and u.

The convexity of Ω and regularity of K is only required for the improved error estimate in the $L^2(\Omega)$-norm. Since the operator induced by $a(\cdot,\cdot)$ is self-adjoint, the additional regularity asserts that problem (2.2) is $H^2(\Omega)$ regular and an Aubin-Nitsche type argument may be applied [48, Theorem 3.12]. The estimate in the $H^1(\Omega)$-norm holds independently of these assumptions.

2.4. Isogeometric analysis

In practice, the problem domains Ω are often constructed using CAD software which use B-splines and NURBS to represent curves, surfaces, and solids. Constructing meshes suitable for the FEM from these CAD surface and solid representations is often very difficult and time intensive [34]. In particular, the meshes for the FEM can often merely approximate the true domains described by NURBS, since they allow the construction of perfect circular boundaries. Such boundaries are not exactly representable even with an isoparametric FEM, where functions from $V_{h,p}^{\mathrm{FE}}$ are used to construct a map from the reference to the physical domain. In order to overcome the issue of investing time to construct a suitable mesh, IGA uses the idea of utilizing the same functions which represent the geometry for the discretization of the PDE.

For the definitions of B-splines and NURBS, we follow [34], but a more rigorous and detailed description can be found in [65]. Let $\Xi = \{\xi_1, \xi_2, \dots, \xi_{m+p+1}\}$ be a one-dimensional non-decreasing knot vector over the unit interval with $\xi_i \in [0,1]$, $m \in \mathbb{N}$ the number of B-spline basis functions associated to Ξ, p the polynomial order, and $h = \max_{1\le k\le m-1} |\xi_{k+1} - \xi_k|$. We assume that Ξ is *open*, meaning that $\xi_1 = \cdots = \xi_{p+1} = 0$ and $\xi_m = \cdots = \xi_{m+p+1} = 1$. Knot vectors are called *uniform* if the knots ξ_i are equally spaced and *non-uniform* if this is not the case. Moreover, we introduce the concept of elements in IGA which are relatable to the elements in the FEM. In 1D, the mesh corresponding to the knot vector Ξ may be defined as the set $\mathcal{T}_h = \{[\xi_k, \xi_{k+1}] : \xi_k \neq \xi_{k+1}, 1 \le k \le m-1\}$.

We define the univariate B-splines over the knot vector Ξ recursively using the Cox-de Boor recursion formula [22, 35]. The basis functions for the initial case $p = 0$ and $1 \le i \le m$ are given as

$$b_{i,0}(\widehat{x}) = \begin{cases} 1 & \text{if } \xi_i \le \widehat{x} < \xi_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

The basis functions for $p > 0$ are defined recursively in the following way

$$b_{i,p}(\widehat{x}) = \frac{\widehat{x} - \xi_i}{\xi_{i+p} - \xi_i} b_{i,p-1}(\widehat{x}) + \frac{\xi_{i+p+1} - \widehat{x}}{\xi_{i+p+1} - \xi_{i+1}} b_{i+1,p-1}(\widehat{x}).$$

We extend the definition of the univariate B-splines on Ξ to the multi-dimensional knot vector $\Xi^d = \Xi \otimes \cdots \otimes \Xi$. The mesh corresponding to Ξ^d is defined as $\mathcal{T}_h = \{[\xi_{k_1}, \xi_{k_1+1}] \times \cdots \times [\xi_{k_d}, \xi_{k_d+1}] : \xi_{k_i} \neq \xi_{k_i+1},\ 1 \le k_i \le m-1,\ 1 \le i \le d\}$. Let $N = m^d$ and for each $1 \le i \le N$, we associate a multi-index $\boldsymbol{i} = (i_1, \dots, i_d)^\intercal \in \mathbb{N}^d$ with $1 \le i_k \le m$ through the colexicographical relationship $i = i_1 + (i_2 - 1)m + \ldots + (i_d - 1)m^{d-1}$. Then the multivariate B-spline basis is given by $\{B_{i,p}\}_{1\le i\le N}$ with basis functions

$$B_{i,p}(\widehat{\boldsymbol{x}}) = b_{i_1,p}(\widehat{x}_1) \cdots b_{i_d,p}(\widehat{x}_d),$$

for all $\widehat{\boldsymbol{x}} = (\widehat{x}_1, \ldots, \widehat{x}_d) \in [0,1]^d = \widehat{\Omega}$. Let $\omega_i > 0$ for $1 \leq i \leq N$ be fixed weight parameters. The NURBS basis functions are then defined as

$$\widehat{N}_{i,p}(\widehat{\boldsymbol{x}}) = \frac{\omega_i B_{i,p}(\widehat{\boldsymbol{x}})}{\sum_{j=1}^{N} \omega_j B_{j,p}(\widehat{\boldsymbol{x}})} \quad \text{for all } \widehat{\boldsymbol{x}} \in \widehat{\Omega},\ 1 \leq i \leq N.$$

The B-spline and NURBS basis functions satisfy the following partition of unity and point-wise positivity properties [34, p. 22]

$$\begin{aligned} \sum_{i=1}^{N} B_{i,p}(\widehat{\boldsymbol{x}}) = 1, \qquad & \sum_{i=1}^{N} \widehat{N}_{i,p}(\widehat{\boldsymbol{x}}) = 1 \qquad && \text{for all } \widehat{\boldsymbol{x}} \in \widehat{\Omega}, \\ B_{i,p}(\widehat{\boldsymbol{x}}) \geq 0, \qquad & \widehat{N}_{i,p}(\widehat{\boldsymbol{x}}) \geq 0 \qquad && \text{for all } \widehat{\boldsymbol{x}} \in \widehat{\Omega},\ 1 \leq i \leq N. \end{aligned} \tag{2.5}$$

Because of the partition of unity property (2.5), the B-spline and NURBS basis functions coincide if $\omega_i = 1$ for $1 \leq i \leq N$. Another favorable property of the B-spline and NURBS basis functions of degree p is, that they are $p-1$ times continuously differentiable in the absence of repeated knots.

We assume that the physical domain Ω is parameterized by a map $\boldsymbol{\varphi}(\widehat{\Omega}) = \Omega$ with

$$\boldsymbol{\varphi}(\widehat{\boldsymbol{x}}) = \sum_{i=1}^{N} \boldsymbol{c}_i \widehat{N}_{i,p}(\widehat{\boldsymbol{x}}) \quad \text{for all } \widehat{\boldsymbol{x}} \in \widehat{\Omega},$$

where the $\boldsymbol{c}_i \in \mathbb{R}^d$ are called control points. The discrete subspace of $V_{h,p}^{\mathrm{IGA}} \subseteq V$ in the IGA context is then defined as the set of NURBS basis functions on the reference domain $\widehat{\Omega}$ composed with the pull-back operator $\boldsymbol{\varphi}^{-1}$, i.e.,

$$V_{h,p}^{\mathrm{IGA}} = \operatorname{span}\left\{\widehat{N}_{i,p} \circ \boldsymbol{\varphi}^{-1}\right\}_{1 \leq i \leq N} \cap H_0^1(\Omega). \tag{2.6}$$

Remark 2 *In core article II (Appendix A.2) and core article III (Appendix A.3), we focus only on uniform open knot vectors. In this case, the B-spline basis functions associated to the knots away from the boundary have an interesting property. They are all alike up to a translational shift which provides a locally uniform structure in the discretization. Such B-splines are called* cardinal *and they play an important role in the construction of stencil functions and surrogate matrices described in Section 3.*

The following Theorem 2 provides a priori error estimates for the discrete solution $u_h \in V_{h,p}^{\mathrm{IGA}}$ of (2.3) in the $H^1(\Omega)$- and $L^2(\Omega)$-norms which can be shown by utilizing the approximation properties of NURBS [16, Theorem 3.2] [34, Appendix 3.B] [98].

Theorem 2 (IGA: A priori error estimates for the model problem)
Let $\Omega \subseteq \mathbb{R}^d$ be a bounded and open domain parameterized by $\boldsymbol{\varphi}(\widehat{\Omega}) = \Omega$. Assume that the IGA mesh is refined by inserting knots into Ξ without changing Ω. Let $u \in V$ be the solution of the continuous problem (2.2) and let $u_h \in V_{h,p}^{\mathrm{IGA}}$ be the solution

of the discrete problem (2.3) with $V_h = V_{h,p}^{\mathrm{IGA}}$. *If* $u \in H^{s+1}(\Omega)$ *with* $0 \leq s \leq p$ *and* $f \in L^2(\Omega)$, *the following error estimate holds true:*

$$\|u - u_h\|_1 \leq \widehat{C}_1 h^s \|u\|_{s+1},$$

and if, additionally, Ω *is a domain of class* $\mathcal{C}^2$ *and* $K \in [\mathcal{C}^1(\overline{\Omega})]^{d \times d}$, *we have*

$$\|u - u_h\|_0 \leq \widehat{C}_0 h^{s+1} \|u\|_{s+1},$$

where $\widehat{C}_0 > 0$ *and* $\widehat{C}_1 > 0$ *are constants independent of* u *and* h.

As in the FEM case, the regularity of Ω and K is only required for the improved error estimate in the $L^2(\Omega)$-norm. Since the operator induced by $a(\cdot,\cdot)$ is self-adjoint, the additional regularity asserts that problem (2.2) is $H^2(\Omega)$ regular and an Aubin-Nitsche type argument may be applied [48, Theorem 3.10]. Theorem 2 also demonstrates that IGA possesses the same convergence properties as the FEM from Section 2.3, but the NURBS spaces require a control on the full norm of u instead of only its seminorm.

The following subsection deals with the issue that the bilinear forms $a(\cdot,\cdot)$ may not always be evaluated exactly which introduces additional consistency errors.

2.5. Variational crimes

Let $a\colon V \times V \to \mathbb{R}$ be a continuous and coercive bilinear form and $F \in V^*$ a bounded linear functional. For instance, $a(\cdot,\cdot)$ and $F(\cdot)$ may be thought of as the forms of the model problem introduced in Section 2.2. Recalling the previous subsections, we can state the following continuous and discrete variational formulations:

$$\text{Find } u \in V \text{ satisfying} \qquad a(u,v) = F(v) \quad \text{for all } v \in V. \tag{2.7a}$$

$$\text{Find } u_h \in V_h \text{ satisfying} \quad a(u_h,v_h) = F(v_h) \quad \text{for all } v_h \in V_h. \tag{2.7b}$$

Additionally, we introduce $\tilde{a}\colon V_h \times V_h \to \mathbb{R}$ as an approximation of $a|_{V_h \times V_h}$ which we call the *surrogate* for $a(\cdot,\cdot)$ and $\widetilde{F}\colon V_h \to \mathbb{R}$ as an approximation of $F|_{V_h}$. The corresponding variational formulation using the surrogate forms is:

$$\text{Find } \tilde{u}_h \in V_h \text{ satisfying} \quad \tilde{a}(\tilde{u}_h,v_h) = \widetilde{F}(v_h) \quad \text{for all } v_h \in V_h. \tag{2.8}$$

In order to obtain a bound on the error of the surrogate solution $\tilde{u}_h$, we can make use of the First Strang Lemma [23, pp. 100–101]:

Lemma 1 (First Strang Lemma)
Assume that $\tilde{a}\colon V_h \times V_h \to \mathbb{R}$ *is uniformly coercive. Then the following error estimate holds true for the surrogate solution* $\tilde{u}_h$*:*

$$\begin{aligned}\|u - \tilde{u}_h\|_1 \leq C_2 \Bigg(\inf_{v_h \in V_h} \left[\|u - v_h\|_1 + \sup_{w_h \in V_h \setminus \{0\}} \frac{|a(v_h,w_h) - \tilde{a}(v_h,w_h)|}{\|w_h\|_1} \right] \\ + \sup_{w_h \in V_h \setminus \{0\}} \left[\frac{|F(w_h) - \widetilde{F}(w_h)|}{\|w_h\|_1} \right] \Bigg),\end{aligned} \tag{2.9}$$

with a constant C_2 *independent of* h.

Henceforth, we assume that the linear forms on the right-hand side do not differ, i.e., $\widetilde{F} = F$. Therefore, the second supremum in (2.9) vanishes and it is sufficient to control only the first supremum

$$\sup_{w_h \in V_h \setminus \{0\}} \frac{|a(v_h, w_h) - \tilde{a}(v_h, w_h)|}{\|w_h\|_1} \quad \text{for } v_h \in V_h,$$

which we refer to as the consistency term. Lemma 1 provides a theoretical framework to obtain a priori error estimates if the true bilinear form $a(\cdot, \cdot)$ is replaced by a surrogate bilinear form $\tilde{a}(\cdot, \cdot)$. Such surrogate bilinear forms do not need to be particularly constructed since they often arise naturally in the discretization process. For instance, integration rules only yield approximations of the integrals in the bilinear forms. The usual integration rules only integrate polynomials up to a certain degree exactly, therefore, NURBS with non-constant weights cannot be integrated without error. In the isoparametric FEM, curved boundaries may not be represented exactly by the underlying mesh which introduces an additional geometry approximation error. These additional consistency errors are called variational crimes. Luckily, it can often be shown that these variational crimes do not deteriorate the convergence order of the discretization [24, 95, 96]. Actually, this fact is being exploited in existing methods which intentionally use reduced or weighted integration rules to speed up the numerical integration [8, 29, 50, 60, 66, 92, 93]. In this work, we assume that the variational crimes introduced by the standard discretization approach do not deteriorate the convergence order of the discretization errors.

In [13], Bauer et al. introduced a novel approach to approximate the matrix entries in matrices emerging from a standard finite element discretization. It is based on introducing a coarser scale $H > h$ and relating the matrix entries by locally smooth functions. There, the approach was applied to low order finite element discretizations of Poisson's problem. Afterwards, in two follow-up articles, Stokes flow was considered in [12, 15]. Each of these articles focused on the massively parallel HPC aspects of the respective methods and provided numerical indication for the convergence rates. In this thesis, we take up this approach by providing the first mathematically rigorous framework and analysis of the consistency error introduced by this approach and by applying it to IGA. Since the original discretization matrices are replaced by surrogates, we call this method the *surrogate matrix methodology*. In Section 3, we briefly present the idea of how to construct surrogate matrices which in turn define surrogate bilinear forms $\tilde{a}(\cdot, \cdot)$. We use these surrogate bilinear forms to obtain error estimates on the consistency error introduced by this approach. We perform this analysis for the model problem discretized with linear finite elements in core article I (Appendix A.1) and for IGA in core article II (Appendix A.2). Here, we briefly sketch the rough steps of the proofs. More detailed information is available in each of the articles.

Let $q \in \mathbb{N}$ be the freely selectable surrogate matrix approximation order, assume that $K \in [W^{q+1,\infty}(\Omega)]^{d \times d}$ and that the assumptions of Theorem 1 in the FEM setting or of Theorem 2 in the IGA setting are fulfilled. The parameter q is related to the order of the approximation space used for the approximation of the matrix entries, as it is shown in Section 3. In a first step, we verify that the constructed bilinear

forms $\tilde{a}(\cdot,\cdot)$ are uniformly coercive if H is small enough. In a second step, we show that for the model problem, the following inequality holds true for some constant $C_3 > 0$ depending on $|K|_{W^{q+1,\infty}(\Omega)}$ but independent of h and H:

$$|a(v_h, w_h) - \tilde{a}(v_h, w_h)| \leq C_3 H^{q+1} |v_h|_1 |w_h|_1 \quad \text{for all } v_h,\, w_h \in V_h. \tag{2.10}$$

Employing Lemma 1 and plugging (2.10) into (2.9) yields

$$\|u - \tilde{u}_h\|_1 \leq C_2(\|u - v_h\|_1 + C_3 H^{q+1} \|v_h\|_1) \quad \text{for all } v_h \in V_h.$$

We choose v_h to be the solution of (2.7b), i.e., $v_h = u_h$. Using the $H^1(\Omega)$ error estimate from Theorem 1 or Theorem 2 and using that $\|u_h\|_1 \leq C_4 |u_h|_1 \leq C_4 |u|_1$, for some h-independent constant $C_4 > 0$, we can bound the surrogate error by the norms of the solution of (2.7a) via

$$\|u - \tilde{u}_h\|_1 \leq C_2(\widetilde{C}_1 h^s \|u\|_{s+1} + C_3\, C_4 H^{q+1} |u|_1),$$

where $\widetilde{C}_1 \in \{C_1, \widehat{C}_1\}$ is the constant from either of the Theorems 1 or 2. In this scenario, the consistency error may be bounded by $C_3\, C_4 H^{q+1} |u|_1$. This term only depends on the $H^1(\Omega)$-seminorm of the solution of the continuous problem (2.7a) in contrast to the discretization error which depends on the full $H^{s+1}(\Omega)$-norm if NURBS basis functions are used. In the FEM setting, this norm may be replaced by the $H^{s+1}(\Omega)$-seminorm. Choosing $q > s - 1$ and $H = \mathcal{O}(h)$ guarantees that the consistency error is asymptotically dominated by the discretization error in the $H^1(\Omega)$-norm.

In the setting of Theorems 1 and 2, we can also provide error estimates in the $L^2(\Omega)$-norm. From the triangle inequality it follows that

$$\begin{aligned} \|u - \tilde{u}_h\|_0 &\leq \|u - u_h\|_0 + \|u_h - \tilde{u}_h\|_0 \\ &\leq \widetilde{C}_0 h^{s+1} \|u\|_{s+1} + \|u_h - \tilde{u}_h\|_0, \end{aligned}$$

where $\widetilde{C}_0 \in \{C_0, \widehat{C}_0\}$ is the constant from either of the Theorems 1 or 2. Therefore, only the term $\|u_h - \tilde{u}_h\|_0$ needs to be controlled. Let $w_h \in V_h$ satisfy $a(w_h, v_h) = (u_h - \tilde{u}_h, v_h)_0$ for all $v_h \in V_h$. The Lax-Milgram lemma asserts that $|w_h|_1 \leq C_5 \|u_h - \tilde{u}_h\|_0$ for some constant C_5 [48, Lemma 2.2]. Using (2.10), we obtain

$$\begin{aligned} \|u_h - \tilde{u}_h\|_0^2 = a(w_h, u_h - \tilde{u}_h) &= \tilde{a}(w_h, \tilde{u}_h) - a(w_h, \tilde{u}_h) \\ &\leq C_3 H^{q+1} |w_h|_1 |\tilde{u}_h|_1 \\ &\leq C_3\, C_5 H^{q+1} \|u_h - \tilde{u}_h\|_0 |\tilde{u}_h|_1. \end{aligned}$$

Finally, we obtain

$$\|u - \tilde{u}_h\|_0 \leq \widetilde{C}_0 h^{s+1} \|u\|_{s+1} + C_3\, C_5 H^{q+1} |\tilde{u}_h|_1,$$

where $|\tilde{u}_h|_1 \to |u_h|_1$ as $H \to 0$ and $|u_h|_1 \leq |u|_1$. In this case, choosing $q > s$ and $H = \mathcal{O}(h)$ guarantees that the consistency error is asymptotically dominated by the

discretization error in the $L^2(\Omega)$-norm. As above, the full norm of u may be replaced by the $H^{s+1}(\Omega)$-seminorm when finite elements are used.

In core article I (Appendix A.1), we establish this theory for a low order finite element discretization on HHGs and verify it numerically for various benchmark problems. In the follow up core article II (Appendix A.2), we transfer the ideas of the method to IGA and provide similar error estimates as in core article I (Appendix A.1). In core article III (Appendix A.3), we develop similar results for the Helmholtz equation (see Section 2.6) and show that the additional consistency error is independent of the wave number.

In the following subsection, we briefly present the benchmark problems considered in the contributed articles.

2.6. Further problems

In core article I (Appendix A.1) and core article II (Appendix A.2), we present a priori error estimates for the variable coefficient Poisson model problem when surrogate bilinear forms are used. In the following, we briefly introduce further benchmark problems we utilize to assert the error convergence rates and performance of our methods. Although we do not show theoretical results for most of the further problems, we expect them to carry over to these more involved problems. In fact, we can verify these expectations numerically in the contributed articles.

One of the further problems considers applications in solid mechanics, or more precisely, the problem of compressible Lamé-Navier linearized elasticity. In this problem one wants to predict the stresses in the material and the deformation a material undergoes when certain loads are applied to it. This model is only valid for very small deformations in which the material behaves linearly in the strain rate. Let Ω be as before. Additionally, we assume that the boundary of Ω is partitioned into two relatively open sets $\overline{\Gamma_\mathrm{D} \cup \Gamma_\mathrm{N}} = \partial\Omega$, $\Gamma_\mathrm{D} \cap \Gamma_\mathrm{N} = \emptyset$, and denote its outward unit normal by $\boldsymbol{n}$. Let $\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}(\vec{\nabla}\boldsymbol{u} + (\vec{\nabla}\boldsymbol{u})^\mathsf{T})$ be the symmetric gradient of $\boldsymbol{u}$ and $\boldsymbol{\sigma}$ the stiffness tensor of Hooke's law for an isotropic material, i.e., $\boldsymbol{\sigma}(\boldsymbol{\varepsilon}) = 2\mu\boldsymbol{\varepsilon} + \lambda \operatorname{tr}(\boldsymbol{\varepsilon})\boldsymbol{I}$, where μ and λ are the Lamé parameters and $\boldsymbol{I}$ is the identity tensor [23, p. 281]. Let $\boldsymbol{f}$ be the body forces, $\boldsymbol{g}$ the prescribed displacement, and $\boldsymbol{t}$ the external forces. The displacement $\boldsymbol{u}$ of the material after the loads are applied is obtained by solving the equilibrium equation

$$\begin{aligned} -\operatorname{Div}\boldsymbol{\sigma} &= \boldsymbol{f} \quad \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{g} \quad \text{on } \Gamma_\mathrm{D}, \\ \boldsymbol{\sigma}\cdot\boldsymbol{n} &= \boldsymbol{t} \quad \text{on } \Gamma_\mathrm{N}, \end{aligned} \tag{2.11}$$

where Div denotes the row-wise divergence operator. We consider the following form of the problem: Find $\boldsymbol{u} \in [V]^d$ solving the variational form of (2.11) in which the boundary conditions are incorporated in the right-hand side linear form. Using Korn's inequalities [48, Theorem 3.77, Thoerem 3.78], it is possible to prove existence and uniqueness of a solution $\boldsymbol{u}$. Furthermore, it is possible to derive optimal a priori error estimates in the $H^1(\Omega)$-norm equivalent to the $H^1(\Omega)$ estimates in Theorem 1 and Theorem 2 [48, Prop. 3.82] [16]. Similar optimal a priori error estimates in the $L^2(\Omega)$-norm are only available for the pure traction problem where $\Gamma_\mathrm{D} = \emptyset$ or

$\boldsymbol{g} = 0$ [48, Prop. 3.84] [16]. In article IV (Appendix B.1), we introduce a vector-valued stencil scaling to solve linear elasticity problems with spatially varying Lamé parameters.

The equilibrium equation (2.11) may also be employed to describe fluids in which the viscous forces dominate the advective inertial forces. In this case, we additionally demand that the mass is conserved, i.e., $\operatorname{div} \boldsymbol{u} = 0$. This particular type of fluid flow is also known as incompressible Stokes flow. Examples for such type of fluids are lava and the very slowly moving Earth's solid mantle which is why the Stokes flow is of large interest in the studying of Earth's mantle convection [12, 15]. In this scenario, we interpret $\boldsymbol{u} \in [V]^d$ as the velocity field and introduce an additional pressure function $p \in L^2(\Omega)$. The corresponding stiffness tensor is given by $\boldsymbol{\sigma}(\boldsymbol{\varepsilon}) = 2\mu\boldsymbol{\varepsilon} - p\boldsymbol{I}$, where μ is the viscosity of the fluid [46, p. 8]. In order for a unique weak solution $(\boldsymbol{u}, p)$ of (2.11) to exist, the bilinear forms of the weak form need to satisfy an inf-sup condition [23, pp. 153–154]. Note that in case of $\Gamma_{\mathrm{N}} = \emptyset$, the pressure p is only unique up to a constant [46, pp. 128–129]. Let $[V_h]^d \subseteq [V]^d$ and $Q_h \subseteq L^2(\Omega)$ be the discrete subspaces for the discrete velocity $\boldsymbol{u}_h \in [V_h]^d$ and the discrete pressure $p_h \in Q_h$. For the existence of discrete solutions $(\boldsymbol{u}_h, p_h)$, the spaces $[V_h]^d$ and Q_h need to form a uniformly inf-sup stable pairing [46, p. 133].

For solving the weak form (2.11) of the Stokes flow problem in core article II (Appendix A.2), we choose the uniformly inf-sup stable isogeometric subgrid element from [25]. In this space pairing, the velocity field is discretized on a subgrid of the pressure grid. Each element of the pressure mesh is subdivided into 2^d elements, yielding the velocity mesh. In article IV (Appendix B.1), because of data-structure limitations and performance reasons, we use equal order linear finite elements for the discretization of the weak form of (2.11). This space pairing is not uniformly inf-sup stable, therefore, we add a residual based stabilization term to the mass conservation equation [27, 64]. This stabilization introduces a consistency error which is at least of the same order as the discretization error [64, Theorem 4.1]. Moreover, in this article, we consider a nonlinear generalized Stokes flow in which the viscosity μ depends on the strain rate of the velocity modeled by a power law. This nonlinear problem is solved by performing a method of successive approximations, in particular Picard fixed-point iterations, which we describe in Section 3.3.

Furthermore, we consider various transient problems. As usual, the partial derivative of u with respect to the time t is denoted by $\dot{u}$. For instance, in core article I (Appendix A.1), we consider the $\mathfrak{p}$-Laplacian diffusion problem over the time interval $t \in [0, T]$, given in strong form as

$$\begin{aligned} \dot{u} - \operatorname{div}\left(\|\nabla u\|_2^{\mathfrak{p}-2}\nabla u\right) &= f \quad && \text{in } \Omega \times (0, T]\,, \\ u &= 0 && \text{on } \partial\Omega \times (0, T]\,, \\ u &= u_0 && \text{in } \Omega \times \{0\}\,, \end{aligned}$$

where f describes the source data and u_0 is the initial datum. For instance, one possible application of the $\mathfrak{p}$-Laplacian diffusion problem is in image processing [73]. For $\mathfrak{p} = 2$, the problem reduces to the classical heat equation for which the theory

is well-known [48, pp. 284–286], but for $\mathfrak{p} \neq 2$, the problem is nonlinear. If $\mathfrak{p} > 2$ and the data f and u_0 are regular enough, it may be shown that the problem has a unique weak solution [74, Chapter 2]. In core article I (Appendix A.1), we solve this nonlinear problem by performing Picard iterations in each time step using the FEM.

Moreover, we consider transient wave propagation problems in core article III (Appendix A.3). Let W be a differentiable energy density functional, $\rho_0 \colon \Omega \to \mathbb{R}_{>0}$ be a mass density function, and $\alpha, \beta \in \mathbb{C}$, $\alpha \neq 0$, two constants. The abstract wave propagation problem over the time interval $t \in [0, T]$ is given by

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{u}_0 && \text{at } t = 0, \\ \dot{\boldsymbol{u}} &= \boldsymbol{v}_0 && \text{at } t = 0, \\ \operatorname{Div} \partial_{\boldsymbol{u}} W(\boldsymbol{u}) + \boldsymbol{f} &= \rho_0 \ddot{\boldsymbol{u}} && \text{in } \Omega \times (0, T], \\ \alpha \boldsymbol{u} + \beta \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} &= \boldsymbol{g} && \text{on } \Gamma_{\mathrm{D}} \times (0, T], \\ \partial_{\boldsymbol{u}} W(\boldsymbol{u}) \boldsymbol{n} &= \boldsymbol{t} && \text{on } \Gamma_{\mathrm{N}} \times (0, T]. \end{aligned} \tag{2.12}$$

In wave propagation problems, we allow the functions $\boldsymbol{u}$ to take on complex values and the corresponding weak forms $a(\cdot, \cdot)$ are sesquilinear instead of bilinear. Such transient problems emerge in the modeling of elastic waves which may be used for the prediction of earthquakes and to nondestructively determine the structural integrity of engineering components [84]. In core article III (Appendix A.3), we consider the nonlinear response of a force impulse within a compressible neo-Hookean material by employing the energy density functional [85]

$$W(\boldsymbol{u}) = \frac{\lambda}{2} \ln(\det(\boldsymbol{F}(\boldsymbol{u})))^2 - \mu \ln(\det(\boldsymbol{F}(\boldsymbol{u}))) + \frac{\mu}{2} \big(\operatorname{tr}(\boldsymbol{F}(\boldsymbol{u})^{\mathsf{T}} \boldsymbol{F}(\boldsymbol{u})) - \operatorname{tr}(\boldsymbol{I}) \big), \tag{2.13}$$

where $\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{I} + \nabla \boldsymbol{u}$. The derivative of the energy density functional is given by

$$\partial_{\boldsymbol{u}} W(\boldsymbol{u}; \delta \boldsymbol{v}) = \Big[\lambda \ln(\det(\boldsymbol{F}(\boldsymbol{u}))) \boldsymbol{F}^{-\mathsf{T}}(\boldsymbol{u}) + \mu \Big(\boldsymbol{F}(\boldsymbol{u}) - \boldsymbol{F}(\boldsymbol{u})^{-\mathsf{T}} \Big) \Big] : \nabla \delta \boldsymbol{v}, \tag{2.14}$$

where ":" denotes the Frobenius inner product. We solve these systems using the nonlinear generalized-α method with underlying quasi-Newton–Raphson multicorrector steps described in Section 3.4.

When W is quadratic in $\boldsymbol{u}$, we may also define the time harmonic form of (2.12) for a wave number $k \in \mathbb{R}_{\geq 0}$ as follows:

$$\begin{aligned} -\operatorname{Div} \partial_{\boldsymbol{u}} W(\boldsymbol{u}) - k^2 \boldsymbol{u} &= \boldsymbol{f} && \text{in } \Omega, \\ \alpha \boldsymbol{u} + \beta \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} &= \boldsymbol{g} && \text{on } \Gamma_{\mathrm{D}}, \\ \partial_{\boldsymbol{u}} W(\boldsymbol{u}) \boldsymbol{n} &= \boldsymbol{t} && \text{on } \Gamma_{\mathrm{N}}. \end{aligned} \tag{2.15}$$

Also in core article III (Appendix A.3), we consider the time-harmonic linearized elastodynamic equations for compressible homogeneous and isotropic materials. The

energy density in this case is defined as

$$W(\boldsymbol{u}) = \frac{\lambda}{2} \operatorname{tr}(\boldsymbol{\varepsilon}(\boldsymbol{u}))^2 + \mu \boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{u}).$$

Note that the stiffness tensor $\boldsymbol{\sigma}$ of Hooke's law in (2.11) is related to W by $\boldsymbol{\sigma} = \partial_{\boldsymbol{\varepsilon}} W$. We consider a plate with a circular hole at which a periodic pressure is being applied to. At the boundaries where the waves are leaving the domain, we apply PML absorbing boundary conditions in order to avoid spurious reflections [17, 81].

Let $W(u) = 1/2 \nabla u^{\mathsf{T}} \nabla u$, $\alpha = -\mathrm{i}k$, $\beta = 1$, and $\Gamma_{\mathrm{D}} = \partial\Omega$. In this setting, (2.15) results in the Helmholtz equation with impedance boundary conditions:

$$\begin{aligned} -\Delta u - k^2 u &= f \quad \text{in } \Omega, \\ \frac{\partial u}{\partial \boldsymbol{n}} - \mathrm{i}ku &= g \quad \text{on } \partial\Omega. \end{aligned}$$

Under certain assumptions on the wave number k, the mesh width h, and the polynomial degree p of the basis functions of V_h, the discrete variational Helmholtz formulation has a unique solution $u_h \in V_h$ [49, Prop. 2.1]. More details on the assumptions may be found in core article III (Appendix A.3). In this article, we also extend the a priori error estimates of the surrogate matrix methodology to the Helmholtz equation and show that the additional consistency error is independent of the wave number k.

Moreover, we consider a fourth-order plate-bending problem in core article II (Appendix A.2) which may be naturally discretized using IGA since the higher-order basis functions are also of higher regularity in the absence of repeated knots. Similar a priori error estimates as for the model problem can be shown for this type of problem [98]. Also in core article II (Appendix A.2), we study the transverse vibrations of a two-dimensional isotropic elastic membrane and we provide a priori error estimates for the eigenvalue error in the presence of surrogate matrices.

In the next section, we briefly present the idea of how to construct surrogate matrices and the techniques we use to solve the linear and nonlinear systems of equations induced by the problems introduced in this section.

3. Solution techniques

In this section, we describe the techniques we use to solve the linear and nonlinear systems of equations induced by the problems presented in the previous section. Since the variational crime framework justifies the use of surrogate bilinear forms, we show the overall rough idea of constructing them by using the surrogate matrix methodology. Moreover, we provide some details on the linear and nonlinear solvers as well as the time integration schemes we use in the articles which are part of this thesis.

Let $a\colon V \times V \to \mathbb{R}$ be a continuous and coercive bilinear form and $F \in V^*$ a bounded linear functional. In this section, let V_h be either the finite element space (2.4) or the IGA space (2.6). Let $N = \dim(V_h)$ be the dimension of V_h and $\{\phi_i\}_{1\leq i\leq N}$ a basis of V_h. For IGA, the NURBS basis functions are a natural choice. For linear finite elements, the natural choice is the basis of nodal shape functions associated to the vertices of the mesh. For higher order finite elements, shape functions associated to the integration points may be used. Let $\mathsf{u} \in \mathbb{R}^N$ be the vector representation of u_h such that $u_h(\boldsymbol{x}) = \sum_{i=1}^N \mathsf{u}_i\, \phi_i(\boldsymbol{x})$. Henceforth, we implicitly switch between both representations where appropriate. We define $\mathsf{A} \in \mathbb{R}^{N\times N}$ and $\mathsf{f} \in \mathbb{R}^N$ via

$$\begin{aligned} [\mathsf{A}]_{ij} &= a(\phi_j, \phi_i) && \text{for } 1 \leq i, j \leq N, \\ [\mathsf{f}]_i &= F(\phi_i) && \text{for } 1 \leq i \leq N. \end{aligned}$$

If wave problems are considered, we allow the components of A, u, and f to be complex which requires implementations with support for complex data types. Since the basis functions only have small overlapping support, a large amount of entries in A are zero which means that the matrix is sparse. Solving the variational formulation (2.7a) is then equivalent to solving the matrix equation

$$\mathsf{A}\mathsf{u} = \mathsf{f}.$$

In the surrogate matrix methodology, we replace the matrix A by a surrogate matrix $\widetilde{\mathsf{A}}$ and define the corresponding surrogate bilinear form implicitly as $\tilde{a}(u_h, v_h) = \mathsf{v}^\mathsf{T}\widetilde{\mathsf{A}}\mathsf{u}$, where $\mathsf{v} \in \mathbb{R}^N$ and $v_h(\boldsymbol{x}) = \sum_{i=1}^N \mathsf{v}_i\, \phi_i(\boldsymbol{x})$. With this definition, we solve the variational equation (2.8) by solving the matrix equation

$$\widetilde{\mathsf{A}}\tilde{\mathsf{u}} = \mathsf{f}.$$

We construct $\widetilde{\mathsf{A}}$ such that assembling $\widetilde{\mathsf{A}}$ or computing the action of $\widetilde{\mathsf{A}}$ in a matrix-free method, is computationally much cheaper than using A instead. Provided that the constructed surrogate bilinear form induced by the surrogate matrix is coercive and satisfies (2.10), the estimates from Section 2.5 theoretically justify the usage of surrogate matrices.

The matrix A often inherits properties of the corresponding bilinear form $a(\cdot,\cdot)$. In case of the model problem (2.1), the resulting matrix is symmetric and positive definite. In core article I (Appendix A.1) and core article II (Appendix A.2), we show that some of these properties like symmetries and kernels can be maintained by the surrogate matrix.

3.1. Matrix assembly, matrix-free methods, and surrogate matrices

The global matrix A is usually assembled by contributions of dense local element matrices $\mathsf{A}_T \in \mathbb{R}^{N_T \times N_T}$ for $T \in \mathcal{T}_h$ where N_T is the number of basis functions having support in an element T. For instance, in an IGA discretization of order p, the number of basis functions having support in an element is $N_T = (p+1)^d$ if uniform knot vectors are used. Let $\mathsf{P}_T \in \mathbb{R}^{N \times N_T}$ be the operator mapping the coefficients of the local basis functions to the corresponding coefficients in the global vector. In practice, the operator P_T is not assembled since its action is computed on-the-fly through index computations. Formally, the assembly of the global matrix A may be written as

$$\mathsf{A} = \sum_{T \in \mathcal{T}_h} \mathsf{P}_T \mathsf{A}_T \mathsf{P}_T^\intercal.$$

The entries of A_T are computed by numerical integration. Let $\mathcal{Q}_T$ be a set of integration points and weights corresponding to an element $T \in \mathcal{T}_h$. For instance, the entries of A_T for the model problem from Section 2.2 may be computed via

$$\begin{aligned}[\mathsf{A}_T]_{ij} &= \int_T (\nabla \phi_j^{(T)})^\intercal K \nabla \phi_i^{(T)} \,\mathrm{d}\boldsymbol{x} \\ &\approx \sum_{(w_q^{(T)}, \boldsymbol{x}_q^{(T)}) \in \mathcal{Q}_T} w_q^{(T)} \, \nabla \phi_j^{(T)}(\boldsymbol{x}_q^{(T)})^\intercal K(\boldsymbol{x}_q^{(T)}) \nabla \phi_i^{(T)}(\boldsymbol{x}_q^{(T)}), \end{aligned} \tag{3.1}$$

where $\phi_i^{(T)}$ and $\phi_j^{(T)}$ are element-local basis functions. In practice, however, these integrals are computed on a reference element $\widehat{T}$ with integration weights and points $\widehat{\mathcal{Q}}$, and maps $\boldsymbol{\varphi}_T \colon \widehat{T} \to T$ with $\det(D\boldsymbol{\varphi}_T) > 0$. Let $\phi^{(T)} = \widehat{\phi} \circ \boldsymbol{\varphi}_T^{-1}$, $\nabla \phi^{(T)} = D\boldsymbol{\varphi}_T^{-\intercal} \widehat{\nabla} \widehat{\phi}$, and $w_q^{(T)} = \widehat{w}_q \det(D\boldsymbol{\varphi}_T(\widehat{\boldsymbol{x}}_q))$. Equation (3.1) may thus be rewritten as

$$[\mathsf{A}_T]_{ij} = \sum_{(\widehat{w}_q, \widehat{\boldsymbol{x}}_q) \in \widehat{\mathcal{Q}}} \widehat{w}_q \, \widehat{\nabla} \widehat{\phi}_j(\widehat{\boldsymbol{x}}_q)^\intercal D\boldsymbol{\varphi}_T^{-1} K(\boldsymbol{\varphi}_T(\widehat{\boldsymbol{x}}_q)) D\boldsymbol{\varphi}_T^{-\intercal} \widehat{\nabla} \widehat{\phi}_i(\widehat{\boldsymbol{x}}_q) \det(D\boldsymbol{\varphi}_T(\widehat{\boldsymbol{x}}_q)).$$

The advantage of performing the numerical integration on a reference element is that the basis functions and its derivatives can be precomputed at the integration points and stored in memory. Only the values corresponding to the Jacobian of $\boldsymbol{\varphi}_T$, its determinant, as well as the coefficient values K need to be recomputed for each $T \in \mathcal{T}_h$. Performing the operations for the whole matrix at once instead of componentwise, allows for a more efficient vectorized implementation. Additionally, if the basis functions have a tensor-product structure, the sums over the single dimensions may be reorganized such that the number of FLOPs is minimized. This is known as sum factorization. In IGA, for instance, using standard Gauss integration loops requires about $\mathcal{O}(p^{3d})$ FLOPs to form A_T, but using sum factorization instead, the complexity may be reduced to $\mathcal{O}(p^{2d+1})$ [5, 26]. Alternatively, weighted integration rules which construct individual integration rules for each test function can reduce the number of integration points per element from $\mathcal{O}(p^d)$ to only $\mathcal{O}(1)$ [29, 92]. In return, using such a rule reduces the complexity of assembling A_T from $\mathcal{O}(p^{3d})$ to $\mathcal{O}(p^{2d})$.

In this thesis, however, we focus on a different approach based on stencils. There, one does not iterate over the elements of the mesh, but over the rows of the global

sparse matrix: $\mathsf{a}_i^\mathsf{T} = ([\mathsf{A}]_{i1}, [\mathsf{A}]_{i2}, \dots, [\mathsf{A}]_{iN})$ for $1 \leq i \leq N$. Row-wise approaches are not yet popular in IGA, but the recent work in [61] illustrates their computational advantages by combining sum factorization and weighted integration. However, we go one step further and exploit some structure in the basis functions. Because of the sparsity of A, each row may be written as

$$\mathsf{a}_i^\mathsf{T} = \mathsf{s}_i^\mathsf{T} \mathsf{R}_i \qquad \text{for all } 1 \leq i \leq N,$$

where $\mathsf{s}_i \in \mathbb{R}^{N_i}$ is called *stencil*, N_i is the number of non-zeros in the i-th row of A, and $\mathsf{R}_i \in \mathbb{R}^{N_i \times N}$ is a restriction matrix which removes the entries in the global vector belonging to the zero entries of a_i^T. Similarly to P_T, the R_i will not be assembled since their actions may be computed through on-the-fly index calculations. If locally-structured meshes are used, the majority of rows possess the same sparsity pattern. In IGA, the meshes display this property for each patch if the corresponding knot vectors are uniform. In the FEM this is true if, for instance, HHGs are employed [18, 20, 54]. HHGs are constructed by starting with a globally unstructured macro-mesh which is used to triangulate the input geometry. Each macro-element in the macro-mesh is uniformly refined for a large number of times, resulting in a locally-structured mesh. The sparsity pattern for rows belonging to basis functions within the macro-elements is the same. If additionally the coefficient and the Jacobian of the geometry map are constant, the stencil components are constant as well in the structured parts of domain.

In the following, we briefly present the idea of the surrogate matrix methodology and the construction of surrogate matrices. The rigorous and detailed description of this methodology is beyond the scope of this subsection. For this purpose, we refer to core article I (Appendix A.1) in the case of finite elements and to core article II (Appendix A.2) and core article III (Appendix A.3) in the case of IGA. In these articles, we show under certain assumptions on the geometry and the PDE coefficient that it is possible to identify the stencil components in a structured part of the domain by a small set of locally smooth functions, called *stencil functions*, $\Phi_\delta \colon \widetilde{\Omega} \to \mathbb{R}$, $1 \leq \delta \leq \mathcal{D}$, with $\mathcal{D} \ll N$ and $\widetilde{\Omega} \subsetneq \Omega$. Let $\mathbb{I} \subseteq \{1, 2, \dots, N\}$ be the index set of stencils for which this identification by stencil functions is possible. In this subsection, we restrict ourselves to a single set $\mathbb{I}$, but depending on the number of locally-structured parts of the domain, there may be a family of index sets $\{\mathbb{I}_k\}_k$ with a set of stencil functions associated to each of them. The number of required stencil functions $\mathcal{D}$ associated to the set $\mathbb{I}$ depends on the discretization and dimension. We show in core article I (Appendix A.1) that for a linear finite element discretization on two dimensional HHGs, the general number of required stencil functions for a macro element is $\mathcal{D} = 7$. In core article II (Appendix A.2), we show that in the interior of a single IGA patch constructed with uniform knot vectors, this number is $\mathcal{D} = (2p+1)^d$. By enforcing symmetry and preserving the original matrix kernels, these numbers may be further reduced.

This identification allows encoding large parts of the global matrix by just the stencil functions Φ_δ. Let $\widetilde{\mathbb{X}} \subseteq \widetilde{\Omega}$ be the set of points corresponding to the index set $\mathbb{I}$, at which the stencil functions are evaluated. With these definitions, the stencil

entries may be identified by

$$\mathsf{s}_i^{\mathsf{T}} = (\Phi_1(\widetilde{\boldsymbol{x}}_i), \dots, \Phi_{\mathcal{D}}(\widetilde{\boldsymbol{x}}_i)) \qquad \text{for } \widetilde{\boldsymbol{x}}_i \in \widetilde{\mathbb{X}}.$$

We show that evaluating a stencil function corresponds to performing integration over the common support of ϕ_i, $i \in \mathbb{I}$, and some ϕ_j, $1 \leq j \leq N$, with $\operatorname{supp}\{\phi_i\} \cap \operatorname{supp}\{\phi_j\} \neq \emptyset$. Our goal is to reduce over-computation by avoiding the expensive numerical integration for almost all $\widetilde{\boldsymbol{x}}_i \in \widetilde{\mathbb{X}}$. Therefore, we choose a small set of sampling points $\widetilde{\mathbb{X}}^s \subseteq \widetilde{\mathbb{X}}$. The maximum pairwise distance between the points in $\widetilde{\mathbb{X}}^s$ defines the coarse scale H. At each of the sampling points $\widetilde{\boldsymbol{x}}_{i^s} \in \widetilde{\mathbb{X}}^s$, we compute the standard stencils s_{i^s} by performing numerical integration. Note that the numerical integration may still be performed using one of the state of the art reduced integration schemes from Section 1.2, to further avoid over-computation.

We use these sampling points to construct an interpolation or projection operator Π_H yielding approximations $\widetilde{\Phi}_\delta = \Pi_H \Phi_\delta$. We denote the resulting approximations as *surrogate stencil functions* $\widetilde{\Phi}_\delta$. For the approximation spaces in IGA discretizations, we choose the natural B-spline space of order q with knot vectors corresponding to the sampling points. In case of the FEM, we choose polynomial basis functions of order q on some arbitrary bounded reference domains. These are just natural choices and any other possible approximation space may be used. It is important to note that performing the interpolations or projections using these spaces and evaluating the approximated functions for all $\widetilde{\boldsymbol{x}}_i \in \widetilde{\mathbb{X}}$ is computationally cheaper than performing standard integration for all $\widetilde{\boldsymbol{x}}_i \in \widetilde{\mathbb{X}}$. We conduct a floating point complexity analysis in core article III (Appendix A.3) which establishes that the computational complexity of the methodology compares favorably to other present fast assembly techniques for isogeometric methods. The surrogate stencil $\tilde{\mathsf{s}}_i$ is then defined as

$$\tilde{\mathsf{s}}_i^{\mathsf{T}} = (\widetilde{\Phi}_1(\widetilde{\boldsymbol{x}}_i), \dots, \widetilde{\Phi}_{\mathcal{D}}(\widetilde{\boldsymbol{x}}_i)) \qquad \text{for } \widetilde{\boldsymbol{x}}_i \in \widetilde{\mathbb{X}}.$$

In the extreme case $H = h$, all the stencils are computed using standard integration, thus the surrogate and standard stencils coincide. It can be shown that for particular coefficients, geometry maps, and for q large enough, the surrogate stencil functions and the original stencil functions are the same, even for $H > h$; see Corollary 8.1 and Corollary 10.1 in core article II (Appendix A.2).

This identification of the stencil entries with a set of locally smooth functions depends strongly on locally-structured meshes as they are obtained by uniform refinement or by using uniform knot vectors. Another limitation is that the geometry maps and coefficients in the PDE need to be regular enough such that the stencil functions may be approximated well by functions of higher order. If the coefficient or geometry map is only locally smooth in sub-patches of a locally structured part, this limitation may be resolved by carefully choosing the interpolation spaces. For instance, if the locations of the irregularities are known beforehand, the knot vectors of a B-spline interpolation may be chosen accordingly.

Using these surrogate stencil functions to approximate the entries of A results in the so called surrogate matrix $\widetilde{\mathsf{A}}$. In core article II (Appendix A.2) and core article III (Appendix A.3), we present theoretical error estimates for the additional consistency

error introduced by this approach in an IGA discretization. For numerical verification of these estimates and for performance benchmarking, the method is implemented in MATLAB with the software library GeoPDEs [51, 102]. A reference implementation with details is presented in the accompanying article [41]. We assemble the global sparse matrix by iterating over all rows which can be described by stencil functions and writing the row and column indices as well as the evaluated surrogate stencil function values into a vector each. The missing rows corresponding to stencils which may not be identified by stencil functions are assembled using standard numerical integration and are also added to these vectors. This sparse matrix representation is known as the coordinate list (COO) format which is often used to efficiently construct sparse matrices. Here, values with the same indices are added up. Afterwards, the matrix present in the COO format is compressed into one of the standard sparse matrix formats: compressed row storage (CRS) or compressed column storage (CCS) [9, pp. 64–65]. Since arrays in MATLAB are stored in column-major order, the sparse matrices are internally stored in the CCS format. Using the surrogate matrices in core article II (Appendix A.2), at just over one million degrees of freedom, our experiments demonstrate assembly speed-ups beyond fifty times with a simple second-order NURBS basis and without any significant loss in the accuracy of the surrogate solution.

In matrix-free methods, however, the global matrices are not assembled. Instead, everything needed for the MVP is computed on-the-fly or by using some small stored data to speed up the evaluation. These data may include, for instance, the reference basis functions pre-evaluated at the integration points. In the predominant element-by-element matrix-free approach [6, 21, 30, 53, 89], the MVP is computed by evaluating

$$\mathsf{Au} = \sum_{T \in \mathcal{T}_h} \mathsf{P}_T(\mathsf{A}_T(\mathsf{P}_T^\mathsf{T}\mathsf{u})),$$

where the product of A_T and $\mathsf{P}_T^\mathsf{T}\mathsf{u}$ is computed on-the-fly without assembling A_T. Many articles address the optimization of this approach and its implementation on accelerators like GPUs [28, 52, 70, 71, 72, 75, 76, 77, 82]. Sum factorization may be utilized in the matrix-free approach as well [11, 88, 92]. It can be shown that using these approaches yields a better performance over matrix-based approaches if higher order discretizations are utilized [71].

Stencil-based methods are also very well suited for matrix-free methods. The action of the global stiffness matrix is computed by iterating over the rows of the stiffness matrix and computing scalar products with the right-hand side vector, yielding a new value at a single degree of freedom. Blocking techniques can be used in vector-valued PDEs to process coefficients belonging to the same basis function simultaneously. Using the stencil, an MVP may be written as

$$[\mathsf{Au}]_i = \mathsf{s}_i^\mathsf{T}(\mathsf{R}_i\mathsf{u}) \qquad \text{for } 1 \leq i \leq N.$$

In this scenario, the MVP with the surrogate matrix may be computed by replacing

the stencils by the surrogate stencils

$$[\widetilde{\mathsf{A}}\mathsf{u}]_i = \begin{cases} \tilde{\mathsf{s}}_i^{\mathsf{T}}(\mathsf{R}_i\mathsf{u}) & \text{if } i \in \mathbb{I}, \\ \mathsf{s}_i^{\mathsf{T}}(\mathsf{R}_i\mathsf{u}) & \text{otherwise,} \end{cases} \qquad \text{for } 1 \leq i \leq N,$$

in which the surrogate stencil functions are evaluated on-the-fly. In core article I (Appendix A.1), we show theoretical error estimates for the additional consistency error introduced by this approach when using a linear finite element discretization. For numerical verification of these estimates and performance benchmarking, the method is implemented in the massively parallel matrix-free finite element framework HyTeG [69]. The reported speed-ups for all tested examples, when evaluating the surrogate stencils on-the-fly instead of performing the numerical integration, range between a factor of 14 and 20.

In article IV (Appendix B.1), we follow another approach to construct the surrogate matrix without using stencil functions. There, we utilize a concept called *stencil scaling* which we initially introduced for scalar elliptic PDEs in [14]. The approach is based on scaling reference stencils $\hat{\mathsf{s}}_i$ from a constant coefficient PDE, e.g. $K = \boldsymbol{I}$ in the model problem from Section 2.2, and where the geometry is described by an affine linear function. If the same PDE with a non-constant coefficient is solved, the components of $\hat{\mathsf{s}}_i$ are scaled by linear combinations of the coefficient. In that work, we showed a priori error estimates and performed numerical benchmarks validating the theoretical results and the performance. In article IV (Appendix B.1), we extend this idea to vector-valued PDEs. We show that the simple scaling for vector-valued PDEs results in the discretization of a different PDE if the coefficient is not constant. In order to overcome this issue, we develop a new stencil scaling approach by adding a correction term to the discrete stencils. We present how to efficiently pre-compute these correction terms in 2D and 3D, respectively. Moreover, we provide numerical verifications with theoretical and practical performance considerations. We can show that matrix-free methods may yield a better performance than matrix-based approaches, even for linear finite element discretizations, when combined with stencil scaling techniques. By using this approach on the older SuperMUC Phase 2 supercomputer, we could observe maximum speed-ups of 122% and on the state of the art supercomputer SuperMUC-NG, we can observe speed-ups of 64% compared to the on-the-fly integration, without any significant loss in the accuracy of the solution.

In the next subsection, we briefly review the solvers we use to solve the linear systems described by A and $\widetilde{\mathsf{A}}$.

3.2. Direct and iterative solvers

In the articles which are part of this thesis, we use various methods for solving the emerging linear systems. In this and the following subsections, the matrix A denotes either the operator obtained by numerical integration or the surrogate matrix. We mainly distinguish between direct and iterative solvers. Depending on the purpose and the properties of A, we decide which method to use.

Direct methods do not require any assumptions on A other than invertibility. They are based on a pivoted Gaussian elimination which may be optimized when applied to sparse matrices. In the general case, however, computing the factorization of A requires a lot of extra memory due to the resulting fill-in which becomes worse the higher the dimension d is. This disadvantage makes direct solvers impractical for solving very large linear systems. We show in core article II (Appendix A.2) that, in two dimensions, the assembly process of IGA matrices may still be more time consuming than solving the system with a direct solver. By using the surrogate matrix methodology this discrepancy may be reduced. Another disadvantage of direct solvers is that they can only be employed when the matrix is available in assembled form.

If this is not the case, iterative solvers which only require MVPs may be used. Here, we focus on Krylov subspace and multigrid methods which are two popular examples of iterative solvers. The matrix A from the discretized model problem in Section 2.2 is symmetric and positive definite. For solving the linear system associated to A, these properties can be exploited by the conjugate gradient (CG) Krylov subspace method [46, Chapter 2] to construct an efficient iterative solver. However, its convergence rate depends on the condition of the system matrix which becomes worse when the mesh is refined. The counterpart to the conjugate gradient method for symmetric but indefinite matrices, is the minimal residual (MINRES) Krylov method [46, Chapter 4]. In order to improve on the convergence of Krylov subspace methods, they may be applied to the left-preconditioned system

$$\mathsf{P}^{-1}\mathsf{A}\mathsf{u} = \mathsf{P}^{-1}\mathsf{f},$$

where P^{-1} is called the preconditioner. Finding suitable preconditioners is problem dependent and a lot of work deals with finding efficient preconditioners. For matrix-based methods, incomplete factorizations which approximate full factorizations as they are obtained by Gaussian elimination, may be used [46, Chapter 2]. For matrix-free methods, preconditioners based on fast diagonalization exploiting the tensor-product structure of the basis functions were developed in [88, 92].

Another large class of preconditioners is based on multigrid methods [6, 11, 18, 19, 52, 54, 71, 72, 76, 82, 99]. Multigrid methods are optimal iterative solvers in the sense that their theoretical FLOP complexity is proportional to the number of degrees of freedom, i.e., $\mathcal{O}(N)$. In geometric multigrid methods, we require a hierarchy of nested spaces $V_1 \subseteq V_2 \subseteq \cdots \subseteq V_L$. The HHGs used in core article I (Appendix A.1) and article IV (Appendix B.1) provide a natural family of hierarchically nested meshes $\{\mathcal{T}_{\widehat{H}}, \mathcal{T}_{2^{-1}\widehat{H}}, \ldots, \mathcal{T}_{2^{-L+1}\widehat{H}}\}$, where $\widehat{H} = 2^{L-1}h$ is the mesh size of the coarse input mesh and $L \in \mathbb{N}$ the number of levels in the hierarchy. We naturally define the space V_ℓ, $1 \leq \ell \leq L$, to be the finite element space related to the mesh $\mathcal{T}_{2^{-\ell+1}\widehat{H}}$. Let I_ℓ, $1 \leq \ell < L$, be the operator transferring the solution from V_ℓ to $V_{\ell+1}$. Since the spaces are conforming and nested, we use the natural interpolation of a coarse grid function to the finer grid, but other choices based on projections are also possible.

Assuming that the matrix on the finest level L is assembled, the matrices on the coarser levels A_ℓ, $1 \leq \ell < L$, may be recursively defined via $\mathsf{A}_\ell = \mathsf{I}_\ell^{\mathsf{T}}\mathsf{A}_{\ell+1}\mathsf{I}_\ell$ for $1 \leq \ell < L$. In the literature, this definition is known as the Galerkin projection. In

matrix-free methods, however, this definition is impractical since the MVP with the fine level A_L matrix would be required to obtain the action of a coarser level matrix. Therefore, we repeatedly discretize the problem on each V_ℓ independently to obtain the operators A_ℓ for $1 \leq \ell < L$.

Let S_ℓ, $1 < \ell \leq L$, be smoothers satisfying the smoothing property [46, Chapter 2]. A typical choice is the damped Jacobi smoother $\mathsf{S}_\ell = \frac{1}{\omega}\mathsf{D}_\ell$, where D_ℓ is a diagonal matrix with the diagonal entries of A_ℓ and $\omega \in \mathbb{R}$ a damping factor. Storing D_ℓ requires only the space of an additional solution vector and it can be easily computed when using the element-by-element approach. Polynomial Chebyshev accelerated smoothers can be used to improve the smoothing properties of the Jacobi smoother when only the diagonals of A_ℓ are available [1]. Gauss–Seidel smoothers are not feasible with the element-by-element approach since they require all the entries of the matrix rows and not just the diagonal value. In stencil-based methods, however, the matrix rows are available and thus the Gauss–Seidel smoother may be utilized. Since in our parallel HyTeG solver not all the dependencies across process boundaries can be kept synchronized, we utilize hybrid Gauss–Seidel methods in which some degrees of freedom are processed in a Jacobi smoother fashion. Let $\nu_\ell \in \mathbb{N}$ be the pre- and postsmoothing steps on level ℓ. With these definitions in mind, the multigrid V-cycle for solving the system $\mathsf{A}_L\mathsf{u}_L = \mathsf{f}_L$ is presented in Algorithm 1. Under the assumptions of Theorem 1 in the H^2 regular setting, the successive application of the multigrid V-cycle with a damped Jacobi smoother, is a suitable iterative solver for the model problem [46, Theorem 2.15]. Note that the matrix on the coarsest level used in line 3 of Algorithm 1 may be assembled and inverted by a direct solver. Alternatively, an iterative solver may be used to avoid over-solving the coarse problem. The latter approach also allows to utilize matrix-free methods.

Algebraic multigrid methods constitute the counterpart to geometric multigrid methods. There, the restriction operators I_ℓ^T are constructed algebraically without requiring any mesh information and the coarse grid matrices are constructed by Galerkin projections. These methods use only information of the matrix sparsity and its entries [46, Chapter 2]. Since they require an assembled matrix, they render very useful for the coarse grid problem in line 3 of Algorithm 1, where the matrices are small. In large parallel applications, however, one needs to ensure that the process local matrices do not get to small or otherwise the communication overhead impedes a good parallel scalability.

In core article I (Appendix A.1), we directly apply geometric multigrid V-cycles to solve the linear systems. As the smoother, we choose a hybrid Gauss–Seidel smoother which is naturally feasible with stencil-based methods. On the coarsest level, we either use a diagonally preconditioned CG method or the direct multifrontal massively parallel sparse direct solver (MUMPS) [3, 4]. For improved parallel scalability of the coarse grid solver, agglomeration techniques as provided by `PCTELESCOPE` [83] are used in computations with many processes.

In core article II (Appendix A.2), we use direct solvers for the accuracy benchmarks of the surrogate matrix methodology for all the two-dimensional problems. For Poisson's problem in three dimensions, however, we employ the preconditioned CG method with HYPRE's BoomerAMG algebraic multigrid as the preconditioner [59].

Algorithm 1 Multigrid V-cycle

function V-CYCLE(u_ℓ, f_ℓ)
  if $\ell = 1$ **then**
    Solve $\mathsf{A}_\ell \mathsf{u}_\ell = \mathsf{f}_\ell$ ▷ Direct or iterative solve
  else
    For ν_ℓ steps $\mathsf{u}_\ell \leftarrow \mathsf{u}_\ell + \mathsf{S}_\ell^{-1}(\mathsf{f}_\ell - A_\ell \mathsf{u}_\ell)$ ▷ Presmoothing
    $\mathsf{r}_{\ell-1} \leftarrow \mathsf{I}_{\ell-1}^{\mathsf{T}}(\mathsf{f}_\ell - \mathsf{A}_\ell \mathsf{u}_\ell)$ ▷ Restrict residual
    $\mathsf{u}_{\ell-1} \leftarrow \mathbf{0}$
    $\mathsf{u}_{\ell-1} \leftarrow$ V-CYCLE($\mathsf{u}_{\ell-1}$, $\mathsf{r}_{\ell-1}$) ▷ Recursive coarse grid correction
    $\mathsf{u}_\ell \leftarrow \mathsf{u}_\ell + \mathsf{I}_{\ell-1}\mathsf{u}_{\ell-1}$ ▷ Add coarse grid correction
    For ν_ℓ steps $\mathsf{u}_\ell \leftarrow \mathsf{u}_\ell + \mathsf{S}_\ell^{-\mathsf{T}}(\mathsf{f}_\ell - A_\ell \mathsf{u}_\ell)$ ▷ Postsmoothing
  end if
  return u_ℓ
end function

In core article III (Appendix A.3), we also use direct solvers provided by MATLAB for all problems in two dimensions. Since the discretized Helmholtz equation is indefinite for larger wave numbers and multigrid solvers do not work as efficiently in this case, we use the MUMPS for solving the Helmholtz equation in three dimensions.

In article IV (Appendix B.1), for solving the elliptic problems, we also directly apply geometric multigrid V-cycles with a hybrid Gauss–Seidel smoother and a preconditioned CG method on the coarsest level. The discretization of Stokes flow problems results in a saddle point problem. In [39], various block-smoothers for saddle points were analyzed, and in [55], a quantitative performance study for massively parallel iterative Stokes solvers was performed. The inexact Uzawa block smoother with variable V-cycles, i.e., adding two more smoothing steps per coarser level correction, rendered itself the most efficient solver compared to a block preconditioned MINRES Krylov subspace method and a Schur complement CG method. Therefore, we utilize the same Uzawa-type solver for solving the Stokes problems discussed in article IV (Appendix B.1). Utilizing this solver, the largest considered Stokes system with about $1.03 \cdot 10^{11}$ degrees of freedom is solved in little more than three minutes on 12 288 compute cores of SuperMUC-NG.

In the next subsection, we present how these linear solvers are employed for solving the nonlinear problems from Section 2.6.

3.3. Nonlinear problems

Some of the problems presented in Section 2.6 require solving a nonlinear problem. Many methods for solving nonlinear problems need to reassemble the discrete problem in every iteration. In such cases, the surrogate matrix methodology may be used to speed up the reassembly and thus reducing the computing time.

For instance, the weak form of the stationary part of the $\mathfrak{p}$-Laplacian in Section 2.6 may be written in semilinear form as

$$a(w; u, v) = \int_\Omega \|\nabla w\|_2^{\mathfrak{p}-2} \nabla u^{\mathsf{T}} \nabla v \, \mathrm{d}\boldsymbol{x}.$$

The corresponding discrete nonlinear problem reads:

$$\text{Find } u_h \in V_h \text{ satisfying} \qquad a(u_h; u_h, v_h) = F(v_h) \qquad \text{for all } v_h \in V_h,$$

and the corresponding nonlinear matrix equation is

$$\mathsf{A}(\mathsf{u})\mathsf{u} = \mathsf{f}. \tag{3.2}$$

The semilinear form $a(\cdot;\cdot,\cdot)$ is linear in the last two arguments and thus provides a linearization of the nonlinear problem. These problems may be solved using Picard iterations shown in Algorithm 2. Assuming that the function $R(\mathsf{u}) = \mathsf{A}(\mathsf{u})^{-1}\mathsf{f}$ is a contraction, the Banach fixed-point theorem asserts that the sequence generated by Algorithm 2 converges to a unique solution of (3.2) [105, p. 166]. We utilize this

Algorithm 2 Picard iterations

1: Initialize $\mathsf{u}^{(0)}$, let $n = 1$, and let *tol* be a fixed relative tolerance
2: **while** true **do**
3: Let $[\mathsf{A}]_{ij} = a(u_h^{(n-1)}; \phi_j, \phi_i)$ for $1 \leq i, j \leq N$
4: Solve the linear system $\mathsf{A}\mathsf{u}^{(n)} = \mathsf{f}$ for $\mathsf{u}^{(n)}$
5: **if** $\|\mathsf{u}^{(n)} - \mathsf{u}^{(n-1)}\|_2 < tol \cdot \|\mathsf{u}^{(0)}\|_2$ **then**
6: **return** $\mathsf{u}^{(n)}$
7: **end if**
8: $n \leftarrow n + 1$
9: **end while**

iterative scheme in core article I (Appendix A.1) for solving the parabolic $\mathfrak{p}$-Laplacian described in Section 2.6. In article IV (Appendix B.1), we apply the same scheme to solve the nonlinear generalized Stokes problem in which the viscosity depends on the strain rate and thus implicitly on the velocity. In both applications, the matrix A in line 3 of Algorithm 2 is not being assembled and only its action to a vector is computed, using either numerical integration or the surrogate matrix methodology. The systems in line 4 of Algorithm 2 are only solved approximately down to a fixed relative tolerance in order to avoid over-solving. In case of the parabolic $\mathfrak{p}$-Laplacian, we perform standard multigrid V-cycles and in case of the nonlinear generalized Stokes problem, we perform variable multigrid V-cycles with an inexact Uzawa block smoother.

The Newton–Raphson method is another method to solve nonlinear systems. It usually displays a better convergence rate compared to Picard's iterations from above, but convergence to a solution is only guaranteed if the initial guess is already close to the actual solution and if the tangent matrices emerging during the iterative process are invertible [86, pp. 183–185].

Let R be a residual function and the corresponding problem:

$$\text{Find } u_h \in V_h \text{ satisfying} \qquad R(u_h; v_h) = 0 \qquad \text{for all } v_h \in V_h.$$

For instance, the residual for a scalar stationary nonlinear wave propagation problem (see Section 2.6) reads

$$R(u_h;\delta v) = \int_\Omega \partial_u W(u_h;\delta v) - f\,\delta v \,\mathrm{d}\boldsymbol{x} \qquad \text{for all } \delta v \in V_h.$$

Linearizing R at $u_h \in V_h$ in direction $\delta u \in V_h$ yields

$$R(u_h+\delta u;\delta v) = R(u_h;\delta v) + \partial_u R(u_h;\delta v,\delta u) + o(\|\delta \mathsf{u}\|_2),$$

where the derivative of R with respect to u in direction $\delta u \in V_h$ reads

$$\partial_u R(u_h;\delta v,\delta u) = \int_\Omega \partial_u^2 W(u_h;\delta v,\delta u)\,\mathrm{d}\boldsymbol{x} \qquad \text{for all } \delta v \in V_h.$$

Given a $u_h \in V_h$ and neglecting the higher order terms, we search for a solution $\delta u \in V_h$ of

$$R(u_h+\delta u;\delta v) \approx R(u_h;\delta v) + \partial_u R(u_h;\delta v,\delta u) = 0 \qquad \text{for all } \delta v \in V_h,$$

and let $u_h \leftarrow u_h + \delta u$. Repeating this process iteratively is known as the Newton–Raphson method shown in Algorithm 3. In the Newton–Raphson method, the tangent

Algorithm 3 Newton–Raphson method

1: Initialize $\mathsf{u}^{(0)}$, let $n=1$, and let *tol* be a fixed relative tolerance
2: **while** true **do**
3: Compute the residual vector $[\mathsf{r}]_i = R(u_h^{(n-1)};\phi_i)$ for $1 \le i \le N$
4: Let A be the tangent matrix $[\mathsf{A}]_{ij} = \partial_u R(u_h^{(n-1)};\phi_i,\phi_j)$ for $1 \le i,j \le N$
5: Solve the linear system $\mathsf{A}\delta\mathsf{u} = -\mathsf{r}$ for $\delta\mathsf{u}$
6: Let $\mathsf{u}^{(n)} = \mathsf{u}^{(n-1)} + \delta\mathsf{u}$
7: **if** $\|\delta\mathsf{u}\|_2 < tol \cdot \|\mathsf{u}^{(0)}\|_2$ **then**
8: **return** $\mathsf{u}^{(n)}$
9: **end if**
10: $n \leftarrow n+1$
11: **end while**

matrix needs to be reassembled in each iteration; see line 4 in Algorithm 3. When using the surrogate matrix methodology to speed up the assembly, only an approximation of the true tangent matrix is obtained. Therefore, the Newton–Raphson method used in conjunction with the surrogate methodology may be interpreted as a quasi-Newton method. Since the residual is computed using standard numerical integration, the consistency error $u_h - \tilde{u}_h$ will vanish with the number of Newton–Raphson iterations.

The Newton–Raphson method may also be utilized within a predictor multicorrector algorithm to solve transient nonlinear problems. We present this application in the following subsection.

3.4. Nonlinear time integration

The surrogate matrix methodology can also be used in implicit time integration schemes for assembling the propagation matrices which propagate the solution forward in time. For nonlinear transient problems, the propagation matrices need to be reassembled multiple times in each time step of the simulation. This is due to the dependence of the tangent matrix on the solution of the previous time steps and on the iterates of the current time step. By employing the surrogate matrix methodology to speed up the reassembly of the propagation matrices, the computing times may be significantly reduced.

In core article III (Appendix A.3), we solve a compressible hyperelastic wave problem modeled by (2.12) utilizing the energy functional (2.13) describing a neo-Hookean medium through which the waves propagate. For this purpose, the system of equations (2.12) may be discretized in time like a second order ordinary differential equation in which a nonlinear PDE needs to be solved in each time step.

For solving the semi-discrete transient problem (2.12), we employ the nonlinear generalized-α time integration scheme proposed in [32]. Here, we closely follow the algorithm presented in [34, Section 7.3]. Let $\boldsymbol{u}_h^{(n)}$, $\dot{\boldsymbol{u}}_h^{(n)}$, $\ddot{\boldsymbol{u}}_h^{(n)} \in [V_h]^d$ be the displacement, velocity, and acceleration at the current time t_n. We identify a function $\boldsymbol{u}_h$ with the vector $\mathbf{u} \in \mathbb{R}^{d \cdot N}$ through $\boldsymbol{u}_h = \sum_{i=1}^{N} \sum_{\ell=1}^{d} \mathbf{u}_{d(i-1)+\ell} \phi_i \boldsymbol{e}_\ell$, where $\boldsymbol{e}_\ell$ is the unit vector with a 1 in the ℓ-th component and 0 elsewhere. The same identification is made for the velocity and acceleration, respectively. The goal is finding $(\boldsymbol{u}_h^{(n+1)}, \dot{\boldsymbol{u}}_h^{(n+1)}, \ddot{\boldsymbol{u}}_h^{(n+1)})$ at the next time t_{n+1} satisfying the time-discrete variant of system (2.12). Let $\Delta t = t_{n+1} - t_n$ denote the time step size, and let $\alpha_m = \frac{1}{2}\frac{3-\rho_\infty}{1+\rho_\infty}$, and $\alpha_f = \frac{1}{1+\rho_\infty}$ be the parameters from [68]. Moreover, let $\gamma = \frac{1}{2} - \alpha_f + \alpha_m$ and $\beta = \frac{1}{4}(1-\alpha_f+\alpha_m)^2$. With $\rho_\infty = \frac{1}{2}$, this choice of parameters yields an unconditionally stable second-order time integration scheme while introducing some damping of the high-frequencies [34, pp. 203–204]. The residual function R corresponding to (2.12) with W from (2.14) is defined as

$$R(\boldsymbol{u}_h, \dot{\boldsymbol{u}}_h, \ddot{\boldsymbol{u}}_h; \delta\boldsymbol{v}) = \int_\Omega \rho_0 \ddot{\boldsymbol{u}}_h : \delta\boldsymbol{v} + \partial_{\boldsymbol{u}} W(\boldsymbol{u}_h; \delta\boldsymbol{v}) - \boldsymbol{f} : \delta\boldsymbol{v} \, \mathrm{d}\boldsymbol{x} \quad \text{for all } \delta\boldsymbol{v} \in [V_h]^d.$$

In each time step of the generalized-α method, we solve the following nonlinear problem: Find $(\boldsymbol{u}_h^{(n+1)}, \dot{\boldsymbol{u}}_h^{(n+1)}, \ddot{\boldsymbol{u}}_h^{(n+1)})$ satisfying

$$\begin{aligned}
R(\boldsymbol{u}_h^{(n+\alpha_f)}, \dot{\boldsymbol{u}}_h^{(n+\alpha_f)}, \ddot{\boldsymbol{u}}_h^{(n+\alpha_m)}; \delta\boldsymbol{v}_h) &= 0 \quad \text{for all } \delta\boldsymbol{v}_h \in [V_h]^d \\
\boldsymbol{u}_h^{(n+\alpha_f)} &= \boldsymbol{u}_h^{(n)} + \alpha_f(\boldsymbol{u}_h^{(n+1)} - \boldsymbol{u}_h^{(n)}), \\
\dot{\boldsymbol{u}}_h^{(n+\alpha_f)} &= \dot{\boldsymbol{u}}_h^{(n)} + \alpha_f(\dot{\boldsymbol{u}}_h^{(n+1)} - \dot{\boldsymbol{u}}_h^{(n)}), \\
\ddot{\boldsymbol{u}}_h^{(n+\alpha_m)} &= \ddot{\boldsymbol{u}}_h^{(n)} + \alpha_m(\ddot{\boldsymbol{u}}_h^{(n+1)} - \ddot{\boldsymbol{u}}_h^{(n)}), \\
\dot{\boldsymbol{u}}_h^{(n+1)} &= \dot{\boldsymbol{u}}_h^{(n)} + \Delta t((1-\gamma)\ddot{\boldsymbol{u}}_h^{(n)} + \gamma\ddot{\boldsymbol{u}}_h^{(n+1)}), \\
\boldsymbol{u}_h^{(n+1)} &= \boldsymbol{u}_h^{(n)} + \Delta t \dot{\boldsymbol{u}}_h^{(n)} \\
&\quad + \frac{(\Delta t)^2}{2}((1-2\beta)\ddot{\boldsymbol{u}}_h^{(n)} + 2\beta\ddot{\boldsymbol{u}}_h^{(n+1)}).
\end{aligned}$$

In order to solve this system of nonlinear equations, we employ the predictor-multicorrector Algorithm 4 [34, Section 7.3]. In the predictor step, we employ a constant velocity predictor and in each corrector step, a nonlinear problem is solved using Newton–Raphson's method described in Section 3.3. The derivative with respect to $\ddot{\boldsymbol{u}}$ of the residual function may be computed using the chain rule, yielding

$$\partial_{\ddot{\boldsymbol{u}}} R(\boldsymbol{u}_h, \dot{\boldsymbol{u}}_h, \ddot{\boldsymbol{u}}_h; \delta\boldsymbol{v}, \delta\ddot{\boldsymbol{u}}) = \int_\Omega \rho_0 \alpha_m \delta\ddot{\boldsymbol{u}}_h : \delta\boldsymbol{v} + \alpha_f \beta (\Delta t)^2 \partial^2_{\boldsymbol{u}} W(\boldsymbol{u}_h; \delta\boldsymbol{v}, \delta\ddot{\boldsymbol{u}}) \, \mathrm{d}\boldsymbol{x},$$

where the second derivative of the energy density functional for a neo-Hookean medium is given by

$$\begin{aligned} \partial^2_{\boldsymbol{u}} W(\boldsymbol{u}; \delta\boldsymbol{v}, \delta\boldsymbol{u}) = \Big[& \left(\mu - \lambda \ln\left(\det\left(\boldsymbol{F}\left(\boldsymbol{u}\right)\right)\right)\right) \boldsymbol{F}(\boldsymbol{u})^{-\mathsf{T}} \nabla\delta\boldsymbol{u} \boldsymbol{F}(\boldsymbol{u})^{-\mathsf{T}} \\ & + \mu \nabla\delta\boldsymbol{u} + \lambda \operatorname{tr}\left(\boldsymbol{F}\left(\boldsymbol{u}\right)^{-1} \nabla\delta\boldsymbol{u}\right) \boldsymbol{F}\left(\boldsymbol{u}\right)^{-\mathsf{T}} \Big] : \nabla\delta\boldsymbol{v}. \end{aligned}$$

In line 10 of Algorithm 4, the tangent matrix needs to be reassembled for each correction step within each time step. In order to accelerate the assembly, we approximate it by using the surrogate method which ultimately results in a quasi-Newton–Raphson scheme as described in Section 3.3. By replacing the standard tangent matrix by a surrogate matrix in core article III (Appendix A.3), we could observe speed-ups of about 142% without any significant loss in accuracy when solving a nonlinear wave propagation problem within a medium modeled by a compressible neo-Hookean model.

Algorithm 4 Generalized-α predictor-multicorrector step from time t_n to t_{n+1}

$k \leftarrow 0$ and let *tol* be a fixed relative tolerance

$\dot{\boldsymbol{u}}_h^{(n+1;k)} = \dot{\boldsymbol{u}}_h^{(n)}$ ▷ Predictor phase

$\ddot{\boldsymbol{u}}_h^{(n+1;k)} \leftarrow \frac{\gamma-1}{\gamma}\dot{\boldsymbol{u}}_h^{(n+1;k)}$

$\boldsymbol{u}_h^{(n+1;k)} \leftarrow \boldsymbol{u}_h^{(n)} + \Delta t\dot{\boldsymbol{u}}_h^{(n)} + \frac{(\Delta t)^2}{2}((1-2\beta)\ddot{\boldsymbol{u}}_h^{(n)} + 2\beta\ddot{\boldsymbol{u}}_h^{(n+1;k)})$

while true **do** ▷ Multicorrector phase

  $\boldsymbol{u}_h^{(n+\alpha_f;k)} \leftarrow \boldsymbol{u}_h^{(n)} + \alpha_f(\boldsymbol{u}_h^{(n+1;k)} - \boldsymbol{u}_h^{(n)})$

  $\dot{\boldsymbol{u}}_h^{(n+\alpha_f;k)} \leftarrow \dot{\boldsymbol{u}}_h^{(n)} + \alpha_f(\dot{\boldsymbol{u}}_h^{(n+1;k)} - \dot{\boldsymbol{u}}_h^{(n)})$

  $\ddot{\boldsymbol{u}}_h^{(n+\alpha_m;k)} \leftarrow \ddot{\boldsymbol{u}}_h^{(n)} + \alpha_m(\ddot{\boldsymbol{u}}_h^{(n+1;k)} - \ddot{\boldsymbol{u}}_h^{(n)})$

  Compute the residual vector

$$[\mathsf{r}^{(k)}]_{d(i-1)+\ell} = R(\boldsymbol{u}_h^{(n+\alpha_f;k)}, \dot{\boldsymbol{u}}_h^{(n+\alpha_f;k)}, \ddot{\boldsymbol{u}}_h^{(n+\alpha_m;k)}; \phi_i \boldsymbol{e}_\ell)$$

  for $1 \le i \le N, 1 \le \ell \le d$

  Let A be the tangent matrix

$$[\mathsf{A}]_{d(i-1)+\ell,d(j-1)+m} = \partial_{\ddot{\boldsymbol{u}}} R(\boldsymbol{u}_h^{(n+\alpha_f;k)}, \dot{\boldsymbol{u}}_h^{(n+\alpha_f;k)}, \ddot{\boldsymbol{u}}_h^{(n+\alpha_m;k)}; \phi_i \boldsymbol{e}_\ell, \phi_j \boldsymbol{e}_m)$$

  for $1 \le i, j \le N, 1 \le \ell, m \le d$

  Solve the linear system $\mathsf{A}\delta\ddot{\mathbf{u}} = -\mathsf{r}^{(k)}$ for $\delta\ddot{\mathbf{u}}$

  $\ddot{\boldsymbol{u}}_h^{(n+1;k+1)} \leftarrow \ddot{\boldsymbol{u}}_h^{(n+1;k)} + \delta\ddot{\boldsymbol{u}}_h$

  $\dot{\boldsymbol{u}}_h^{(n+1;k+1)} \leftarrow \dot{\boldsymbol{u}}_h^{(n+1;k)} + \gamma\Delta t\delta\ddot{\boldsymbol{u}}_h$

  $\boldsymbol{u}_h^{(n+1;k+1)} \leftarrow \boldsymbol{u}_h^{(n+1;k)} + \beta(\Delta t)^2\delta\ddot{\boldsymbol{u}}_h$

  if $\|\mathsf{r}^{(k)}\|_2 < tol \cdot \|\mathsf{r}^{(0)}\|_2$ **then**

    return $(\boldsymbol{u}_h^{(n+1;k+1)}, \dot{\boldsymbol{u}}_h^{(n+1;k+1)}, \ddot{\boldsymbol{u}}_h^{(n+1;k+1)})$

  end if

  $k \leftarrow k+1$

end while

4. Performance modeling

In this section, we provide a brief introduction to performance modeling and illustrate its main concepts by considering a specific model, called the roofline model. In general, performance models can explain and predict the performance of a given code when being executed on a computer. Moreover, these models can help in the decision making progress which code optimizations are useful and which are not.

For instance, assessing only the number of FLOPs an algorithm is performing and the memory it requires, is a too simplistic model for estimating the actual run-time of the code. While this approach may give some initial understanding of the computational complexity, the actual performance of a code depends heavily on the balance between FLOPs and the required memory traffic. In fact, data access is the most import performance-limiting factor in HPC, especially in loop-based scientific applications where a lot of data is being moved in and out of the CPU [57]. In such applications, the available compute resources are often underutilized and the performance is limited by the slow data paths. On a single compute node, the slowest data path is usually the path from main memory to the CPU if all the data fits in memory. Otherwise, if data needs to be read from hard disks, the slowest path is usually the path from the disk to the memory.

A famous representative for a loop-based scientific code which is limited by data access, is the MVP of a large sparse matrix with a vector, where the matrix is stored in a compressed format [57, p. 79]. These codes are predominant in iterative solvers for linear systems originating from discretized PDEs. Using matrix-free methods reduces the amount of data-traffic by avoiding loading the indices and values of a stored sparse matrix. Instead, they only need to access the source, destination, and PDE coefficient vectors as well as some usually small helper data which often fit into the CPU caches. The constant reference and correctional stencil components from article IV (Appendix B.1) are a particular example of small data fitting in the caches. The MVP is then computed by forming parts of the global matrix on-the-fly using the PDE coefficient and helper data. By doing this, the processors are performing more work per loaded byte from memory which allows for a better utilization of the compute resources. In many cases, the performance of a code may be increased by doing more FLOPs per byte of data which needs to be loaded from memory.

In order to formalize these considerations, we make some assumptions on the computers and the loop code. We assume that a single compute node with possibly multiple sockets is used where each socket is equipped with a CPU having possibly multiple physical cores. Moreover, we presume the following assumptions stipulated in [57, p. 66]:

- The loop code uses all available instructions of the CPU in an optimal way such that all the execution units are utilized.
- All floating point arithmetics operations are performed with double precision.
- All data transfer overlaps perfectly with the arithmetic operations.
- The performance of the loop code is determined by the slowest data path.

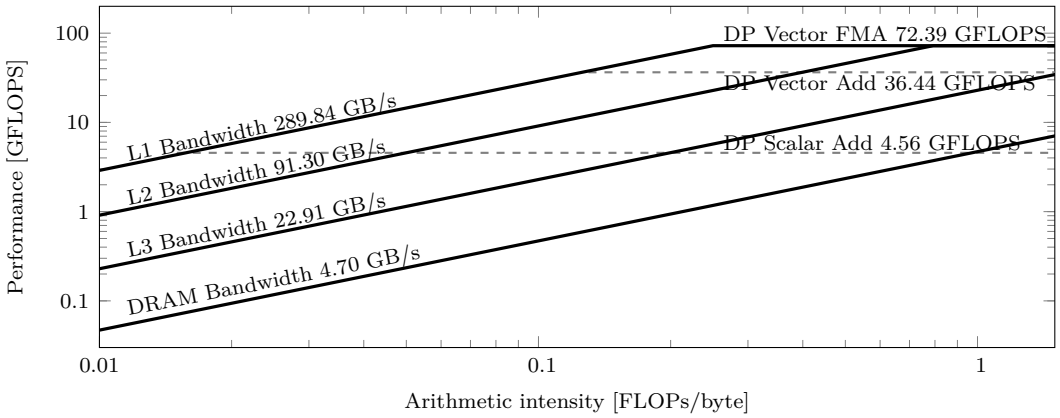


Figure 1: Roofline plot obtained on single compute node of SuperMUC-NG.

- Memory latency effects can be neglected.
- The memory bandwidth can be saturated. This usually requires the simultaneous utilization of several cores in multicore CPUs.

The absolute peak performance of a compute node is denoted by $P_{\max}$ and is measured in FLOPS. Depending on the instructions used in a loop code, the actual peak performance may vary. To illustrate this, we present in Figure 1 three performance levels which were measured on a single compute node of the state of the art supercomputer SuperMUC-NG. Each compute node is equipped with two Intel Xeon Platinum 8174 processors with a nominal clock rate of 3.1 GHz. Each processor has 24 physical cores which results in 48 cores per compute node. If a loop code only utilizes double-precision scalar additions, the maximum achievable performance is 4.56 GFLOPS. If the vector addition instructions are used instead, the maximum performance is about 36.44 GFLOPS, a factor of eight larger than the scalar performance. This is due to the vector instructions included in the modern CPUs which may perform up to eight additions in a single instruction by using AVX-512 instructions. If multiplications are followed directly by additions, special vector fused multiply-add (FMA) instructions may be used. These instructions allow performing 16 FLOPs per instruction which results in a peak performance of about 72.39 GFLOPS. In practice, loop codes limited by data access do not reach these absolute performances since the slow data paths must be taken into account.

This effect can be explained by the following reasons. The *code balance* [57, p. 66]

$$B_c = \frac{\text{data traffic [bytes]}}{\text{floating point ops [FLOPs]}}$$

of a loop code describes how many bytes of data are accessed per FLOP. The reciprocal of B_c is also often denoted by the *arithmetic intensity* $I = 1/B_c$. The maximum bandwidth of the slowest data path is denoted by $b_{\max}$ and is measured in bytes per second ($\mathrm{B\,s^{-1}}$). Similar to the different peak performance levels $P_{\max}$, there also exist different maximum bandwidth levels for the L1, L2, and L3 CPU

caches, as well as the dynamic random access memory (DRAM). Depending on the size of the data and the code balance of the loop code, one of these bandwidths may be limiting the performance. Each physical core on a SuperMUC-NG compute node has a dedicated L1 data cache of size 32 kB and a dedicated L2 cache of size 1024 kB. Each of the two processors on the compute node has a L3 cache of size 33 MB, shared across all its cores. In Figure 1, we present four bandwidth limits measured on a single compute node. The DRAM bandwidth is the slowest data path with a bandwidth of $4.70\,\mathrm{GB\,s^{-1}}$. In contrast, the CPU cache bandwidths are much larger than the DRAM bandwidth and they increase the smaller the caches become.

With the assumptions from above, the actual achievable performance P of a loop code may be estimated by

$$P = \min\left\{P_{\max}, \frac{b_{\max}}{B_c}\right\} = \min\left\{P_{\max}, I \cdot b_{\max}\right\}.$$

This simple performance model is known as the *balance* or *roofline* model [57, 106]. Plotting the performance P for the different levels of $P_{\max}$ and $b_{\max}$ over the arithmetic intensity I results in the ceilings illustrated in Figure 1. This model can be used to judge a code's performance by measuring it experimentally and placing the result in the roofline plot. Since this model takes data movement into account, it gives more insight into the performance than just the percentage of the maximum peak performance.

In article IV (Appendix B.1), we perform a roofline analysis on a compute node of SuperMUC-NG for MVPs used for residual computations in a discretization of linear elastostatics. There, we compare the performance of using stored matrices, on-the-fly integration of the bilinear forms, and the presented stencil scaling method for vector-valued PDEs. We show that matrix-free methods are beneficial regarding both, the memory consumption and the memory traffic, even in the case of low-order finite element discretizations if combined with stencil scaling techniques.

Nonetheless, the roofline model is a very simple performance model with some limitations. For instance, only the bandwidth of the slowest data path is taken into account and the remaining memory hierarchies are neglected. The roofline model also cannot explain saturation effects in multicore CPUs. Moreover, in practice, the data transfers and arithmetic operations do not overlap perfectly. A possible alternative model taking these facts into account, is the execution cache memory (ECM) model [94]. In this model, the overall runtime is decomposed into various contributions that are combined according to machine models. Since it is tedious to set up this model manually, tools like *kerncraft* [58] provide a framework to help collecting data required for the ECM model.

Acronyms

CAD computer-aided design

CCS compressed column storage

CG conjugate gradient

COO coordinate list

CPU central processing unit

CRS compressed row storage

DRAM dynamic random access memory

ECM execution cache memory

GPU graphics processing unit

FEM finite element method

FLOP floating point operation

FLOPS floating point operations per second

FMA fused multiply-add

FPGA field programmable gate array

HHG hierarchical hybrid grid

HPC high performance computing

IGA isogeometric analysis

IIL integration by interpolation and lookup

MINRES minimal residual

MUMPS multifrontal massively parallel sparse direct solver

MVP matrix vector product

NURBS non-uniform rational B-splines

PDE partial differential equation

PML perfectly matched layers

SIMD single instruction, multiple data

Bibliography

[1] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. "Parallel multigrid smoothing: polynomial versus Gauss–Seidel". In: *Journal of Computational Physics* 188.2 (July 2003), pp. 593–610. DOI: 10.1016/s0021-9991(03)00194-3.

[2] R. A. Adams and J. J. F. Fournier. *Sobolev Spaces*. Elsevier LTD, Oxford, June 1, 2003. ISBN: 0-12-044143-8.

[3] P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. "Performance and scalability of the block low-rank multifrontal factorization on multicore architectures". In: *ACM Transactions on Mathematical Software* 45 (1 2019), 2:1–2:26. DOI: 10.1145/3242094.

[4] P. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. "A fully asynchronous multifrontal solver using distributed dynamic scheduling". In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (2001), pp. 15–41. DOI: 10.1137/s0895479899358194.

[5] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, and G. Sangalli. "Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization". In: *Computer Methods in Applied Mechanics and Engineering* 285 (Mar. 2015), pp. 817–828. DOI: 10.1016/j.cma.2014.12.013.

[6] P. Arbenz, G. H. van Lenthe, U. Mennel, R. Müller, and M. Sala. "A scalable multi-level preconditioner for matrix-free μ-finite element analysis of human bone structures". In: *International Journal for Numerical Methods in Engineering* 73.7 (2008), pp. 927–947. DOI: 10.1002/nme.2101.

[7] Argonne National Laboratory. *U.S. Department of Energy and Intel to deliver first exascale supercomputer*. Mar. 18, 2019. URL: https://www.anl.gov/article/us-department-of-energy-and-intel-to-deliver-first-exascale-supercomputer (visited on 03/31/2020).

[8] F. Auricchio, F. Calabrò, T. Hughes, A. Reali, and G. Sangalli. "A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis". In: *Computer Methods in Applied Mechanics and Engineering* 249-252 (Dec. 2012), pp. 15–27. DOI: 10.1016/j.cma.2012.04.014.

[9] R. Barrett et al. *Templates for the solution of linear systems: Building blocks for iterative methods*. Society for Industrial and Applied Mathematics, Jan. 1994. DOI: 10.1137/1.9781611971538.

[10] P. Bastian, J. Kraus, R. Scheichl, and M. Wheeler, eds. *Simulation of Flow in Porous Media*. De Gruyter, Jan. 2013. DOI: 10.1515/9783110282245.

[11] P. Bastian, E. H. Müller, S. Müthing, and M. Piatkowski. "Matrix-free multigrid block-preconditioners for higher order discontinuous Galerkin discretisations". In: *Journal of Computational Physics* 394 (Oct. 2019), pp. 417–439. DOI: 10.1016/j.jcp.2019.06.001.

[12] S. Bauer, M. Huber, S. Ghelichkhan, M. Mohr, U. Rüde, and B. Wohlmuth. "Large-scale simulation of mantle convection based on a new matrix-free approach". In: *Journal of Computational Science* 31 (Feb. 2019), pp. 60–76. DOI: 10.1016/j.jocs.2018.12.006.

[13] S. Bauer, M. Mohr, U. Rüde, J. Weismüller, M. Wittmann, and B. Wohlmuth. "A two-scale approach for efficient on-the-fly operator assembly in massively parallel high performance multigrid codes". In: *Applied Numerical Mathematics* 122 (Dec. 2017), pp. 14–38. DOI: 10.1016/j.apnum.2017.07.006.

[14] S. Bauer, D. Drzisga, M. Mohr, U. Rüde, C. Waluga, and B. Wohlmuth. "A stencil scaling approach for accelerating matrix-free finite element implementations". In: *SIAM Journal on Scientific Computing* 40.6 (2018), pp. C748–C778. DOI: 10.1137/17m1148384.

[15] S. Bauer, M. Huber, M. Mohr, U. Rüde, and B. Wohlmuth. "A new matrix-free approach for large-scale geodynamic simulations and its performance". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2018, pp. 17–30. DOI: 10.1007/978-3-319-93701-4_2.

[16] Y. Bazilevs, L. Beirão Da Veiga, J. A. Cottrell, T. J. R. Hughes, and G. Sangalli. "Isogeometric analysis: approximation, stability and error estimates for h-refined meshes". In: *Mathematical Models and Methods in Applied Sciences* 16.07 (July 2006), pp. 1031–1090. DOI: 10.1142/s0218202506001455.

[17] J.-P. Berenger. "A perfectly matched layer for the absorption of electromagnetic waves". In: *Journal of Computational Physics* 114.2 (1994), pp. 185–200. DOI: 10.1006/jcph.1994.1159.

[18] B. Bergen, G. Wellein, F. Hülsemann, and U. Rüde. "Hierarchical hybrid grids: achieving TERAFLOP performance on large scale finite element simulations". In: *International Journal of Parallel, Emergent and Distributed Systems* 22.4 (Aug. 2007), pp. 311–329. DOI: 10.1080/17445760701442218.

[19] B. Bergen. *Hierarchical Hybrid Grids: Data Structures and Core Algorithms for Efficient Finite Element Simulations on Supercomputers*. Erlangen: SCS Publishing House, 2005.

[20] B. K. Bergen and F. Hülsemann. "Hierarchical hybrid grids: data structures and core algorithms for multigrid". In: *Numerical Linear Algebra with Applications* 11.23 (Mar. 2004), pp. 279–291. DOI: 10.1002/nla.382.

[21] J. Bielak, O. Ghattas, and E.-J. Kim. "Parallel octree-based finite element method for large-scale earthquake ground motion simulation". In: *Computer Modeling in Engineering & Sciences* 10.2 (2005), pp. 99–112. DOI: 10.3970/cmes.2005.010.099.

[22] C. de Boor. "On calculating with B-splines". In: *Journal of Approximation Theory* 6.1 (July 1972), pp. 50–62. DOI: 10.1016/0021-9045(72)90080-9.

[23] D. Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Berlin Heidelberg New York: Springer, 2007. ISBN: 978-3-540-72449-0.

[24] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*. Springer-Verlag GmbH, Jan. 1, 2008. ISBN: 0-387-75933-6. DOI: 10.1007/978-0-387-75934-0.

[25] A. Bressan and G. Sangalli. "Isogeometric discretizations of the Stokes problem: stability analysis by the macroelement technique". In: *IMA Journal of Numerical Analysis* 33.2 (2013), pp. 629–651.

[26] A. Bressan and S. Takacs. "Sum factorization techniques in isogeometric analysis". In: *Computer Methods in Applied Mechanics and Engineering* 352 (Aug. 2019), pp. 437–460. DOI: 10.1016/j.cma.2019.04.031.

[27] F. Brezzi and J. Pitkäranta. "On the stabilization of finite element approximations of the Stokes equations". In: *Efficient Solutions of Elliptic Systems*. Vieweg+Teubner Verlag, 1984, pp. 11–19. DOI: 10.1007/978-3-663-14169-3_2.

[28] J. Brown. "Efficient nonlinear solvers for nodal high-order finite elements in 3D". In: *Journal of Scientific Computing* 45.1-3 (2010), pp. 48–63. DOI: 10.1007/s10915-010-9396-8.

[29] F. Calabrò, G. Sangalli, and M. Tani. “Fast formation of isogeometric Galerkin matrices by weighted quadrature”. In: *Computer Methods in Applied Mechanics and Engineering* 316 (2017). Special Issue on Isogeometric Analysis: Progress and Challenges, pp. 606–622. ISSN: 0045-7825. DOI: 10.1016/j.cma.2016.09.013.

[30] G. F. Carey and B.-N. Jiang. “Element-by-element linear and nonlinear solution schemes”. In: *Communications in Applied Numerical Methods* 2.2 (1986), pp. 145–153. DOI: 10.1002/cnm.1630020205.

[31] A. J. Chorin. “Numerical solution of the Navier-Stokes equations”. In: *Mathematics of Computation* 22.104 (1968), pp. 745–745. DOI: 10.1090/s0025-5718-1968-0242392-2.

[32] J. Chung and G. M. Hulbert. “A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method”. In: *Journal of Applied Mechanics* 60.2 (June 1993), pp. 371–375. DOI: 10.1115/1.2900803.

[33] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. Society for Industrial and Applied Mathematics, Jan. 2002. DOI: 10.1137/1.9780898719208.

[34] J. A. Cottrell, T. J. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[35] M. G. Cox. “The numerical evaluation of B-splines”. In: *IMA Journal of Applied Mathematics* 10.2 (1972), pp. 134–149. DOI: 10.1093/imamat/10.2.134.

[36] L. B. Da Veiga, A. Buffa, J. Rivas, and G. Sangalli. “Some estimates for h–p–k-refinement in isogeometric analysis”. In: *Numerische Mathematik* 118.2 (Oct. 2010), pp. 271–305. DOI: 10.1007/s00211-010-0338-z.

[37] L. B. Da Veiga, A. Buffa, G. Sangalli, and R. Vázquez. “Mathematical analysis of variational isogeometric methods”. In: *Acta Numerica* 23 (2014), pp. 157–287. DOI: 10.1017/s096249291400004x.

[38] D. Drzisga, B. Gmeiner, U. Rüde, R. Scheichl, and B. Wohlmuth. “Scheduling massively parallel multigrid for multilevel Monte Carlo methods”. In: *SIAM Journal on Scientific Computing* 39.5 (2017), S873–S897. DOI: 10.1137/16m1083591.

[39] D. Drzisga, L. John, U. Rüde, B. Wohlmuth, and W. Zulehner. “On the analysis of block smoothers for saddle point problems”. In: *SIAM Journal on Matrix Analysis and Applications* 39.2 (2018), pp. 932–960. DOI: 10.1137/16m1106304.

[40] D. Drzisga, B. Keith, and B. Wohlmuth. “The surrogate matrix methodology: a priori error estimation”. In: *SIAM Journal on Scientific Computing* 41.6 (2019), A3806–A3838. DOI: 10.1137/18M1226580.

[41] D. Drzisga, B. Keith, and B. Wohlmuth. “The surrogate matrix methodology: A reference implementation for low-cost assembly in isogeometric analysis”. In: *MethodsX* 7 (2020), p. 100813. DOI: 10.1016/j.mex.2020.100813.

[42] D. Drzisga, B. Keith, and B. Wohlmuth. “The surrogate matrix methodology: Accelerating isogeometric analysis of waves”. In: *Computer Methods in Applied Mechanics and Engineering* 372 (Dec. 2020), p. 113322. DOI: 10.1016/j.cma.2020.113322.

[43] D. Drzisga, B. Keith, and B. Wohlmuth. “The surrogate matrix methodology: Low-cost assembly for isogeometric analysis”. In: *Computer Methods in Applied Mechanics and Engineering* 361 (2020), p. 112776. DOI: 10.1016/j.cma.2019.112776.

[44] D. Drzisga, T. Köppl, U. Pohl, R. Helmig, and B. Wohlmuth. “Numerical modeling of compensation mechanisms for peripheral arterial stenoses”. In: *Computers in Biology and Medicine* 70 (Mar. 2016), pp. 190–201. DOI: 10.1016/j.compbiomed.2016.01.015.

[45] D. Drzisga, U. Rüde, and B. Wohlmuth. “Stencil scaling for vector-valued PDEs on hybrid grids with applications to generalized Newtonian fluids”. In: *SIAM Journal on Scientific Computing* 42.6 (Dec. 1, 2020), B1429–B1461. DOI: 10.1137/19m1267891.

[46] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press, USA, 2014. DOI: 10.1093/acprof:oso/9780199678792.001.0001.

[47] C. Engwer, R. D. Falgout, and U. M. Yang. “Stencil computations for PDE-based applications with examples from DUNE and hypre”. In: *Concurrency and Computation: Practice and Experience* 29.17 (Feb. 2017), e4097. DOI: 10.1002/cpe.4097.

[48] A. Ern and J.-L. Guermond. *Theory and Practice of Finite Elements*. New York, NY: Springer New York, 2004. ISBN: 978-1-4757-4355-5. DOI: 10.1007/978-1-4757-4355-5.

[49] S. Esterhazy and J. Melenk. “An analysis of discretizations of the Helmholtz equation in L^2 and in negative norms”. In: *Computers & Mathematics with Applications* 67.4 (Mar. 2014), pp. 830–853. DOI: 10.1016/j.camwa.2013.10.005.

[50] F. Fahrendorf, L. D. Lorenzis, and H. Gomez. “Reduced integration at superconvergent points in isogeometric analysis”. In: *Computer Methods in Applied Mechanics and Engineering* 328 (2018), pp. 390–410. ISSN: 0045-7825. DOI: 10.1016/j.cma.2017.08.028.

[51] C. de Falco, A. Reali, and R. Vázquez. “GeoPDEs: a research tool for Isogeometric Analysis of PDEs”. In: *Advances in Engineering Software* 42.12 (2011), pp. 1020–1034. DOI: 10.1016/j.advengsoft.2011.06.010.

[52] N. Fehn, P. Munch, W. A. Wall, and M. Kronbichler. “Hybrid multigrid methods for high-order discontinuous Galerkin discretizations”. In: *Journal of Computational Physics* 415 (Aug. 15, 2020), p. 109538. DOI: 10.1016/j.jcp.2020.109538.

[53] C. Flaig and P. Arbenz. “A highly scalable matrix-free multigrid solver for μFE analysis based on a pointer-less octree”. In: *Large-Scale Scientific Computing: 8th International Conference, LSSC 2011, Sozopol, Bulgaria, June 6-10, 2011, Revised Selected Papers*. Springer Berlin Heidelberg, 2012, pp. 498–506. ISBN: 978-3-642-29843-1. DOI: 10.1007/978-3-642-29843-1_56.

[54] B. Gmeiner, T. Gradl, H. Köstler, and U. Rüde. “Highly parallel geometric multigrid algorithm for hierarchical hybrid grids”. In: *NIC Symposium*. Vol. 45. 2012, pp. 323–330.

[55] B. Gmeiner, M. Huber, L. John, U. Rüde, and B. Wohlmuth. “A quantitative performance study for Stokes solvers at the extreme scale”. In: *Journal of Computational Science* (2016). DOI: 10.1016/j.jocs.2016.06.006.

[56] B. Gmeiner, U. Rüde, H. Stengel, C. Waluga, and B. Wohlmuth. “Towards textbook efficiency for parallel multigrid”. In: *Numerical Mathematics: Theory, Methods and Applications* 8.1 (Feb. 2015), pp. 22–46. DOI: 10.4208/nmtma.2015.w10si.

[57] G. Hager and G. Wellein. *Introduction to high performance computing for scientists and engineers*. CRC Press, 2010. DOI: 10.1201/ebk1439811924.

[58] J. Hammer, J. Eitzinger, G. Hager, and G. Wellein. “Kerncraft: A tool for analytic performance modeling of loop kernels”. In: *Tools for High Performance Computing 2016*. Springer International Publishing, 2017, pp. 1–22. DOI: 10.1007/978-3-319-56702-0_1.

[59] V. E. Henson and U. M. Yang. "BoomerAMG: A parallel algebraic multigrid solver and preconditioner". In: *Applied Numerical Mathematics* 41.1 (Apr. 2002), pp. 155–177. DOI: 10.1016/s0168-9274(01)00115-5.

[60] R. R. Hiemstra, F. Calabrò, D. Schillinger, and T. J. Hughes. "Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis". In: *Computer Methods in Applied Mechanics and Engineering* 316 (Apr. 2017), pp. 966–1004. DOI: 10.1016/j.cma.2016.10.049.

[61] R. R. Hiemstra, G. Sangalli, M. Tani, F. Calabrò, and T. J. Hughes. "Fast formation and assembly of finite element matrices with application to isogeometric linear elasticity". In: *Computer Methods in Applied Mechanics and Engineering* 355 (Oct. 2019), pp. 234–260. DOI: 10.1016/j.cma.2019.06.020.

[62] C. Hofreither. "A black-box low-rank approximation algorithm for fast matrix assembly in isogeometric analysis". In: *Computer Methods in Applied Mechanics and Engineering* 333 (May 2018), pp. 311–330. DOI: 10.1016/j.cma.2018.01.014.

[63] K. Höllig. *Finite Element Methods with B-Splines*. Society for Industrial and Applied Mathematics, Jan. 2003. DOI: 10.1137/1.9780898717532.

[64] T. J. Hughes, L. P. Franca, and M. Balestra. "A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška-Brezzi condition: a stable Petrov–Galerkin formulation of the Stokes problem accommodating equal-order interpolations". In: *Computer Methods in Applied Mechanics and Engineering* 59.1 (Nov. 1986), pp. 85–99. DOI: 10.1016/0045-7825(86)90025-3.

[65] T. J. Hughes, J. A. Cottrell, and Y. Bazilevs. "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement". In: *Computer Methods in Applied Mechanics and Engineering* 194.39-41 (2005), pp. 4135–4195.

[66] T. J. Hughes, A. Reali, and G. Sangalli. "Efficient quadrature for NURBS-based isogeometric analysis". In: *Computer Methods in Applied Mechanics and Engineering* 199.5-8 (2010), pp. 301–313. DOI: 10.1016/j.cma.2008.12.004.

[67] T. J. Hughes and G. Sangalli. "Mathematics of isogeometric analysis: a conspectus". In: *Encyclopedia of Computational Mechanics Second Edition* (2018), pp. 1–40.

[68] K. E. Jansen, C. H. Whiting, and G. M. Hulbert. "A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method". In: *Computer Methods in Applied Mechanics and Engineering* 190.3-4 (Oct. 2000), pp. 305–319. DOI: 10.1016/s0045-7825(00)00203-6.

[69] N. Kohl, D. Thönnes, D. Drzisga, D. Bartuschat, and U. Rüde. "The HyTeG finite-element software framework for scalable multigrid solvers". In: *International Journal of Parallel, Emergent and Distributed Systems* 34.5 (Aug. 2018), pp. 477–496. DOI: 10.1080/17445760.2018.1506453.

[70] M. Kronbichler and K. Kormann. "A generic interface for parallel cell-based finite element operator application". In: *Computers and Fluids* 63 (2012), pp. 135–147. DOI: 10.1016/j.compfluid.2012.04.012.

[71] M. Kronbichler and K. Ljungkvist. "Multigrid for matrix-free high-order finite element computations on graphics processors". In: *ACM Transactions on Parallel Computing* 6.1 (June 2019), pp. 1–32. DOI: 10.1145/3322813.

[72] M. Kronbichler and W. A. Wall. "A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers". In: *SIAM Journal on Scientific Computing* 40.5 (Jan. 2018), A3423–A3448. DOI: 10.1137/16m110455x.

[73] A. Kuijper. "P-Laplacian driven image processing". In: *2007 IEEE International Conference on Image Processing*. IEEE, 2007. DOI: 10.1109/icip.2007.4379814.

[74] J. Lions. *Quelques méthodes de résolution des problèmes aux limites non linéaires*. Etudes mathématiques. Dunod, 1969.

[75] K. Ljungkvist. "Matrix-free finite-element computations on graphics processors with adaptively refined unstructured meshes". In: *Proceedings of the 25th High Performance Computing Symposium*. HPC '17. Society for Computer Simulation International, 2017, 1:1–1:12.

[76] K. Ljungkvist and M. Kronbichler. *Multigrid for matrix-free finite element computations on graphics processors*. Tech. rep. 2017-006. Department of Information Technology, Uppsala University, 2017.

[77] J. Loffeld and J. Hittinger. "On the arithmetic intensity of high-order finite-volume discretizations for hyperbolic systems of conservation laws". In: *The International Journal of High Performance Computing Applications* (2017). DOI: 10.1177/1094342017691876.

[78] A. Mantzaflaris and B. Jüttler. "Exploring matrix generation strategies in isogeometric analysis". In: *Mathematical Methods for Curves and Surfaces*. Springer Berlin Heidelberg, 2014, pp. 364–382. DOI: 10.1007/978-3-642-54382-1_21.

[79] A. Mantzaflaris and B. Jüttler. "Integration by interpolation and look-up for Galerkin-based isogeometric analysis". In: *Computer Methods in Applied Mechanics and Engineering* 284 (Feb. 2015), pp. 373–400. DOI: 10.1016/j.cma.2014.09.014.

[80] A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, and U. Langer. "Low rank tensor methods in Galerkin-based isogeometric analysis". In: *Computer Methods in Applied Mechanics and Engineering* 316 (Apr. 2017), pp. 1062–1085. DOI: 10.1016/j.cma.2016.11.013.

[81] P. J. Matuszyk and L. F. Demkowicz. "Parametric finite elements, exact sequences and perfectly matched layers". In: *Computational Mechanics* 51.1 (Mar. 2012), pp. 35–45. DOI: 10.1007/s00466-012-0702-1.

[82] D. A. May, J. Brown, and L. L. Pourhiet. "A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow". In: *Computer Methods in Applied Mechanics and Engineering* 290 (2015), pp. 496–523. DOI: 10.1016/j.cma.2015.03.014.

[83] D. A. May, P. Sanan, K. Rupp, M. G. Knepley, and B. F. Smith. "Extreme-scale multigrid components within PETSc". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. PASC '16. Lausanne, Switzerland: ACM, 2016, 5:1–5:12. ISBN: 978-1-4503-4126-4. DOI: 10.1145/2929908.2929913.

[84] F. Moser, L. J. Jacobs, and J. Qu. "Modeling elastic wave propagation in waveguides with the finite element method". In: *NDT & E International* 32.4 (June 1999), pp. 225–234. DOI: 10.1016/s0963-8695(98)00045-0.

[85] R. W. Ogden. *Non-Linear Elastic Deformations*. Guilford Publications, Apr. 26, 2013. 544 pp.

[86] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Society for Industrial and Applied Mathematics, Jan. 2000. DOI: 10.1137/1.9780898719468.

[87] M. Pan, B. Jüttler, and A. Giust. "Fast formation of isogeometric Galerkin matrices via integration by interpolation and look-up". In: *Computer Methods in Applied Mechanics and Engineering* 366 (July 2020), p. 113005. DOI: 10.1016/j.cma.2020.113005.

[88] W. Pazner and P.-O. Persson. "Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods". In: *Journal of Computational Physics* 354 (Feb. 2018), pp. 344–369. DOI: 10.1016/j.jcp.2017.10.030.

[89] B. van Rietbergen, H. Weinans, R. Huiskes, and B. Polman. "Computational strategies for iterative solutions of large FEM applications employing voxel data". In: *International Journal for Numerical Methods in Engineering* 39.16 (1996), pp. 2743–2767. DOI: 10.1002/(SICI)1097-0207(19960830)39:16<2743::AID-NME974>3.0.CO;2-A.

[90] B. Rivière. *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations*. Society for Industrial and Applied Mathematics, Jan. 2008. DOI: 10.1137/1.9780898717440.

[91] D. Rypl and B. Patzák. "Study of computational efficiency of numerical quadrature schemes in the isogeometric analysis". In: *Engineering Mechanics* 304 (2012).

[92] G. Sangalli and M. Tani. "Matrix-free weighted quadrature for a computationally efficient isogeometric k-method". In: *Computer Methods in Applied Mechanics and Engineering* 338 (Aug. 2018), pp. 117–133. DOI: 10.1016/j.cma.2018.04.029.

[93] D. Schillinger, S. J. Hossain, and T. J. Hughes. "Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis". In: *Computer Methods in Applied Mechanics and Engineering* 277 (Aug. 2014), pp. 1–45. DOI: 10.1016/j.cma.2014.04.008.

[94] H. Stengel, J. Treibig, G. Hager, and G. Wellein. "Quantifying performance bottlenecks of stencil computations using the execution-cache-memory model". In: *Proceedings of the 29th ACM on International Conference on Supercomputing - ICS '15*. ACM Press, 2015. DOI: 10.1145/2751205.2751240.

[95] G. Strang. "Piecewise polynomials and the finite element method". In: *Bulletin of the American Mathematical Society* 79.6 (1973), pp. 1128–1137. DOI: 10.1090/s0002-9904-1973-13351-8.

[96] G. Strang. "Variational crimes in the finite element method". In: *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*. Elsevier, 1972, pp. 689–710. DOI: 10.1016/b978-0-12-068650-6.50030-7.

[97] G. Strang and G. J. Fix. *An analysis of the finite element method*. Prentice-Hall series in automatic computation. Englewood Cliffs, NJ: Prentice-Hall, 1973. DOI: 10.2307/2005716.

[98] A. Tagliabue, L. Dedè, and A. Quarteroni. "Isogeometric analysis and error estimates for high order partial differential equations in fluid dynamics". In: *Computers & Fluids* 102 (Oct. 2014), pp. 277–303. DOI: 10.1016/j.compfluid.2014.07.002.

[99] R. Tielen, M. Möller, D. Göddeke, and C. Vuik. *A p-multigrid method enhanced with an ILUT smoother and its comparison to h-multigrid methods within isogeometric analysis*. Jan. 7, 2019. arXiv: 1901.01685v3 [math.NA].

[100] U.S. Energy Information Administration. *2018 Average Monthly Bill- Residential*. Oct. 1, 2019. URL: https://www.eia.gov/electricity/sales_revenue_price/pdf/table5_a.pdf (visited on 04/04/2020).

[101] U.S. Energy Information Administration. *How much electricity does an American home use?* Oct. 2, 2019. URL: https://www.eia.gov/tools/faqs/faq.php?id=97&t=3 (visited on 04/04/2020).

[102] R. Vázquez. "A new design for the implementation of isogeometric analysis in Octave and Matlab: GeoPDEs 3.0". In: *Computers & Mathematics with Applications* 72.3 (2016), pp. 523–554. DOI: 10.1016/j.camwa.2016.05.010.

[103] T. Vijayaraghavan et al. “Design and analysis of an APU for exascale computing”. In: *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE. 2017, pp. 85–96. DOI: 10.1109/hpca.2017.42.

[104] M. M. Waldrop. “The chips are down for Moore’s law”. In: *Nature* 530.7589 (Feb. 2016), pp. 144–147. DOI: 10.1038/530144a.

[105] D. Werner. *Funktionalanalysis*. Springer Berlin Heidelberg, 2011. DOI: 10.1007/978-3-642-21017-4.

[106] S. Williams, A. Waterman, and D. Patterson. “Roofline: An insightful visual performance model for multicore architectures”. In: *Communications of the ACM* 52.4 (Apr. 2009), pp. 65–76. DOI: 10.1145/1498765.1498785.

A. Core articles

A.1. The surrogate matrix methodology: a priori error estimation

The surrogate matrix methodology: a priori error estimation

Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth

In this article, we reconsider the classical lowest-order Bubnov–Galerkin finite element method and analyze a modification of it which is especially advantageous for stencil-based matrix-free computations. This methodology, which we call *the surrogate matrix methodology*, is based on approximating the matrix entries emerging in a standard finite element discretization by piece-wise smooth functions. This modification was originally introduced by Bauer et al. in [13] where it was applied to Poisson's problem. Afterwards, in two follow-up articles, Stokes flow was considered in [12, 15]. Each of these articles focuses on the massively parallel high performance computing aspects of the respective methods and they provide numerical indication for the convergence rates. In our work, however, we present the first mathematically rigorous framework and analysis of this methodology by performing an a priori error analysis for the variable coefficient Poisson equation. In this approach, a globally unstructured macro-mesh is used to triangulate the model geometry. Each of the macro-elements in the macro-mesh are uniformly refined for a large number of times, resulting in a fine-scale locally-structured mesh. Afterwards, for each macro-element, a local approximation of the fine-scale global matrix is constructed, resulting in a fine-scale surrogate matrix. We show that the solutions obtained by using these surrogate matrices preserve the asymptotical approximation properties of the standard fine-scale discrete solution. In several numerical experiments, we demonstrate the efficiency of this new method in a matrix-free framework using geometric multigrid solvers.

In Section 3, we introduce the concept of stencil functions and the construction of surrogate matrices. Section 4 provides a short overview of the considered examples, namely, the variable coefficient Poisson equation, the linear elastostatics model, and the $\mathfrak{p}$-Laplacian diffusion problem. In Section 5, we discuss the incorporation of non-homogeneous boundary conditions and the zero row sum property. In Section 6, sufficient conditions for the discrete stability of the surrogate bilinear forms are briefly discussed and some preliminary results are presented. There, we perform a rigorous a priori error analysis of our approach applied to the variable coefficient Poisson equation and provide estimates for the errors in the $H^1(\Omega)$- and $L^2(\Omega)$-norms. A brief description of our matrix-free implementation using the finite element software framework HyTeG [69] is provided in Section 8. Then, in Section 9, we present several numerical experiments. There, we thoroughly verify the proven a priori convergence rates for the variable coefficient Poisson equation. We also include proof-of-concept demonstrations from numerical experiments with the linear elastostatics and $\mathfrak{p}$-Laplacian diffusion problems which are complemented by performance measurements.

I was significantly involved in finding the ideas and primarily responsible for setting up the mathematical framework and carrying out the scientific work presented in this article. Furthermore, I was in charge of writing the article while the co-authors contributed by making corrective changes.

Permission to include:

Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth
The surrogate matrix methodology: a priori error estimation
SIAM Journal on Scientific Computing 41.6 (2019): A3806–A3838
(see also article [40] in the bibliography)

On the following page, a copy of the first page of the consent to publish agreement by SIAM may be found. This page includes the author's rights. A digital version of the consent to publish form may be found at

https://www.siam.org/publications/journals/about-siam-journals/information-for-authors

(Accessed on 22 March 2020)

Society for Industrial and Applied Mathematics (SIAM)

Consent to Publish

SIAM ("Publisher") requires Authors of articles in SIAM publications to provide a formal written Consent to Publish. The Author must sign the agreement except, in the case of "work-for-hire", when the Author's employer may sign as the party that has the right to grant rights to the Publisher. If there are multiple Authors of the material governed by this document, the term "Author" as used here refers to each and all of them, jointly and severally. [1]

Title of Contribution ("Work"): The surrogate matrix methodology: a priori error estimation
Authors: Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth
Name of Journal: SIAM Journal on Scientific Computing
Manuscript Number: M122658

1. Author's Warranty

By signing this Consent, the Author warrants all of the following: The Work has not been published before in any form except as a preprint, unless explicitly noted as a footnote to the title. The Work is not being concurrently submitted to and is not under consideration by another publisher. The names listed above as authors appear in the manuscript itself, no author entitled to credit has been omitted, and there are no unnamed authors. The Author has the right to make the grants made to the Publisher complete and unencumbered. The Author also warrants that the Work does not libel anyone, violate anyone's privacy or publicity rights, infringe anyone's copyright, trademark, or trade secrets, or otherwise violate anyone's statutory or common law rights.

2. Author's Rights

A1. The Author may reproduce and distribute the Work (including derivative works) in connection with the Author's teaching, technical collaborations, conference presentations, lectures, or other scholarly works and professional activities as well as to the extent the fair use provisions of the U.S. Copyright Act permit. If the copyright is granted to the Publisher, then the proper notice of the Publisher's copyright should be provided.
A2. The Author may post the final draft of the Work, as it exists immediately prior to editing and production by the Publisher, on noncommercial pre-print servers such as arXiv.org.
A3. The Author may post the final published version of the Work on the Author's personal web site and on the web server of the Author's institution, provided that proper notice of the Publisher's copyright is included and that no separate or additional fees are collected for access to or distribution of the work.

3. Publisher's Rights

Even if the Author does not transfer Copyright to the Publisher, the Author grants the Publisher the following rights in perpetuity.

P1. The Publisher has unlimited rights throughout the world to publish and distribute the final version of the Work in any form and in all media now known or hereafter discovered.
P2. The Publisher has unlimited rights throughout the world to translate the final version of the Work and exercise all rights in all media in the resulting translations.

Notice of publication and copyright

First Published in “The surrogate matrix methodology: a priori error estimation” in SIAM Journal on Scientific Computing 41.6 (2019), published by the Society for Industrial and Applied Mathematics (SIAM).

DOI: https://doi.org/10.1137/18M1226580

SIAM J. SCI. COMPUT.
Vol. 41, No. 6, pp. A3806–A3838

THE SURROGATE MATRIX METHODOLOGY: A PRIORI ERROR ESTIMATION*

DANIEL DRZISGA†, BRENDAN KEITH†, AND BARBARA WOHLMUTH†

Abstract. We give the first mathematically rigorous analysis of an emerging approach to finite element analysis (see, e.g., Bauer et al. [*Appl. Numer. Math.*, 122 (2017), pp. 14–38]), which we hereby refer to as the surrogate matrix methodology. This methodology is based on the piecewise smooth approximation of the matrices involved in a standard finite element discretization. In particular, it relies on the projection of smooth so-called stencil functions onto high-order polynomial subspaces. The performance advantage of the surrogate matrix methodology is seen in constructions where each stencil function uniquely determines the values of a significant collection of matrix entries. Such constructions are shown to be widely achievable through the use of locally structured meshes. Therefore, this methodology can be applied to a wide variety of physically meaningful problems, including nonlinear problems and problems with curvilinear geometries. Rigorous a priori error analysis certifies the convergence of a novel surrogate method for the variable coefficient Poisson equation. The flexibility of the methodology is also demonstrated through the construction of novel methods for linear elasticity and nonlinear diffusion problems. In numerous numerical experiments, we demonstrate the efficacy of these new methods in a matrix-free environment with geometric multigrid solvers. In our experiments, up to a twenty-fold decrease in computation time is witnessed over the classical method with an otherwise identical implementation.

Key words. surrogate numerical methods, finite element methods, matrix-free, a priori analysis, low order, geometric multigrid

AMS subject classifications. 65D05, 65M60, 65N30, 65Y05, 65Y20

DOI. 10.1137/18M1226580

1. Introduction.

In the field of computational science, major funding initiatives in North America, Europe, and Asia have thrust high performance computing (HPC) to ascendancy. In anticipation of future exascale computers, much work in this discipline involves the deep and careful reconstruction of long-established computing practices. An important characteristic of numerical algorithms aimed at these computers is the floating-point operation (FLOP) per byte ratio. In order to achieve optimal performance and power efficiency on future machines, the time spent on FLOPs relative to memory transfer needs to be substantial.

Most traditional finite element softwares assemble global stiffness matrices by looping over elements and adding the corresponding local contributions to the global matrix. Storing the resulting sparse matrices requires significantly more memory than just storing the degrees of freedom. However, memory consumption is certainly not the only obstacle at the computational frontier. Indeed, at such scales, the memory traffic and latency involved in loading indices and entries for matrix-vector products (MVPs) also present critical challenges.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section November 14, 2018; accepted for publication (in revised form) September 17, 2019; published electronically December 3, 2019.

https://doi.org/10.1137/18M1226580

Funding: This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement 800898 and by the German Research Foundation through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) and by grant WO671/11-1.

†Lehrstuhl für Numerische Mathematik, Fakultät für Mathematik (M2), Technische Universität München, Garching bei München, 85748, Germany (drzisga@ma.tum.de, keith@ma.tum.de, wohlmuth@ma.tum.de).

Since iterative solvers only require MVPs, it is not necessary to store all of the nonzeros of the global matrix in memory. Instead, it is sufficient to compute the nonzero entries on the fly, i.e., matrix-free. Different tactics exist to implement matrix-free methods, but the predominant candidate for low-order finite elements is the element-by-element approach [3, 15, 19, 26, 40], wherein local stiffness matrices are multiplied by local vectors and later added to the global solution vector. These local stiffness matrices may either be stored in memory—which actually requires more memory than storing the global matrix—or computed on-the-fly. When using high-order finite elements, the weak forms can be integrated on-the-fly using standard or reduced quadrature formulas [17, 32, 33, 34, 36]. This is a well-suited tactic for future machines because of its large arithmetic intensity [35].

Significant performance gains are often attributed to a problem, scale, and architecture-specific balance between FLOPs and memory traffic. As a matter of course, exploiting symmetries in a problem or discretization can significantly improve the time to solution. This is often the cause of enormous speed-ups in computations on structured meshes. Likewise, high performance of matrix-free methods can be most easily achieved in homogeneous problems with simple geometries. Nevertheless, most geometries coming from significant real-world problems cannot be adequately approximated by fully structured meshes. A possible trade-off is to use locally structured meshes like hierarchical hybrid grids (HHGs) where initially unstructured coarse grids are locally refined in a uniform way. This local structure allows the application of stencil-based finite element procedures which operate similar to finite difference methods. By using these grids, efficient stencil-based methods have been successfully applied to a wide range of problems [11, 12, 13, 24, 28]. Related approaches, suitable for low-order finite element discretizations of elliptic partial differential equations (PDEs) with variable coefficients, based on scaling of reference stencils, are discussed in [7, 23].

In this paper, we revisit the classical lowest-order Bubnov–Galerkin finite element method and analyze a modification of it which is strongly amenable to stencil-based matrix-free computation. In our approach, a macromesh, which is not required to have any global structure, is used to triangulate the model geometry. This macromesh is then uniformly refined a large number of times, resulting in a fine-scale locally structured mesh. For each macroelement, a *local approximation* of the fine-scale global matrix delivers a fine-scale *surrogate matrix* which maintains the convergence properties of the fine-scale discrete solution, up to the original order of the approximation. A related investigation [10] illustrated the promise of this methodology and provided numerical evidence for the convergence rates which are proven here rigorously. This work considered only Poisson's equation. Later on, Stokes flow (with variable viscosity) was considered in two follow-up articles [8, 9]. Each of these initial studies focused on the massively parallel high performance computing aspects of their respective methods. These studies used the HHG software framework [11, 12, 13] in their experiments. Here, the finite element hybrid tetrahedral grids (HyTeG) software framework [31] is used.

In this paper, we recast the central features of the original work as a methodology complete with a mathematical framework suitable for rigorous analysis. The principal novelty is the mathematical foundation developed here, which can be used to analyze further incarnations of the methodology. In total, we consider three specific mathematical models; namely, the variable coefficient Poisson equation, linear elastostatics, and p-Laplacian diffusion. Although our presentation demonstrates that the surrogate matrix methodology applies to each of these models equally well, we only employ a complete a priori analysis of the simplest model, the variable

coefficient Poisson equation. In our numerical experiments, we carefully verify the proven a priori convergence rates with the variable coefficient Poisson equation. We also include proof-of-concept demonstrations from numerical experiments with the linear elastostatics and p-Laplacian diffusion problems.

2. Notation and outline. Let V be a reflexive Banach space over $\mathbb{R}$, the field of real numbers, and let $V_h \subsetneq V$ be a finite-dimensional subspace. Consider a continuous and weakly coercive bilinear form $a : V \times V \to \mathbb{R}$ and a bounded linear functional $F \in V^*$, the topological dual of V.

In this paper, we are concerned with the solutions u, u_h, and $\widetilde{u}_h$ of the following three abstract variational problems.

$$\text{(2.1a)} \qquad \text{Find } u \in V \text{ satisfying} \qquad a(u,v) = F(v) \quad \text{for all } v \in V\,.$$
$$\text{(2.1b)} \qquad \text{Find } u_h \in V_h \text{ satisfying} \qquad a(u_h,v_h) = F(v_h) \quad \text{for all } v_h \in V_h\,.$$
$$\text{(2.1c)} \qquad \text{Find } \widetilde{u}_h \in V_h \text{ satisfying} \qquad \widetilde{a}(\widetilde{u}_h,v_h) = F(v_h) \quad \text{for all } v_h \in V_h\,.$$

In (2.1c), a *surrogate* bilinear form $\widetilde{a} : V_h \times V_h \to \mathbb{R}$ has been introduced. In order to properly define $\widetilde{a}(\cdot,\cdot)$, some additional assumptions on $a(\cdot,\cdot)$ are still required; see section 3.

The discrete variational problems (2.1b) and (2.1c) induce matrix equations for coefficients $\mathsf{u}, \widetilde{\mathsf{u}}$ in some $\mathbb{R}^N$,

$$\text{(2.2)} \qquad \mathsf{A}\mathsf{u} = \mathsf{f} \qquad \text{and} \qquad \widetilde{\mathsf{A}}\widetilde{\mathsf{u}} = \mathsf{f},$$

respectively. In the first case, fix a basis for V_h, say $\{\phi_i\}_{i=1}^N$. For this basis, each (i,j)-component of the stiffness matrix A is simply $\mathsf{A}_{ij} = a(\phi_j,\phi_i)$. In the following section, we present a methodology to construct a *surrogate* stiffness matrix $\widetilde{\mathsf{A}} \approx \mathsf{A}$ which can be used in place of the *true* stiffness matrix A. This methodology stands apart from technical details, such as differences in quadrature formulas. Section 4 provides a short (noncomprehensive) list of examples fitting into our framework. In section 5, we discuss the incorporation of nonhomogeneous boundary conditions and what we herein refer to as *the zero row sum property.* In section 6, sufficient conditions for the discrete stability of surrogate bilinear forms $\widetilde{a}(\cdot,\cdot)$ are briefly discussed. Next, in section 7, we perform a rigorous a priori error analysis of our approach applied to the variable coefficient Poisson equation. A brief description of our implementation is given in section 8. Then, in section 9, we document several numerical experiments. Here, a thorough verification of each error estimate in section 7 is given. This is complemented by performance measurements for the additional examples.

Throughout this article, we assume that $\Omega \subseteq \mathbb{R}^n$ is a bounded Lipschitz domain. For matrices $\mathsf{M} \in \mathbb{R}^{l\times m}$, define the ℓ^∞- and max-norms, $\|\mathsf{M}\|_\infty = \max_i \sum_j |\mathsf{M}_{ij}|$ and $\|\mathsf{M}\|_{\max} = \max_{i,j} |\mathsf{M}_{ij}|$. Likewise, for any function $v : \Omega \to \mathbb{R}$, we will use the similar notation, $\|v\|_0$, $\|v\|_1$, and $\|v\|_2$, for the canonical $L^2(\Omega)$-, $H^1(\Omega)$-, and $H^2(\Omega)$-norms, respectively. When dealing with a subset $T \subseteq \Omega$, denote the related $L^2(T)$-, $H^1(T)$-, and $H^2(T)$-norms by $\|v\|_{0,T}$, $\|v\|_{1,T}$, and $\|v\|_{2,T}$, respectively. For any simplex T and integer $0 \le q < \infty$, we denote the space of polynomials of degree at most q as $\mathcal{P}_q(T)$. All remaining notation will be defined as it arises.

3. Surrogate stiffness matrices. In this section, we present the constitutive elements of the surrogate matrix methodology. Our approach here is to gradually introduce the necessary concepts, all the while maintaining a clear sense of generality. In order to arrive at a tractable framework for our problems of interest, we gradually

refine the presentation from general n-dimensional spaces to only $n = 1, 2$, or 3 and from general Banach spaces to only $W^{1,p}(\Omega)$ (or products thereof), where $1 < p < \infty$. The intention of proceeding in this way is to indicate that the methodology can be applied to an extremely broad set of problems and, specifically, to most problems where finite element methods are traditionally applied.

3.1. Preliminary assumptions. Given a bounded domain $\Omega \subseteq \mathbb{R}^n$, assume that the true bilinear form can be expressed as

$$a(u,v) = \int_\Omega G(x, u(x), v(x))\,\mathrm{d}x \qquad \text{for all } u, v \in V. \tag{3.1a}$$

Additionally, upon defining $\operatorname{supp}(u) = \{y \in \Omega : u(y) \neq 0\}$ for smooth functions, make the following sparsity assumption:

$$G(x, u(y), v(y)) = 0 \qquad \text{whenever } y \notin \operatorname{supp}(u) \cap \operatorname{supp}(v). \tag{3.1b}$$

These assumptions permit us to consider the discretization of most classical differential operators. Indeed, in the assumptions above, the integrand $G(x, u, v)$ may induce distributional derivatives on its second and third arguments. Meanwhile, the first argument can be identified with the spatial argument of any associated variable coefficients. For example, in the weak form of a Poisson-type equation, $-\operatorname{div}(K\nabla u) = f$, with a variable, symmetric positive-definite diffusion tensor $K(x)$ (cf. subsection 4.1), we have the bilinear form

$$a_1(u,v) = \int_\Omega \nabla u(x)^\top K(x) \nabla v(x)\,\mathrm{d}x \qquad \text{for all } u, v \in V = H^1(\Omega)\,. \tag{3.2}$$

Here, if one takes any point $x \in \Omega$, the integrand in (3.1a) reduces to $G(x, u, v) = \nabla u^\top K(x) \nabla v$. Evidently, this G satisfies the sparsity assumption (3.1b).

3.2. Stencil functions. Let $\phi \in V$ be a test function with compact support in Ω, and, for any fixed $y \in \mathbb{R}^n$, define $\phi_y(x) = \phi(x - y)$. Now, consider any fixed set of ordered points $\mathbb{X} = \{x_i\}$ in Ω and recall (3.1a). Assuming that both $\phi_{x_i}, \phi_{x_j} \in V$, observe (via a simple change of variables) that

$$a(\phi_{x_j}, \phi_{x_i}) = \int_\Omega G(y, \phi_{x_j}(y), \phi_{x_i}(y))\,\mathrm{d}y = \int_{\Omega_\delta} G(x_i + y, \phi_\delta(y), \phi(y))\,\mathrm{d}y\,, \tag{3.3}$$

where $\delta = x_j - x_i$ and $\Omega_\delta = \operatorname{supp}(\phi) \cap \operatorname{supp}(\phi_\delta)$. In the second equality, passing from an integral over Ω to an integral over the subset $\Omega_\delta \subseteq \Omega$ follows immediately from the sparsity assumption (3.1b). For each fixed x_i, the affine structure of the identity above may be illuminated by collecting each contributing translation δ into the set $\mathcal{D}(x_i) = \{x_j - x_i : x_j \in \mathbb{X},\, a(\phi_{x_j}, \phi_{x_i}) \neq 0\}$ and defining a *stencil function*

$$\Phi_i^\delta(x) = \int_{\Omega_\delta} G(x + y, \phi_\delta(y), \phi(y))\,\mathrm{d}y \qquad \text{for each } \delta \in \mathcal{D}(x_i). \tag{3.4}$$

We have just reduced the computation of any $a(\phi_{x_j}, \phi_{x_i})$ to the evaluation of scalar-valued functions enumerated by affine coordinates $(x_i, x_j - x_i)$. Indeed,

$$a(\phi_{x_j}, \phi_{x_i}) = \begin{cases} \Phi_i^\delta(x_i) & \text{if } \delta = x_j - x_i \in \mathcal{D}(x_i), \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

In the present scenario, there may be a different set of translations $\mathcal{D}(x_i)$ for every point x_i. However, if each point is drawn from a point lattice, most of the sets $\mathcal{D}(x_i)$ are identical. This observation is the subject of the following subsection and a foundational principle in our approach.

Remark 3.1. In the scenario that the bilinear form is symmetric, $a(u,v) = a(v,u)$, it is natural to assume that the stencil functions (3.4) will inherit a similar symmetry. Indeed, under the equivalent symmetry condition $G(x,u,v) = G(x,v,u)$ almost everywhere, one may easily verify that if $\delta = x_j - x_i$, then

(3.6)
$$\Phi_i^{\delta}(x_i) = \int_{\Omega_\delta} G(x_i + y, \phi(y), \phi_\delta(y))\,\mathrm{d}y = \int_{\Omega} G(x_j + y, \phi_{-\delta}(y), \phi(y))\,\mathrm{d}y = \Phi_j^{-\delta}(x_j)$$

or, equivalently, $\Phi_i^{\delta}(x_i) = \Phi_j^{-\delta}(x_i + \delta)$.

3.3. Local stencil functions and locally structured meshes. An affine point lattice $\mathbb{L}$, from here on referred to only as a *lattice*, is a regularly spaced array of points in $\mathbb{R}^n$ where every point $x_i \in \mathbb{L}$ belongs to a neighborhood containing no other points in $\mathbb{L}$. In this paper, each (possibly finite) lattice is determined by a finite linearly independent set of translations in $\mathbb{R}^n$; i.e., $\mathbb{L} \subseteq \{\delta_0 + a_1\delta_1 + \cdots + a_l\delta_l \,:\, a_1, \ldots, a_l \in \mathbb{Z}\}$.

Assuming that the test function $\phi \in V$ is sufficiently localized and each point x_i is drawn from a lattice $\mathbb{L} \subseteq \Omega$, then each $\mathcal{D}(x_i)$ is a subset of a small number of admissible translations $\mathcal{D}(\mathbb{L}) = \bigcup\{\mathcal{D}(x_i) : x_i \in \mathbb{L}\}$, determined solely by the lattice structure. In such a scenario, every stencil function is closely related; i.e., $\Phi_i^{\delta}(x) = \Phi_j^{\delta}(x)$, whenever both are defined. Therefore, it is prudent to drop the subscript and define only one common stencil function $\Phi^{\delta}(x)$ for each $\delta \in \mathcal{D}(\mathbb{L})$. Clearly,

$$a(\phi_{x_j}, \phi_{x_i}) = \begin{cases} \Phi^{\delta}(x_i) & \text{if } \delta = x_j - x_i \in \mathcal{D}(\mathbb{L}), \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

We are interested in exploiting (3.7) for solving a wide variety of PDEs with curvilinear geometries. Toward this end, the following examples help motivate our construction further.

Remark 3.2. The main scope of this work consists only of classical lowest-order conforming Bubnov–Galerkin finite element methods. Therefore, from now on, we assume that V is a closed subset of $[W^{1,p}(\Omega)]^l$ for some $1 < p < \infty$ and $l \in \mathbb{N}$, where, unless explicitly stated otherwise, $l = 1$. This will allow us to define a basis for $V_h \subseteq V$ consisting only of (componentwise) finite element vertex functions [25]. The ideas here can be extended to high-order polynomial and nonuniform rational basis spline (NURBS) bases; cf. [22]. Generalizations to nonconforming methods or methods for more exotic energy spaces (e.g., $H(\mathrm{curl})$ or $H(\mathrm{div})$) are also possible.

3.3.1. The one-dimensional setting. Let $V = H_0^1(\Omega)$, where $\Omega = (0,1) \subseteq \mathbb{R}$, and fix a small translation $dx = 1/(N+1)$. Consider the scenario where each point, $x_i = x_{i-1} + dx$, evenly divides Ω and ϕ is the piecewise-linear hat function defined $\phi(x) = \max(1 - \frac{|x|}{dx}, 0)$. Let $V_h = \{v \in H_0^1(\Omega) : v|_t \in \mathcal{P}_1(t),\ t = (x_i, x_{i+1}) \text{ for each } 1 \le i \le N\}$, and identify each shifted hat function with the standard basis, $\phi_{x_i} = \phi_i \in V_h$. Here, we may define $\mathbb{L} = \{x_i\}$. In this case, for each $i \ge 1$, the value $a(\phi_i, \phi_j)$ can either be computed directly from (3.1a), in the standard way, or evaluated using (3.7), assuming that each Φ^{δ} is available at the onset of computation. Note that $\mathcal{D}(x_1) = \{0, dx\}$ and $\mathcal{D}(x_N) = \{-dx, 0\}$, but $\mathcal{D}(x_i) = \{-dx, 0, dx\}$ for each $2 \le i \le N-1$. Therefore, $\mathcal{D}(\mathbb{L}) = \{-dx, 0, dx\}$. Ultimately, defining each structured stencil function $\Phi^{\delta}(x) = \Phi((x_i, \delta); x)$ from an arbitrary candidate point x_i, one may verify that

$$a(\phi_j, \phi_i) = \begin{cases} \Phi^{\delta}(x_i) & \text{if } \delta = x_j - x_i \in \{-dx, 0, dx\}, \\ 0 & \text{otherwise,} \end{cases} \tag{3.8}$$

which is clearly the same format as (3.7).

Recall Remark 3.1. If $a(\cdot,\cdot)$ is symmetric, then, by (3.6), $\Phi^{dx}(x_i) = \Phi^{-dx}(x_i+dx)$ and the number of required stencil functions can be reduced to two. In some situations—e.g., when the zero row sum property can be employed (see section 5)—only a single stencil function is actually required.

3.3.2. Locally structured meshes with triangles. Let $m \in \mathbb{N}_0$. Beginning with a scaled Cartesian lattice $\mathbb{L}_m = 2^{-m}\mathbb{Z}^n$, it is useful to define its intersection with the closure of the right-angled reference simplex $\hat{T} = \{\hat{x} \in \mathbb{R}^n : \|\hat{x}\|_1 < 1, \hat{x} \cdot e_i > 0 \text{ for all } i = 1,\dots,n\}$. This simplicial lattice, $\hat{T}_m = \mathbb{L}_m \cap \overline{\hat{T}}$, can easily be transformed into a similar simplicial lattice T_m for any arbitrary simplex $T \subseteq \Omega$ via an affine transformation (see, e.g., Figure 1). Indeed, fixing the unique $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$ such that $T = \{A\hat{x} + b : \hat{x} \in \hat{T}\}$, the corresponding lattice is clearly $T_m = \{A\hat{x}_i + b : \hat{x}_i \in \hat{T}_m\}$. Note that there are no interior points, $\mathring{T}_m = T_m \cap T = \emptyset$, if $m < 2$. We also define the set of boundary points $\partial T_m = T_m \setminus \mathring{T}_m$, which is always nonempty.

Let $\Omega \subseteq \mathbb{R}^n$, where $n = 2, 3$. The utility of this transformation is evident upon considering a *macro*-triangulation of Ω, say $\mathcal{T}_H$, where each *macro*-element $T \in \mathcal{T}_H$ is endowed with a simplicial lattice T_m, as defined above. Here, $H = \max_{T\in\mathcal{T}_H} H_T$, where each $H_T = \operatorname{diam}(T)$ denotes the diameter of T. Notice that, for any fixed level $m \geq 0$, all interface points $x_i \in \partial T_m \cap \Omega$ are coincident with an interface lattice point on some neighboring simplex. We may now define the set of all vertices on level m:

$$\mathbb{X}_m = \bigcup \{T_m : T \in \mathcal{T}_H\}.$$

For every $\mathbb{X}_m$, there is a corresponding finite element mesh such that each vertex function within a fixed macroelement $T \in \mathcal{T}_H$ is self-similar. In the cases $n = 2$ or 3, we are left to define each triangle or tetrahedron whose vertices coincide with points in $\mathbb{X}_m$. Let $[y_0, y_1, \dots, y_k] \subseteq \mathbb{R}^n$ denote the convex combination of the points $y_0, y_1, \dots, y_k \in \Omega$. When $n = 2$, the natural construction begins by considering the following uniform subdivision of a triangle $T = [y_0, y_1, y_2] \in \mathcal{T}_H$ into a set of four equal-volume triangles:

(3.9)
$$\mathcal{S}(T) = \left\{ \left[y_0, \frac{y_0+y_1}{2}, \frac{y_0+y_2}{2}\right], \left[\frac{y_0+y_1}{2}, y_1, \frac{y_1+y_2}{2}\right], \left[\frac{y_0+y_2}{2}, \frac{y_1+y_2}{2}, y_2\right], \left[\frac{y_0+y_1}{2}, \frac{y_1+y_2}{2}, \frac{y_0+y_2}{2}\right] \right\}.$$

For an illustration of this $n = 2$ case, see Figure 1. For $n = 3$, see the construction in [14]. Further subdivisions can then be defined recursively, viz.,

$$\mathcal{S}^{m+1}(T) = \bigcup \{\mathcal{S}(t) : t \in \mathcal{S}^m(T)\} \qquad \text{for all } m \geq 1\,. \tag{3.10}$$

The set of all vertices in a given subdivision $\mathcal{S}^m(T)$ forms an evenly spaced set of points inside T. The set of vertices in $\mathcal{S}^1(T)$ clearly coincides with T_1, and one can easily verify from the recursive definition that the set of vertices in $\mathcal{S}^m(T)$ also coincides with T_m. We may finally define the sequence of *locally structured meshes*:

$$\mathcal{S}^m(\mathcal{T}_H) = \bigcup \{\mathcal{S}^m(T) : T \in \mathcal{T}_H\} \qquad \text{for all } m \geq 1\,. \tag{3.11}$$

Notice that for each m, $\mathbb{X}_m$ coincides with the set of all vertices in $\mathcal{S}^m(\mathcal{T}_H)$. Therefore, each point x_i in $\mathbb{X}_m$ can be identified with a vertex function ϕ_i supported by only the neighboring elements appearing in $\mathcal{S}^m(\mathcal{T}_H)$; see Figure 1.

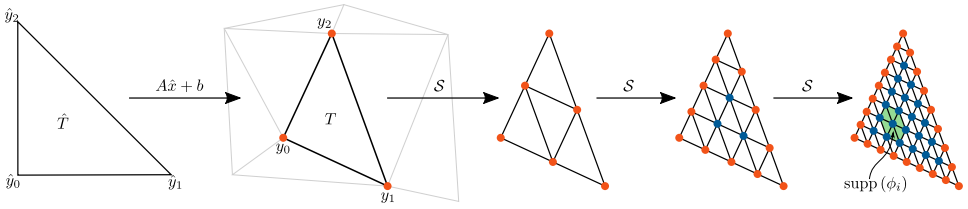


FIG. 1. *Illustration of three refinement steps of a single macroelement T for $n = 2$ with the corresponding vertex lattices T_0, T_1, T_2, and T_3. The interior lattice points $\mathring{T}_m$ are colored blue, and the boundary lattice points $\partial T_m = T_m \setminus \mathring{T}_m$ are colored in orange. Additionally, the support of an exemplary vertex function ϕ_i is shaded in green.*

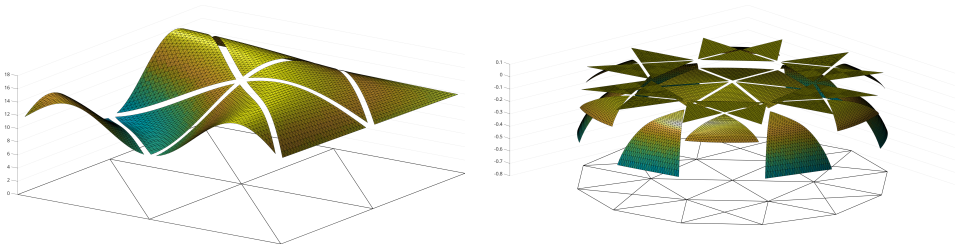


FIG. 2. *Left: Surface plots of the local stencil functions $\Phi_T^\delta(x)$, for the degenerate direction $\delta = 0$ and level $m = 5$, from the numerical experiments recounted in subsection 9.1.1. Here, the function is plotted over each subset $\mathrm{conv}(\mathring{T}_m) \subseteq T \in \mathcal{T}_H$. In this case, it clearly appears that each stencil function can be related to the restriction of a globally continuous function $\Phi_T^\delta(x)$. This is a result of the structure of the macromesh only. Right: Surface plots of the stencil functions $\Phi_T^\delta(x)$ after the first time step from the experiment recounted in subsection 9.3 for the eastern direction δ relative to each macroelement and level $m = 5$. Moreover, although they are clearly related, it is evident that the corresponding stencil functions lack any global smoothness property.*

Assume that the fine mesh level, m, is chosen large enough that one may find several points in $\mathbb{X}_m$ which lie in the interior of some macroelement T. If $x_i \in \mathring{T}_m$ is any such point, then the translation set $\mathcal{D}(x_i) = \mathcal{D}(\mathring{T}_m)$ will only contain translations aligned with edges appearing in the original subdivision $\mathcal{S}^1(T)$. Identifying $\phi_{x_i} = \phi_i$ we see that for every $x_i \in \mathring{T}_m$ and $x_j \in T_m$,

$$a(\phi_j, \phi_i) = \begin{cases} \Phi_T^\delta(x_i) & \text{if } \delta = (x_j - x_i) \in \mathcal{D}(\mathring{T}_m), \\ 0 & \text{otherwise,} \end{cases} \tag{3.12}$$

where $\Phi_T^\delta : \mathrm{conv}(\mathring{T}_m) \to \mathbb{R}$ is a local stencil function for the current level m and macroelement T and $\mathrm{conv}(\mathring{T}_m)$ is the convex hull of $\mathring{T}_m$. In general, notice that $\Phi_T^\delta \neq \Phi_T^{2\delta}$ are two different stencil functions, corresponding to the same direction but different mesh levels.

Consider the bilinear form (3.2) with a variable diffusion coefficient. A visualization of several corresponding local stencil functions, coming from locally structured meshes used in our numerical experiments, is given in Figure 2. It is clear from this figure that each Φ_T^δ has the potential to be a smooth function. We now come to the final essential component of our surrogate methodology: the approximation of Φ_T^δ.

Remark 3.3. The lattice structure of locally structured meshes is destroyed under smooth, non-affine transformations $\hat{T} \to \widetilde{T} \subseteq \Omega$. This offers a possible impediment to

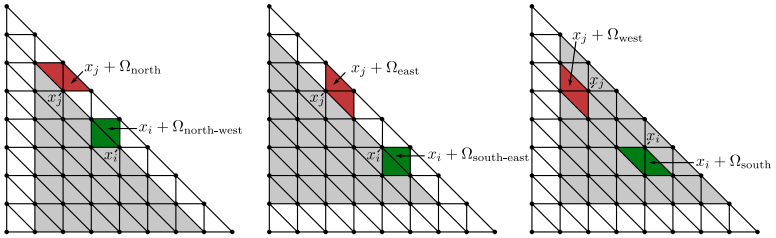


FIG. 3. *Illustrations of the domains T_δ in gray and Ω_δ for six exemplary directions δ. Left: Northern and northwestern direction. Middle: Eastern and southeastern direction. Right: Western and southern direction.*

our construction in the case of nonpolygonal domains $\overline{\Omega} \neq \overline{\Omega}_H = \bigcup_{T\in\mathcal{T}_H} \overline{T}$. In fact, if a globally continuous transformation $\varphi : \overline{\Omega}_H \to \overline{\Omega}$ is available, with $\varphi|_T$ a smooth bijection for every $T \in \mathcal{T}_H$, then an equivalent method can be found using local pullbacks of φ. For example, the bilinear form in (3.2), $a_1 : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$, simply transforms to $a_{1,H} : H^1(\Omega_H) \times H^1(\Omega_H) \to \mathbb{R}$, where

$$a_{1,H}(u,v) = \int_{\Omega_H} \nabla u(x)^\top K_H(x) \nabla v(x)\,\mathrm{d}x\,, \qquad K_H = \frac{D\varphi^{-1}(K\circ\varphi)D\varphi^{-\top}}{|\det(D\varphi^{-1})|}\,. \tag{3.13}$$

Therefore, from now on, we operate under the assumption that the macromesh $\mathcal{T}_H$ is geometrically conforming, $\overline{\Omega} = \overline{\Omega}_H$.

3.4. The inherited regularity of stencil functions. Our approach is to locally project each stencil function Φ_T^δ in (3.12) onto a high-dimensional space of polynomials and later use this projection to compute approximate values of the stiffness matrix A. Let $\mathcal{T}_H$ be a shape-regular simplicial mesh, and consider a locally structured mesh of level m subordinate to this mesh, $\mathcal{S}^m(\mathcal{T}_H)$. Before moving on, observe that definition (3.12) in fact holds for any $x_i, x_j \in T_m$ if x_i or $x_j \in \mathring{T}_m$. Therefore, due to the structure of the vertex functions in a locally structured mesh, the domain of each Φ_T^δ can actually be extended to a set $\overline{T}_\delta$ lying between $\mathrm{conv}(\mathring{T}_m)$ and $\overline{T}$. Indeed, identifying the test function in (3.4) with the ith vertex function, $\phi(x) = \phi_i(x + x_i)$, for an arbitrary vertex $x_i \in \mathring{T}_m$, define

$$T_\delta = \{x \in T : x + y \in T \text{ for all } y \in \Omega_\delta = \mathrm{supp}(\phi) \cap \mathrm{supp}(\phi_\delta)\}. \tag{3.14}$$

From now on, we assume $\Phi_T^\delta : \overline{T}_\delta \to \mathbb{R}$. See Figure 3 for a depiction of the sets in a triangular mesh, and note that $T_{-\delta} = T_\delta + \delta$ for every $\delta \in \mathcal{D}(T_m)$.

For any $T \in \mathcal{T}_H$, let $\mathcal{P}_q(T_\delta)$ denote the space of polynomials of degree at most q on the simplex $\overline{T}_\delta$ and let $\Pi_T^\delta : C^0(\overline{T}_\delta) \to \mathcal{P}_q(T_\delta)$ be an L^∞-continuous projection operator, $\Pi_T^\delta \circ \Pi_T^\delta = \Pi_T^\delta$. For each macroelement $T \in \mathcal{T}_H$ and level $m \in \mathbb{N}$, define the *surrogate stencil function* $\widetilde{\Phi}_T^\delta : T_\delta \to \mathbb{R}$ to be the corresponding polynomial projection of Φ_T^δ. Namely,

$$\widetilde{\Phi}_T^\delta = \Pi_T^\delta\, \Phi_T^\delta. \tag{3.15}$$

In order to correctly argue that a polynomial approximation of Φ_T^δ is feasible, it is necessary to classify its regularity depending on the problem at hand. In the following proposition, we show, under the modest assumptions above, that if $G(\cdot|_T, \cdot, \cdot)$ is a polynomial in its first argument, then Φ_T^δ is also a polynomial of the same degree.

LEMMA 3.1. *Fix a simplex $T \in \mathcal{T}_H$. Assume that the bilinear form $a(\cdot,\cdot)$ in* (2.1a) *satisfies assumptions* (3.1a) *and* (3.1b)*, where the integrand $G(\cdot|_T,\cdot,\cdot)$ is a polynomial of at most degree q in its first argument. Then, for any locally structured mesh $\mathcal{S}^m(\mathcal{T}_H)$, as defined above, every local stencil function $\Phi_T^\delta \in \mathcal{P}_q(T_\delta)$ is a polynomial of the same degree.*

Proof. Recall definition (3.4). For every $x_i, x_j \in \mathbb{X}_m$, every stencil function $\Phi((x_i, x_j - x_i), x)$ is defined with a test function $\phi \in V$. Fixing any arbitrary vertex $x_i \in \mathring{T}_m$, identify this test function with the ith vertex function, $\phi(x) = \phi_i(x + x_i)$. Let $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}_0$ denote a standard multiindex, $|\alpha| = \sum_{i=1}^n |\alpha_i|$ and $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. By assumption, we may express $G(x, \phi_\delta(y), \phi(y)) = \sum_{|\alpha|\le l} c_\alpha(y) x^\alpha$, where each coefficient function $c(y)$ has support only in $\Omega_\delta = \operatorname{supp}(\phi) \cap \operatorname{supp}(\phi_\delta)$. Moreover, if $x + y \in T$, then

$$G(x+y, \phi_\delta(y), \phi(y)) = \sum_{|\alpha|\le l} c_\alpha(y)(x+y)^\alpha = \sum_{|\alpha|\le l} \sum_{|\nu|\le\alpha} \binom{\alpha}{\nu} c_\alpha(y) y^\nu x^{\alpha-\nu} \tag{3.16}$$

is clearly an equal degree polynomial in the variable x. The proof is completed by noting that the integral in (3.4) is performed only over the variable $y \in \Omega_\delta$ and so the stencil function acts like a convolution. Indeed, under the assumption $x \in T$, only the subset of points $x \in T_\delta = \{x \in T : x + y \in T \text{ for all } y \in \Omega_\delta\}$ guarantee that (3.16) holds at every point of integration y. In this case, by linearity of integration, Φ_T^δ is a member of $\mathcal{P}_q(T_\delta)$, with its coefficients defined by the associated integrals of the y-dependent functions in the right-hand side of (3.16). ∎

COROLLARY 3.2. *In the setting of Lemma* 3.1*, $\widetilde{\Phi}_T^\delta = \Phi_T^\delta$.*

Proof. Since $\Phi_T^\delta \in \mathcal{P}_q(T_\delta)$, we immediately see that $\Pi_T^\delta \Phi_T^\delta = \Phi_T^\delta$. ∎

3.5. Surrogate stiffness matrices. The main goal of the entire effort above is to guide us in reducing the vast majority of the finite element assembly process to the evaluation of a small set of functions which can, in fact, be locally approximated by polynomials. As in subsection 3.3.2, let $\mathcal{T}_H$ be a shape-regular triangulation of a domain Ω into disjoint *macro*-simplices T.

Our construction of the surrogate matrix $\widetilde{\mathsf{A}}$ is obviously built to exploit the local lattice structure of locally structured meshes. As argued previously, a surrogate stencil function $\widetilde{\Phi}_T^\delta$ can be used to approximate any matrix entry A_{ij} coming from a locally structured mesh if at least one of the corresponding vertices x_i or x_j belongs to $\mathring{T}_m$. This leaves us to define only the nonzero matrix entries coming from the mutual interaction of vertex functions at the boundaries of the macroelements. Although these entries could also be approximated by surrogate stencil functions—in this case, these additional functions would be defined on each subsimplex of the macromesh $\mathcal{T}_H$—let us assume that they are computed directly. Because the growth of the macromesh boundary and interface interactions grow at an order of magnitude less than the interior interactions, computing these matrix entries directly does not affect the asymptotic performance of the methodology. Finally, letting $\partial\mathbb{X}_m = \bigcup_{T\in\mathcal{T}_H} \partial T_m$ denote the union of all macromesh boundary vertices, we define the general surrogate stiffness matrix

$$\widetilde{\mathsf{A}}_{ij} = \begin{cases} \int_\Omega G(y, \phi_j(y), \phi_i(y))\,\mathrm{d}y & \text{if both } x_i \text{ and } x_j \in \partial\mathbb{X}_m, \\ \widetilde{\Phi}_T^\delta(x_i) & \text{if } \delta = (x_j - x_i) \in \mathcal{D}(\mathring{T}_m) \text{ and } x_i \text{ or } x_j \in \mathring{T}_m, \\ 0 & \text{otherwise.} \end{cases} \tag{3.17}$$

Remark 3.4. Due to the presence of the surrogate stencil functions $\widetilde{\Phi}_T^\delta$, even if A is symmetric, $\widetilde{\mathsf{A}}$ will generally not be. However, recalling (3.6), observe that if $a(\cdot,\cdot)$ is symmetric, then $\Phi_T^\delta(x) = \Phi_T^{-\delta}(x+\delta)$. Therefore, if we use related projection operators

$$\left[\Pi_T^\delta \Phi_T^\delta\right](x) = \left[\Pi_T^{-\delta} \Phi_T^{-\delta}\right](x+\delta) \tag{3.18}$$

for each opposing direction δ and $-\delta$, then $\widetilde{\mathsf{A}}$ will be symmetric. Indeed, if $\delta = x_j - x_i$, then

$$\widetilde{\mathsf{A}}_{ij} = \Pi_T^\delta \Phi_T^\delta(x_i) = \Pi_T^{-\delta}\Phi_T^{-\delta}(x_i+\delta) = \Pi_T^{-\delta}\Phi_T^{-\delta}(x_j) = \widetilde{\mathsf{A}}_{ji}\,. \tag{3.19}$$

4. Examples. In this section, we present three example problems which easily fit into the framework above.

4.1. The variable coefficient Poisson equation. Consider the Poisson-type equation $-\mathrm{div}(K\nabla u) = f$ in Ω, $u = 0$ on $\partial\Omega$, with a load $f \in L^2(\Omega)$ and a variable, symmetric positive-definite tensor K. Furthermore, assume that for each index a, b, $K_{ab} \in \mathcal{P}_q(\Omega)$. Recall (3.2), and note that we have already shown that the weak form of this problem can be cast into the framework above.

Assume that, within some set $T \subseteq \Omega$, each vertex function ϕ_i is a translation of a fixed test function $\phi(x) = \phi_i(x - x_i)$. Then for each ϕ_i, ϕ_j, the stiffness matrix entry

$$\mathsf{A}_{ij} = \int_\Omega \nabla\phi_i(x)^\top K(x) \nabla\phi_j(x)\,\mathrm{d}x \tag{4.1}$$

can equally well be expressed as the evaluation (at the point x_i) of a stencil function, which, by Lemma 3.1, is simply a polynomial of the same degree as the diffusion tensor K. In the case of locally structured meshes, there is a locally defined stencil function $\Phi_T^\delta : \Omega \to \mathbb{R}$ for each macroelement T and level m. In this case, each $\Phi_T^\delta : \overline{T}_\delta \to \mathbb{R}$ is a polynomial (of degree at most q) on T_δ.

4.2. Lamé–Navier linearized elasticity. Let $\vec{\nabla}$ and Div denote the row-wise distributional gradient and divergence, respectively. Now define $\epsilon(u) = \frac{1}{2}[\vec{\nabla}u + (\vec{\nabla}u)^\top]$ to be the symmetric gradient operator $\epsilon : H^1(\Omega)^n \to L^2(\Omega)^n$, where $n \geq 2$. Consider the following standard PDE model for the displacement $u \in H_0^1(\Omega)^n$ of a linearly elastic isotropic material: $-\mathrm{Div}\,\sigma = \vec{f}$, where the stress $\sigma = 2\mu\epsilon(u) + \lambda I \mathrm{div}\,u$ and the load $f \in L^2(\Omega)^n$.

The weak form of this equation is well known in the literature [20], and the associated bilinear form is simply

$$a_2(u,v) = \int_\Omega 2\mu\epsilon(u) : \epsilon(v) + \lambda \mathrm{div}(u)\,\mathrm{div}(v)\,\mathrm{d}x \qquad \text{for all } u, v \in \left[H_0^1(\Omega)\right]^n. \tag{4.2}$$

This bilinear form obviously satisfies assumptions (3.1a) and (3.1b). If we assume that the Lamé parameters $\mu, \lambda : \Omega \to \mathbb{R}$ are piecewise polynomials on a collection of disjoint subdomains $T \in \mathcal{T}_H$, then each associated stencil function is also a piecewise polynomial.

4.3. *p*-Laplacian diffusion. For any $1 < p < \infty$, let $\Delta_p u = \mathrm{div}(|\nabla u|^{p-2}\nabla u)$ be the p-Laplacian operator. Fix a valid parameter p, and consider the nonlinear diffusion equation $\frac{\partial u}{\partial t} - \Delta_p u = f$, where $f \in L^p(\Omega)$. A simple Euler time stepping scheme replaces the time derivative $\frac{\partial u}{\partial t}$ by the quotient $\frac{u_{k+1} - u_k}{dt}$, where $dt > 0$ is

a fixed time step parameter. Choosing backward Euler time stepping and defining $f_k = f(k \cdot dt)$, we arrive at a semidiscrete nonlinear elliptic PDE for the solution variable u_k, which must be solved at each step $k \in \mathbb{N}$: $u_k - dt\Delta_p u_k = dt\, f_k + u_{k-1}$. Upon fixed point linearization of the weak form of this equation, we uncover the following bilinear form:

$$b(u,v) = \int_\Omega dt\, |\nabla \widetilde{u}|^{p-2}\, \nabla u \cdot \nabla v + u\, v \,\mathrm{d}x \qquad \text{for all } u, v \in W^{1,p}(\Omega)\,. \tag{4.3}$$

Here, the variable coefficient $\widetilde{u} \in W^{1,p}(\Omega)$ is usually identified with the previous solution iteration in the associated fixed point algorithm (cf. subsection 9.3).

The bilinear form $b(\cdot,\cdot)$ can easily be placed into the form of (3.1a), and each matrix entry can therefore be superseded by stencil function evaluations, $\Phi_T^\delta(x)$, as in (3.4). Alternatively, one may split $b(\cdot,\cdot)$ into a mass term $m(\cdot,\cdot)$ and a dt-weighted stiffness term $a_3(\cdot,\cdot)$. Specifically, $b(u,v) = m(u,v) + dt \cdot a_3(u,v)$, where

$$m(u,v) = \int_\Omega u\, v \,\mathrm{d}x \qquad \text{and} \qquad a_3(u,v) = \int_\Omega |\nabla \widetilde{u}|^{p-2}\, \nabla u \cdot \nabla v \,\mathrm{d}x\,. \tag{4.4}$$

With this observation in hand, we see that $b(u,v)$ may be discretized by a linear combination of independent surrogates; one for $m(\cdot,\cdot)$ and one for $a_3(\cdot,\cdot)$ (cf. subsection 9.3). In either approach, the variable coefficient $|\nabla \widetilde{u}(x)|^{p-2}$ will generally not remain a polynomial in a subdomain of Ω, and the accuracy of a surrogate stencil function $\widetilde{\Phi}_T^\delta$ will reflect the local regularity of the solution from the previous iteration, $\widetilde{u}$.

5. Boundary conditions and the zero row sum property. It is generally appropriate to define the surrogate stiffness matrix componentwise by the rule given in (3.17). Nevertheless, in some problems the operator to be discretized has a kernel which is not guaranteed to be respected by the surrogate. In such scenarios, it is possible that better performance and accuracy can be achieved if elements of this kernel are incorporated into the construction of the surrogate. This occurrence is most easily illustrated with the Poisson example from subsection 4.1.

Consider the bilinear form $a_1 : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$, defined in (3.2). Define $V_h^{\mathrm{ext}} = \{v \in H^1(\Omega) : v|_t \in \mathcal{P}_1(t) \text{ for each } t \in \mathcal{S}^m(\mathcal{T}_H)\}$ and $V_h = \{v \in H_0^1(\Omega) : v|_t \in \mathcal{P}_1(t) \text{ for each } t \in \mathcal{S}^m(\mathcal{T}_H)\} \subseteq V_h^{\mathrm{ext}}$. Let the corresponding vertex function bases be $\{\phi_i\} \subseteq \{\phi_i^{\mathrm{ext}}\}$ with $\phi_i = \phi_i^{\mathrm{ext}}$ for $1 \le i \le N$. In most finite element software, a space like V_h^{ext} is used to impose Dirichlet boundary conditions. Indeed, a "lift" of the Dirichlet data, say, $u_h^{\mathrm{ext}} = \sum_i \mathsf{u}_i^{\mathrm{ext}} \phi_i^{\mathrm{ext}}$, is generally constructed from a linear combination of the set $\{\phi_i^{\mathrm{ext}}\} \setminus \{\phi_i\}$. Then, taking $\mathsf{A}_{ij}^{\mathrm{ext}} = a(\phi_j^{\mathrm{ext}}, \phi_i)$ for each valid i, j, a modified load vector $\mathsf{f}^{\mathrm{ext}} = \mathsf{f} - \mathsf{A}^{\mathrm{ext}}\mathsf{u}^{\mathrm{ext}}$ is used in computation.

It is obvious that $\nabla 1 = 0$, and so $a_1(1, v) = 0$ for any $v \in H^1\Omega$. Therefore, by the partition of unity property $\sum_i \phi_i^{\mathrm{ext}} = 1$, the zero row sum of the matrix $\mathsf{A}^{\mathrm{ext}}$ also vanishes. Namely, $\sum_j \mathsf{A}_{ij}^{\mathrm{ext}} = 0$. This property may be induced in the corresponding surrogate matrix if we simply define $\widetilde{\mathsf{A}}_{ii}^{\mathrm{ext}} = -\sum_{j \ne i} \widetilde{\mathsf{A}}_{ij}^{\mathrm{ext}}$ for every $x_i \in \mathbb{X}_m$, where $\widetilde{\mathsf{A}}_{ij}^{\mathrm{ext}} = \mathsf{A}_{ij}^{\mathrm{ext}}$ for every j where A_{ij} is not defined, and $\widetilde{\mathsf{A}}_{ij}^{\mathrm{ext}} = \widetilde{\mathsf{A}}_{ij}$ otherwise. With this extra condition, the surrogate matrix (3.17) actually requires one fewer independent stencil function; i.e., $\Phi_T^0 = -\sum_{\delta \in \mathcal{D}(T_m) \setminus \{0\}} \Phi_T^\delta$, for every $T \in \mathcal{T}_H$. By this definition, although $\widetilde{\mathsf{A}}$ does not satisfy the zero row sum property, the matrix $\mathsf{A} - \widetilde{\mathsf{A}}$ does. Indeed,

$$\mathsf{A}_{ii} - \widetilde{\mathsf{A}}_{ii} = -\sum_{j \ne i} \left(\mathsf{A}_{ij}^{\mathrm{ext}} - \widetilde{\mathsf{A}}_{ij}^{\mathrm{ext}}\right) = -\sum_{j \ne i} \left(\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}\right). \tag{5.1}$$

In this way, the stiffness matrix coming from linearized elasticity (4.2) is similar to the stiffness matrix coming from the Laplacian. Indeed, the zero row sum property can be incorporated into its surrogate via a straightforward generalization.

6. Analyzing the surrogate discretization. In this section, we define and motivate what we see as some of the most essential features in the analysis of our surrogate methods. We begin with a review of discrete stability in the context of (2.1c). We then touch on the concept of spectral convergence of the surrogate matrix $\widetilde{\mathsf{A}} \to \mathsf{A}$, which helps us motivate the need to control $\|\mathsf{A} - \widetilde{\mathsf{A}}\|_{\max}$. This specific quantity will repeatedly appear in the a priori error analysis in section 7. Recall that q is the polynomial order of the image of the projection operator Π_T^δ appearing in the definition of the surrogate stencil function (3.15). Lemma 6.3 demonstrates that $\|\mathsf{A} - \widetilde{\mathsf{A}}\|_{\max} \to 0$ algebraically, at a rate dependent on the minimum of q and the local regularity r of the diffusion tensor K.

6.1. Discrete stability. Let $S = \{v \in V : \|v\|_V = 1\}$ be the surface of the unit ball in V. Recall (2.2), and assume that the discretization $\mathsf{Au} = \mathsf{f}$ is stable. In the present context, this is equivalent to the existence of a constant $\alpha > 0$ such that

$$\sup_{v_h \in V_h \cap S} a(w_h, v_h) \geq \alpha \|w_h\|_V \quad \text{for all } w_h \in V_h \,. \tag{6.1a}$$

Likewise, in order for the surrogate discretization $\widetilde{\mathsf{A}}\widetilde{\mathsf{u}} = \mathsf{f}$ to be stable, we must show that there exists a constant $\widetilde{\alpha} > 0$ such that

$$\sup_{v_h \in V_h \cap S} \widetilde{a}(w_h, v_h) \geq \widetilde{\alpha} \|w_h\|_V \quad \text{for all } w_h \in V_h \,. \tag{6.1b}$$

Inequality (6.1b) guarantees that $\widetilde{\mathsf{A}}\widetilde{\mathsf{u}} = \mathsf{f}$ has a unique solution and that $\|\widetilde{u}_h\|_V \leq \widetilde{\alpha}^{-1}\|F\|_{V^*}$. Equally important, however, it is a necessary precursor to Strang's first lemma, which in some cases can be used, in part, to show that $\widetilde{u}_h$ converges to the exact solution u (see, e.g., subsection 7.2).

6.2. Spectral convergence. By a direct analysis of the singular values of A, (6.1b) can sometimes be proven by showing that the spectrum of $\widetilde{\mathsf{A}}$ converges to the spectrum of A at a fast enough rate. The main takeaway from this section is that spectral convergence can be guaranteed by showing that $\widetilde{\mathsf{A}} \to \mathsf{A}$ in the matrix maximum norm, $\|\cdot\|_{\max}$. Before moving on, denote the k-smallest eigenvalue of a matrix $\mathsf{M} \in \mathbb{R}^{N\times N}$ as $\lambda_k(\mathsf{M})$, and let $\ell(\mathsf{M}) = \max_{1\leq i\leq N} \#\{\mathsf{M}_{ij} \neq 0 \text{ where } 1 \leq j \leq N\}$ be the maximum number of nonzero components in M, taken across all individual rows.

PROPOSITION 6.1. *Let $\mathsf{M}, \mathsf{N} \in \mathbb{R}^{N\times N}$ be symmetric matrices. Then, for each $k = 1, \ldots, N$, it holds that*

$$|\lambda_k(\mathsf{M}) - \lambda_k(\mathsf{N})| \leq \|\mathsf{M} - \mathsf{N}\|_\infty \,. \tag{6.2}$$

The proof of this proposition is standard, so it is placed in Appendix A. Because A and $\widetilde{\mathsf{A}}$ have the same sparsity pattern, the next result follows readily.

COROLLARY 6.2. *Let A and $\widetilde{\mathsf{A}}$ be the true and surrogate stiffness matrices in* (2.2)*, respectively. If both A and $\widetilde{\mathsf{A}}$ are real symmetric matrices, then*

$$|\lambda_k(\mathsf{A}) - \lambda_k(\widetilde{\mathsf{A}})| \leq \ell(\mathsf{A} - \widetilde{\mathsf{A}}) \cdot \|\mathsf{A} - \widetilde{\mathsf{A}}\|_{\max} \,, \qquad k = 1, \ldots, N. \tag{6.3}$$

Moreover, $\ell(\mathsf{A} - \widetilde{\mathsf{A}}) \leq 7$ when $n = 2$, and $\ell(\mathsf{A} - \widetilde{\mathsf{A}}) \leq 15$ when $n = 3$.

Proof. Recall definition (3.17). Inequality (6.3) follows trivially from Proposition 6.1 because both A and $\widetilde{\mathsf{A}}$ have the same sparsity pattern. Next, $\widetilde{\mathsf{A}}_{ij} \neq \mathsf{A}_{ij}$ only if either i or j corresponds to a vertex basis function with support lying entirely inside some macroelement $T \in \mathcal{T}_H$. Without loss of generality, assume that i corresponds to such a basis function $\operatorname{supp}(\phi_i) \subseteq T$. In the case $n = 2$, the support of this vertex function consists of exactly 6 triangular elements and 7 vertices; see, e.g., Figure 1. Thus, there are only 7 vertex functions ϕ_j (with j possibly equal to i) whose support will intersect $\operatorname{supp}(\phi_i)$ and $\ell(\mathsf{A} - \widetilde{\mathsf{A}}) \leq 7$. A similar argument follows for $n = 3$. ∎

Remark 6.1. As stated above, if $\|\mathsf{A} - \widetilde{\mathsf{A}}\|_{\max} \to 0$ fast enough, then Corollary 6.2 can be used in proving the stability condition (6.1b). However, (6.3) is generally a very pessimistic bound, and, when available, we recommend using more direct means to prove discrete stability (see, e.g., Theorem 7.1). Nevertheless, this result illustrates the importance of controlling $\|\mathsf{A} - \widetilde{\mathsf{A}}\|_{\max}$, which is a central feature in all of the coming analysis.

6.3. Controlling $\|\mathsf{A} - \widetilde{\mathsf{A}}\|_{\max}$ with the variable coefficient Poisson equation. Before we begin, some new notation is required. For any tensor $K : \Omega \to \mathbb{R}^{n\times n}$, define $\|K\|_{L^\infty(\Omega)} = \max_{a,b} \|K_{ab}\|_{L^\infty(\Omega)}$; likewise, for any $r \geq 0$, define $|K|_{W^{r+1,\infty}(T)} = \max_{a,b} |K_{ab}|_{W^{r+1,\infty}(T)}$. From now on, the notation $A \lesssim B$ will be used when two mesh-dependent quantities A and B satisfy an inequality $A \leq CB$, where C is some positive H-independent constant. Likewise, when $A \lesssim B$ and $B \lesssim A$, we write $A \eqsim B$. Recall that for a macromesh $\mathcal{T}_H$, the diameter of a single element $T \in \mathcal{T}$ is denoted H_T and the mesh size is denoted $H = \max_{T\in\mathcal{T}_H} H_T$. We also denote the fine-scale element diameter $h_T = 2^{-m} H_T$ for each $T \in \mathcal{T}_H$ and $h = 2^{-m} H$.

LEMMA 6.3. *Let A and $\widetilde{\mathsf{A}}$, respectively, be the true and surrogate stiffness matrices corresponding to the bilinear form* (3.2). *Namely, let each component of A be given by* (4.1) *and each component of $\widetilde{\mathsf{A}}$ be defined by* (3.17) *with $G(x, u(y), v(y)) := \nabla u(y)^\top K(x) \nabla v(y)$. Fix $T \in \mathcal{T}_H$ and $0 \leq r \leq q$. If $K_{ab}|_T \in W^{r+1,\infty}(T)$ for each index a, b, then*

$$\left\|\mathsf{A} - \widetilde{\mathsf{A}}\right\|_{\max,T_m} \lesssim h_T^{n-2} H_T^{r+1} |K|_{W^{r+1,\infty}(T)}\,, \tag{6.4a}$$

where $\|\mathsf{C}\|_{\max,T_m} = \max\{|\mathsf{C}_{ij}| : x_i, x_j \in T_m\}$ for any matrix C. Moreover, if each component $K_{ab} \in W^{r+1,\infty}(\mathcal{T}_H) = \prod_{T\in\mathcal{T}_H} W^{r+1,\infty}(T)$, then

$$\left\|\mathsf{A} - \widetilde{\mathsf{A}}\right\|_{\max} \lesssim h^{n-2} H^{r+1} |K|_{W^{r+1,\infty}(\mathcal{T}_H)}\,. \tag{6.4b}$$

Proof. We prove only (6.4a); (6.4b) then follows immediately. Recall (3.17) and fix $T \in \mathcal{T}_H$. Let i and j be the indices of the maximal value $|\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}| = \left\|\mathsf{A} - \widetilde{\mathsf{A}}\right\|_{\max,T_m}$. Next, because the theorem trivially holds in the degenerate case $\left\|\mathsf{A} - \widetilde{\mathsf{A}}\right\|_{\max,T_m} = 0$, we proceed under the assumption that $\mathsf{A}_{ij} \neq \widetilde{\mathsf{A}}_{ij}$. Notably, it follows from Corollary 3.2 that if each $K_{ab}|_T \in \mathcal{P}_q(T)$, then $\widetilde{\mathsf{A}}_{ij} = \mathsf{A}_{ij}$, and, therefore, we find ourselves in the scenario where the diffusion tensor $K|_T$ is not a polynomial (of degree at most q). Here, we may also freely assume that $i \neq j$ because of (5.1). Indeed, for each i, $|\mathsf{A}_{ii} - \widetilde{\mathsf{A}}_{ii}| \leq \sum_{j\neq i} |\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}| \leq \ell(\mathsf{A} - \widetilde{\mathsf{A}}) \cdot \max_{j\neq i} |\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}|$.

To fix notation in the remainder of the proof, we take $\phi = \phi_i$ and express

$$\widetilde{\mathsf{A}}_{ij} = \left[\Pi_T^\delta \int_{\Omega_\delta} \nabla\phi_\delta(y)^\top K(\cdot + y)\, \nabla\phi(y)\, \mathrm{d}y\right](x_i) \tag{6.5}$$

for some nonzero $\delta \in \mathcal{D}(T_m)$. Let $\mathcal{I}_T : C^0(\overline{T}) \to \mathcal{P}_r(T)$ be the local Lagrange interpolant, and define $\big[\mathcal{I}_T^{n\times n} K\big]_{ab} = \mathcal{I}_T K_{ab}$, for each index a, b. Splitting the two matrix entries A_{ij} and $\widetilde{\mathsf{A}}_{ij}$ into polynomial and nonpolynomial parts and rewriting

(6.6)
$$\int_\Omega \nabla\phi_j(x)^\top \big[K - \mathcal{I}_T^{n\times n} K\big](x)\nabla\phi_i(x)\,\mathrm{d}x = \int_{\Omega_\delta} \nabla\phi_\delta(y)^\top \big[K - \mathcal{I}_T^{n\times n} K\big](x_i + y)\nabla\phi(y)\,\mathrm{d}y\,,$$

(6.7)
$$\int_\Omega \nabla\phi_j(x)^\top \big[\mathcal{I}_T^{n\times n} K\big](x)\nabla\phi_i(x)\,\mathrm{d}x = \int_{\Omega_\delta} \nabla\phi_\delta(y)^\top \big[\mathcal{I}_T^{n\times n} K\big](x_i + y)\nabla\phi(y)\,\mathrm{d}y\,,$$

we find that

(6.8)
$$\mathsf{A}_{ij} = \int_\Omega \nabla\phi_j^\top\, \mathcal{I}_T^{n\times n} K\, \nabla\phi_i\,\mathrm{d}x + \int_{\Omega_\delta} \nabla\phi_\delta(y)^\top \big[K - \mathcal{I}_T^{n\times n} K\big](x_i + y)\nabla\phi(y)\,\mathrm{d}y,$$
$$\widetilde{\mathsf{A}}_{ij} = \int_\Omega \nabla\phi_j^\top\, \mathcal{I}_T^{n\times n} K\, \nabla\phi_i\,\mathrm{d}x + \left[\Pi_T^\delta \int_{\Omega_\delta} \nabla\phi_\delta(y)^\top \big[K - \mathcal{I}_T^{n\times n} K\big](\cdot + y)\nabla\phi(y)\,\mathrm{d}y\right](x_i).$$

Upon canceling the first two terms in the expressions above, we arrive at the inequality

$$\big|\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}\big| \le |\beta_{ij}(x_i)| + |(\Pi_T^\delta \beta_{ij})(x_i)|\,, \tag{6.9}$$

where $\beta_{ij}(x) = \int_{\Omega_\delta} \nabla\phi_\delta(y)^\top \big[K - \mathcal{I}_T^{n\times n} K\big](x+y)\nabla\phi(y)\,\mathrm{d}y$. Recall that the projection $\Pi_T^\delta : C^0(\overline{T}_\delta) \to \mathcal{P}(T_\delta)$ is continuous in the $L^\infty(\Omega)$ norm. Therefore,

$$\begin{aligned} |(\Pi_T^\delta \beta_{ij})(x_i)| &\le \|\Pi_T^\delta \beta_{ij}\|_{L^\infty(T_\delta)} \\ &\lesssim \|\beta_{ij}\|_{L^\infty(T_\delta)} \lesssim \|K - \mathcal{I}_T^{n\times n} K\|_{L^\infty(T)} \|\nabla\phi_\delta \cdot \nabla\phi\|_{L^1(\Omega_\delta)}\,. \end{aligned} \tag{6.10}$$

A standard scaling argument shows that $\|\nabla\phi_\delta \cdot \nabla\phi\|_{L^1(\Omega_\delta)} \lesssim h_T^{n-2}$. This, together with the well-known property $\|K_{ab} - \mathcal{I}_T K_{ab}\|_{L^\infty(T)} \lesssim H_T^{r+1} |K_{ab}|_{W^{r+1,\infty}(T)}$, yields the sufficient result

$$|\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}| \lesssim h_T^{n-2} H_T^{r+1} |K|_{W^{r+1,\infty}(T)}\,. \tag{6.11}$$

∎

Remark 6.2. The proof of Lemma 6.3 can be read as a blueprint which extends to the settings of the bilinear forms $a_2(\cdot,\cdot)$, $a_3(\cdot,\cdot)$, defined in (4.2) and (4.4). Indeed, when $a_2(\cdot,\cdot)$ is considered, the only significant modification to the proof above is that an interpolation operator $\mathcal{I}_T : C^0(\overline{T}) \to \mathcal{P}_r(T)$ must be introduced for each Lamé parameter μ and λ. The adaption to the setting $a(\cdot,\cdot) = a_3(\cdot,\cdot)$ is obvious. Ultimately,

(6.12)
$$\big\|\mathsf{A} - \widetilde{\mathsf{A}}\big\|_{\max} \lesssim h^{n-2} H^{r+1} \cdot \begin{cases} \big|\lambda\big|_{W^{r+1,\infty}(\mathcal{T}_H)} + \big|\mu\big|_{W^{r+1,\infty}(\mathcal{T}_H)} & \text{if } a(\cdot,\cdot) = a_2(\cdot,\cdot)\,, \\ \big||\nabla\widetilde{u}|^{p-2}\big|_{W^{r+1,\infty}(\mathcal{T}_H)} & \text{if } a(\cdot,\cdot) = a_3(\cdot,\cdot)\,. \end{cases}$$

Moreover, the proof may be easily modified to permit surrogates $\widetilde{\mathsf{A}}$ without the zero row sum property. Likewise, scenarios involving fewer derivatives (which generally do not possess the zero row sum property), e.g., $a(\cdot,\cdot) = m(\cdot,\cdot)$, have similar bounds but invoke a different scaling in h.

7. A priori error estimation for the variable coefficient Poisson equation. In this section, we present a thorough analysis of a surrogate discretization of the variable coefficient Poisson equation. Given a load $f \in L^2(\Omega)$ and symmetric positive-definite tensor $K : \Omega \to \mathbb{R}^{n\times n}$, the corresponding weak form may be written as follows.

$$\text{Find } u \in H^1_0(\Omega) \text{ satisfying} \quad a(u,v) = F(v) \quad \text{for all } v \in H^1_0(\Omega)\,, \tag{7.1}$$

where $a(u,v) = \int_\Omega \nabla u^\top K \nabla v \,\mathrm{d}x$ and $F(v) = \int_\Omega f v \,\mathrm{d}x$. As done in section 5, define $V_h = \{v \in H^1_0(\Omega) : v|_t \in \mathcal{P}_1(t) \text{ for each } t \in \mathcal{S}^m(\mathcal{T}_H)\}$, and let the corresponding vertex function basis be $\{\phi_i\}$.

7.1. Coercivity. In the present setting, observe that each $v_h \in V_h$ can be expressed as $v_h(x) = \sum_i v_h(x_i)\phi_i(x)$. Therefore, due to the zero row/column sum property (5.1), we find

$$\begin{aligned} \widetilde{a}(v_h,w_h) - a(v_h,w_h) &= \sum_{i,j} \big(\widetilde{\mathsf{A}}_{ij} - \mathsf{A}_{ij}\big) v_h(x_i) w_h(x_j) \\ &= \frac{1}{2} \sum_{i\neq j} \big(\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}\big)(v_h(x_i) - v_h(x_j))(w_h(x_i) - w_h(x_j))\,. \end{aligned} \tag{7.2}$$

Due to the mutual sparsity of the matrices $\widetilde{\mathsf{A}}$ and A, every nonzero term in the sum above can be rewritten as $\big(\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}\big)(v_h(x_i) - v_h(x_i+\delta))(w_h(x_i) - w_h(x_i+\delta))$ for some nonzero δ. Because $|\delta| \eqsim h$ by construction, one easily arrives at the following upper bound:

$$a(v_h,w_h) - \widetilde{a}(v_h,w_h) \lesssim h^{2-n} \big\|\mathsf{A} - \widetilde{\mathsf{A}}\big\|_{\max} \|\nabla v_h\|_0 \, \|\nabla w_h\|_0\,. \tag{7.3}$$

We now arrive at the main result of this subsection.

THEOREM 7.1. *Let $0 \le r \le q$. Assume that $a(\cdot,\cdot)$ is coercive and that $K \in \big[W^{r+1,\infty}(\mathcal{T}_H)\big]^{n\times n}$. Then, for any fine enough macromesh $\mathcal{T}_H$, the surrogate bilinear form $\widetilde{a} : V_h \times V_h \to \mathbb{R}$ is also coercive.*

Proof. Let $S = \{v \in H^1 : \|v\|_1 = 1\}$ be the surface of the unit ball in H^1. Recall that since $a(\cdot,\cdot)$ is coercive, there exists a coercivity constant $\alpha > 0$ such that $a(v,v) \ge \alpha$ for all $v \in S$. Notice that $\alpha \le a(v_h,v_h) \le \widetilde{a}(v_h,v_h) + |a(v_h,v_h) - \widetilde{a}(v_h,v_h)|$ for all $v_h \in V_h \cap S$ and, therefore,

$$\alpha - |a(v_h,v_h) - \widetilde{a}(v_h,v_h)| \le \widetilde{a}(v_h,v_h) \quad \text{for all } v_h \in V_h \cap S\,. \tag{7.4}$$

Here, the second term on the left may be bounded from above using (7.3) and Lemma 6.3 as follows:

$$|a(v_h,v_h) - \widetilde{a}(v_h,v_h)| \lesssim h^{2-n} \|\mathsf{A} - \widetilde{\mathsf{A}}\|_{\max} \|\nabla v_h\|_0^2 \lesssim H^{r+1} |K|_{W^{r+1,\infty}(\Omega)}\,. \tag{7.5}$$

Thus, for any small enough H, we see that $0 < \alpha - |a(v_h,v_h) - \widetilde{a}(v_h,v_h)| \le \widetilde{a}(v_h,v_h)$, as necessary. ∎

7.2. Convergence of the surrogate solution in the H^1 norm. The purpose of this subsection is to derive a mesh-dependent upper bound on the error in the surrogate solution $\widetilde{u}_h$ of the form $\|u - \widetilde{u}_h\|_1 \le C(K,\Omega,u)h + \widetilde{C}(K,\Omega,u)H^{r+1}$. In doing so, we choose to emphasize the primary difference from the classical $\|u - u_h\|_1 \le C(K,\Omega,u)h$ error estimate by absorbing the coercivity and continuity constants (which depend on both K and Ω) into the $\lesssim$ symbol. We begin with a particular version of the first Strang lemma [39].

LEMMA 7.2. *Let $S = \{v \in H^1 : \|v\|_1 = 1\}$ be the surface of the unit ball in H^1. Assume that $\widetilde{a} : V_h \times V_h \to \mathbb{R}$ is coercive. The following error estimate holds for the surrogate solution $\widetilde{u}_h$ of the variable coefficient model problem* (7.1)*:*

$$\|u - \widetilde{u}_h\|_1 \lesssim \inf_{w_h \in V_h} \left[\|u - w_h\|_1 + \sup_{v_h \in V_h \cap S} |\widetilde{a}(w_h, v_h) - a(w_h, v_h)| \right]. \tag{7.6}$$

THEOREM 7.3. *Let $0 \le r \le q$, and assume that $K \in \left[W^{r+1,\infty}(\Omega)\right]^{n\times n}$ is symmetric and positive definite with $\lambda_1(K)$ bounded away from zero almost everywhere. Let $u \in H^1(\Omega)$ and $\widetilde{u}_h \in V_h$ be the unique solutions to* (2.1a) *and* (2.1c)*, respectively, where $a(u,v) = \int_\Omega \nabla u^\top K \nabla v \,\mathrm{d}x$ and $F(v) = \int_\Omega f v \,\mathrm{d}x$. Then, for any sufficiently fine macromesh $\mathcal{T}_H$, the following upper bound holds:*

$$\|u - \widetilde{u}_h\|_1 \lesssim h|u|_2 + H^{r+1}|K|_{W^{r+1,\infty}(\Omega)}|u|_1 . \tag{7.7}$$

Proof. With the assumptions above, $a(\cdot,\cdot)$ is coercive. Therefore, by Theorem 7.1, if the macromesh $\mathcal{T}_H$ is taken fine enough, then $\widetilde{a} : V_h \times V_h \to \mathbb{R}$ is coercive. We now bound the right-hand side of (7.6). Invoking (7.3), we find

$$\|u - \widetilde{u}_h\|_1 \lesssim \|u - w_h\|_1 + h^{2-n}\big\|\mathsf{A} - \widetilde{\mathsf{A}}\big\|_{\max}\|\nabla w_h\|_0 \tag{7.8}$$

for every $w_h \in V_h$. Setting $w_h = \mathcal{S}\mathcal{Z}_h u$, the Scott–Zhang interpolant of u [38], we see that

$$\|u - \widetilde{u}_h\|_1 \lesssim \|u - \mathcal{S}\mathcal{Z}_h u\|_1 + h^{2-n}\big\|\mathsf{A} - \widetilde{\mathsf{A}}\big\|_{\max}|\mathcal{S}\mathcal{Z}_h u|_1 \lesssim h|u|_2 + h^{2-n}\big\|\mathsf{A} - \widetilde{\mathsf{A}}\big\|_{\max}|u|_1 . \tag{7.9}$$

In order to finish the proof, recall that $h^{2-n}\|\mathsf{A} - \widetilde{\mathsf{A}}\|_{\max} \lesssim H^{r+1}|K|_{W^{r+1,\infty}(\Omega)}$, by Lemma 6.3. ∎

7.3. Convergence of the surrogate solution in the L^2 norm. In this subsection, we prove an L^2 error estimate of the form $\|u - \widetilde{u}_h\|_0 \le C(K,\Omega,u)h^2 + \widetilde{C}(K,\Omega,u)H^{r+1}$. A second result, which elicits accelerated H-convergence, is also proved under the additional assumption $\sum_{x_i,\in T_m^\delta}\left[\Phi_T^\delta - \Pi_T^\delta \Phi_T^\delta\right](x_i) = 0$, where $T_m^\delta = T_m \cap \overline{T}_\delta$. This is a property which naturally arises for the specific class of least-squares projections introduced in subsection 8.1. Again, we emphasize the primary differences from the corresponding classical error estimate by absorbing the standard constants into the $\lesssim$ symbol.

THEOREM 7.4. *Under the conditions of Theorem* 7.3*, if $\Omega \subseteq \mathbb{R}^n$ is a convex domain, then the following additional upper bound on the error in the surrogate solution holds:*

$$\|u - \widetilde{u}_h\|_0 \lesssim h^2|u|_2 + H^{r+1}|K|_{W^{r+1,\infty}(\Omega)}|u|_1 . \tag{7.10a}$$

Moreover, if $r > 0$ and $\sum_{x_i \in T_m^\delta}\left[\Phi_T^\delta - \Pi_T^\delta \Phi_T^\delta\right](x_i) = 0$, for each $T \in \mathcal{T}_H$ and $\delta \in \mathcal{D}(\mathring{T}_m)$, then

$$\|u - \widetilde{u}_h\|_0 \lesssim h^2|u|_2 + H^{r+2}|K|_{W^{r+1,\infty}(\Omega)}\|\nabla u\|_1 . \tag{7.10b}$$

Proof. By the triangle inequality, $\|u - \widetilde{u}_h\|_0 \le \|u - u_h\|_0 + \|u_h - \widetilde{u}_h\|_0$, where $u_h \in V_h$ is the discrete solution coming from (2.1b). It can be shown that if Ω is convex, then $u \in H^2(\Omega) \cap H_0^1(\Omega)$; see, e.g., [29]. It then follows from standard

arguments that $\|u-u_h\|_0 \lesssim h^2|u|_2$; see, e.g., [16, Theorem 5.7.6]. Therefore, we only need to analyze the term $\|u_h-\widetilde{u}_h\|_0$. Since, $\|u_h-\widetilde{u}_h\|_0 \le \|u_h-\widetilde{u}_h\|_1$, proceeding as in the proof of Theorem 7.3, we quickly arrive at (7.10a).

In order to prove (7.10b), first define $w_h \in V_h$ satisfying $a(w_h,v_h) = (u_h-\widetilde{u}_h, v_h)_\Omega$ for all $v_h \in V_h$. Observe that the exact solution of the problem $a(w,v) = (u_h-\widetilde{u}_h,v)_\Omega$ for all $v \in H_0^1(\Omega)$ belongs to the space $H^2(\Omega)\cap H_0^1(\Omega)$, $\|w\|_2 \lesssim \|u_h-\widetilde{u}_h\|_0$. Moreover,

$$\begin{aligned}\|u_h-\widetilde{u}_h\|_0^2 &= a(w_h,u_h-\widetilde{u}_h) = \widetilde{a}(w_h,\widetilde{u}_h) - a(w_h,\widetilde{u}_h)\\ &= \frac{1}{2}\sum_{i\neq j}(\mathsf{A}_{ij}-\widetilde{\mathsf{A}}_{ij})(\widetilde{u}_h(x_i)-\widetilde{u}_h(x_j))(w_h(x_i)-w_h(x_j))\,,\end{aligned} \tag{7.11}$$

where the final line follows from (7.2). As remarked previously, each nonzero term in this sum can be written as $\big(\mathsf{A}_{ij}-\widetilde{\mathsf{A}}_{ij}\big)(\widetilde{u}_h(x_i)-\widetilde{u}_h(x_i+\delta))(w_h(x_i)-w_h(x_i+\delta))$ for some $T\in\mathcal{T}_H$ and nonzero $\delta\in\mathcal{D}(T_m)$. We can make better use of this expression with the identity $v_h(x_i)-v_h(x_i+\delta) = -\nabla v_h(y_{i,\delta})\cdot\delta$, wherein each $y_{i,\delta}$ is chosen from the edge connecting x_i and $x_i+\delta$, and with the relationship $\mathsf{A}_{ij}-\widetilde{\mathsf{A}}_{ij} = \big[\Phi_T^\delta - \Pi_T^\delta\Phi_T^\delta\big](x_i)$, since $\mathsf{A}_{ij}-\widetilde{\mathsf{A}}_{ij}\neq 0$ and $i\neq j$. With these observations in hand, we have

$$2\|u_h-\widetilde{u}_h\|_0^2 = \sum_{T\in\mathcal{T}_H}\sum_{\delta\in\mathcal{D}(\mathring{T}_m)}\sum_{x_i\in T_m^\delta}\big[\Phi_T^\delta-\Pi_T^\delta\Phi_T^\delta\big](x_i)\,(\nabla\widetilde{u}_h(y_{i,\delta})\cdot\delta)(\nabla w_h(y_{i,\delta})\cdot\delta) \tag{7.12}$$

$$\le \sum_{T\in\mathcal{T}_H}\sum_{\delta\in\mathcal{D}(\mathring{T}_m)}\sum_{x_i\in T_m^\delta}\big[\Phi_T^\delta-\Pi_T^\delta\Phi_T^\delta\big](x_i)\Big((\nabla\widetilde{u}_h(y_{i,\delta})\cdot\delta)(\nabla w_h(y_{i,\delta})\cdot\delta) - C\Big) \tag{7.13}$$

$$\lesssim h^{-n}\|\mathsf{A}-\widetilde{\mathsf{A}}\|_{\max}\sum_{T\in\mathcal{T}_H}\sum_{\delta\in\mathcal{D}(\mathring{T}_m)}\|(\nabla\widetilde{u}_h\cdot\delta)(\nabla w_h\cdot\delta)-C\|_{L^1(T)}\,. \tag{7.14}$$

If we set the constant above to the following average value, $C := \frac{1}{\mathrm{vol}(T)}\int_T(\nabla u\cdot\delta)(\nabla w\cdot\delta)\,\mathrm{d}x$, then $\|(\nabla u\cdot\delta)(\nabla w\cdot\delta)-C\|_{L^1(T)} \lesssim H_T|(\nabla u\cdot\delta)(\nabla w\cdot\delta)|_{W^{1,1}(T)}$. Therefore,

$$\|(\nabla\widetilde{u}_h\cdot\delta)(\nabla w_h\cdot\delta)-C\|_{L^1(T)} \le \|(\nabla\widetilde{u}_h\cdot\delta)(\nabla w_h\cdot\delta)-(\nabla u\cdot\delta)(\nabla w_h\cdot\delta)\|_{L^1(T)} \tag{7.15}$$

$$+\|(\nabla u\cdot\delta)(\nabla w_h\cdot\delta)-(\nabla u\cdot\delta)(\nabla w\cdot\delta)\|_{L^1(T)}+\|(\nabla u\cdot\delta)(\nabla w\cdot\delta)-C\|_{L^1(T)} \tag{7.16}$$

$$\lesssim \|\nabla(u-\widetilde{u}_h)\cdot\delta\|_{0,T}\,\|\nabla w_h\cdot\delta\|_{0,T} + \|\nabla(w-w_h)\cdot\delta\|_{0,T}\,\|\nabla u\cdot\delta\|_{0,T} \tag{7.17}$$

$$+ H_T\|\nabla u\cdot\delta\|_{1,T}\|\nabla w\cdot\delta\|_{1,T}\,. \tag{7.18}$$

After summing over all $T\in\mathcal{T}_H$ and $\delta\in\mathcal{D}(T_m)$ and taking into account $|\delta|\eqsim h$, we arrive at the bound

$$\|u_h-\widetilde{u}_h\|_0^2 \lesssim h^{2-n}\|\mathsf{A}-\widetilde{\mathsf{A}}\|_{\max}\big(\|u-\widetilde{u}_h\|_1\,|w_h|_1 + \|w-w_h\|_1\,|u|_1 + H\|\nabla u\|_1\|\nabla w\|_1\big) \tag{7.19}$$

$$\lesssim h^{2-n}\|\mathsf{A}-\widetilde{\mathsf{A}}\|_{\max}\big(h\,|u|_2 + H^{r+1}|K|_{W^{r+1,\infty}(\Omega)}|u|_1 + H\,\|\nabla u\|_1\big)\|w\|_2 \tag{7.20}$$

$$\lesssim H^{r+1}|K|_{W^{r+1,\infty}(\Omega)}\big(H\,\|\nabla u\|_1 + H^{r+1}|K|_{W^{r+1,\infty}(\Omega)}|u|_1\big)\|w\|_2\,. \tag{7.21}$$

The proof is completed by recalling that $\|w\|_2 \lesssim \|u_h-\widetilde{u}_h\|_0$. ∎

Remark 7.1. As stated previously, the error estimates in Theorems 7.3 and 7.4 not do track the constants appearing in the corresponding classical estimates. When $r > 0$, none of the constants in either classical estimate depend on the higher-order norms $|K|_{W^{r+1,\infty}(\Omega)}$.

Remark 7.2. As mentioned in Remark 3.2, surrogate matrices may be constructed for high-order finite element or isogeometric methods. Likewise, Theorems 7.3 and 7.4 may be restated for some high-order surrogate methods; cf. [22, Theorem 7.4]. Many of the details are given in sufficient detail in [22], so we only briefly summarize them here. The upshot is as follows: so long as (6.4b) and (7.3) still hold, Theorems 7.3 and 7.4 can be generalized for any trial space $V_h \subseteq H^1(\Omega)$ with polynomial degree $k \geq 1$.

Note that the proof of (6.4b) reduces to analyzing the maximum L^∞-distance between a stencil function and its surrogate, $\|\Phi_T^\delta - \Pi_T^\delta \Phi_T^\delta\|_{L^\infty(T)}$, for some macroelement T. Up to a scale-independent constant, this quantity does not depend on k, and so neither do the exponents of h or H in (6.4b); cf. [22, Theorem 4.2]. Moreover, making use of (7.2), one may show that (7.3) actually holds for any nodal basis. An analogous bound may be proven to hold for NURBS bases; see [22, Lemma 7.1]. Ultimately, because each H-dependent (i.e., consistency error) term in (7.7) and (7.10) is controlled using (6.4b) and (7.3), the H-dependent scaling in (7.7) and (7.10) remains unchanged if $k > 1$. In fact, only the h-dependent scaling terms in (7.7) and (7.10) are influenced by k. These terms behave exactly like the standard discretization errors $\|u - u_h\|_\bullet$, $\bullet = 0, 1$, and high-order bounds on on such quantities are well-known in the literature. For example, given a sufficiently smooth solution u, (7.7) changes to

$$\|u - \widetilde{u}_h\|_1 \lesssim h^k |u|_{k+1} + H^{r+1} |K|_{W^{r+1,\infty}(\Omega)} |u|_1 \,. \tag{7.22}$$

k-dependent restatements of inequalities (7.10) are obvious and left for the reader.

Remark 7.3. The main ingredients in the error analysis presented in this section are Lemma 6.3, Strang's first lemma, and (7.2), which simply follows from the zero row sum property and symmetry. Simple generalizations of Lemma 6.3, for the bilinear forms $a_2(\cdot,\cdot)$ and $a_3(\cdot,\cdot)$ have been stated in (6.12). Meanwhile, (7.2) simply follows from the zero row sum property of $\widetilde{\mathsf{A}}$ and symmetry. Therefore, we see no reason to doubt that our analysis here can be generalized to the other problems of interest in this paper. Hence, we proceed with numerical verification and demonstration.

8. Implementation. The performance of a numerical method largely depends on its implementation. Therefore, in this section, we highlight the important features of ours. We use the HyTeG finite element software framework [31] as the core framework for all the numerical experiments in section 9. It offers efficient distributed data structures for simplicial meshes in two and three dimensions, which serve as a basis for the implementation of massively parallel fast iterative solvers. Its main concept is based on the idea that a coarse input mesh is split into its geometrical primitives, i.e., vertices, edges, and faces, and each of these primitives is uniformly refined. Because the primitives of the same dimension are decoupled from the others, all primitives of the same dimension may be processed in parallel. This partitioning and the hierarchy of locally structured meshes allows for efficient parallel implementations of geometric multigrid methods. More importantly, these data structures fit perfectly to the concept of macroelements introduced in subsection 3.3. The problems in this paper are mainly solved by employing a geometric multigrid solver using V-cycles with a hybrid Gauss–Seidel smoother. On the coarsest grid, either a preconditioned

conjugate gradient method or the multifrontal massively parallel sparse direct solver (MUMPS) [1, 2], as provided by the PETSc interface [4, 5], is used. For improved parallel scalability of the coarse grid solver, agglomeration techniques as provided by `PCTELESCOPE` [37] are used in runs with many processes.

8.1. Polynomial least squares regression. An important factor in the performance of the surrogate approach is the approximation of the stencil functions Φ_T^δ by polynomials $\widetilde{\Phi}_T^\delta$. This step in the solver process must be very fast and is usually done in a preprocess step before the actual solve. After various preparatory experiments, we have seen satisfactory performance and accuracy from simply computing $\widetilde{\Phi}_T^\delta = \Pi_T^\delta \Phi_T^\delta$ via solving a simple least-squares problem, which we now describe.

Let $T \in \mathcal{T}_H$ be a macroelement, and recall that T_m is the associated lattice on level m. Suppose that Φ_T^δ is the stencil function in direction $\delta \in \mathcal{D}(T_m)$ which we want to approximate. For the least-squares regression, we fix a level m_{LS} with $m \geq m_{\mathrm{LS}} \geq 2$ and define the set of least-squares points $T_{\mathrm{LS}}^\delta \coloneqq T_{m_{\mathrm{LS}}} \cap \overline{T}_\delta$. Furthermore, let $\{p_k\}_{k=1}^M$ be a basis of $\mathcal{P}_q(T)$, the space of polynomials with maximal degree q. Assume that m_{LS} is chosen large enough such that $\left|T_{\mathrm{LS}}^\delta\right| \geq M$, and introduce the following norm on $\mathcal{P}_q(T)$: $\|p\|_{T_{\mathrm{LS}}^\delta}^2 \coloneqq \sum_{x_i \in T_{\mathrm{LS}}^\delta} p(x_i)^2$. The least-squares regression problem, which in turn defines Π_T^δ, is formalized as follows:

$$\text{Find } \mathsf{c} \in \mathbb{R}^M \text{ satisfying} \quad \mathsf{c} = \operatorname*{arg\,min}_{\mathsf{d} \in \mathbb{R}^M} \left\| \Phi_T^\delta - \sum_{k=1}^M d_k p_k \right\|_{T_{\mathrm{LS}}^\delta}^2 . \tag{8.1}$$

The approximated stencil function is then defined as $\widetilde{\Phi}_T^\delta \coloneqq \sum_{k=1}^M c_k p_k$. This problem is equivalent to solving the possibly overdetermined linear system of equations $\mathsf{B}\mathsf{c} = \mathsf{f}$ in a least-squares sense, where $\mathsf{B}_{ij} = p_j(x_i)$ and $\mathsf{f}_i = \Phi_T^\delta(x_i)$ for $1 \leq i \leq |T_{\mathrm{LS}}^\delta|$ and $1 \leq j \leq N$. The choice of the polynomial basis is arbitrary. However, for an easier implementation, we employ the monomial basis, even knowing that the resulting linear system is ill conditioned. Since it is crucial to get numerically precise results, a stable solver for this problem has to be chosen. For this purpose, we apply the `colPivHouseholderQr` method from the Eigen 3.3.5 library [30], which offers a good balance between speed and accuracy. Obviously, each of these linear systems is independent of others; therefore, they may be solved in parallel.

Remark 8.1. Taking into account $\widetilde{\Phi}_T^\delta = \sum_{k=1}^M c_k p_k$, the first-order optimality condition for (8.1) can be stated as $\sum_{x_i \in T_{\mathrm{LS}}^\delta} \left[\Phi_T^\delta - \Pi_T^\delta \Phi_T^\delta\right](x_i) = 0$. If $m = m_{\mathrm{LS}}$, then the secondary assumption in Theorem 7.4 is satisfied and we see higher-order convergence in H, as stated in (7.10b). Usually, when m_{LS} is close but not equal to m, we see preasymptotic H-convergence in between the two estimates given in (7.10); see Figure 6.

Remark 8.2. In the case where the bilinear form $a(\cdot,\cdot)$ is symmetric, we need only approximate a single stencil function Φ_T^δ for both directions δ and $-\delta$. Indeed, as observed in Remark 3.1, the corresponding stencil functions are identical, up to a shift by δ. Furthermore, the symmetry requirement (3.18), from Remark 3.4, is satisfied with the projection operator defined above. Indeed, one may verify that for every δ, $T_\delta = T_{-\delta} - \delta$. Therefore,

$$\left\| \Phi_T^\delta - \widetilde{\Phi}_T^\delta \right\|_{T_{\mathrm{LS}}^\delta}^2 = \left\| \Phi_T^{-\delta} - \widetilde{\Phi}_T^{-\delta} \right\|_{T_{\mathrm{LS}}^{-\delta}}^2 \qquad \text{and} \qquad \widetilde{\Phi}_T^\delta(x) = \widetilde{\Phi}_T^{-\delta}(x+\delta)\,. \tag{8.2}$$

Thus, on simplicial meshes in two dimensions, only four instead of seven polynomials per macroelement have to be determined and stored in memory. In some cases, where the zero row sum property holds, the number of required polynomials may be even reduced to three.

8.2. Fast polynomial evaluation. An even more important factor with respect to the performance of our implementation is the fast evaluation of the surrogate stencil functions $\widetilde{\Phi}_T^\delta$. Contrary to the computation of each $\widetilde{\Phi}_T^\delta$, which will happen only once per solve, these evaluations will be made during every matrix-vector multiplication. Therefore, the costs of evaluating the stiffness matrix entries associated to a degree of freedom may not exceed the costs of evaluating the bilinear forms with the respective ansatz functions. In this case not only the reduction of FLOPS per degree of freedom is of importance, but also the required memory traffic has to be taken into account.

When performing a matrix-vector multiplication in HyTeG, the degrees of freedom in a macroelement are processed in a row-wise fashion as illustrated in the left of Figure 4. In each row, the stencil function may be interpreted as a one-dimensional (1D) function. We assume without loss of generality, that the 1D stencil functions are aligned with the x-axis. This property is also inherited by the approximated stencil function.

To further optimize the evaluation of the 1D polynomial, we exploit that the stencil functions have to be evaluated on a line subdivided into uniformly sized intervals of length h. Let (x_i, y_j) be a vertex node in the lattice, and let $p_{y_j}(\cdot) \coloneqq \widetilde{\Phi}_\delta(\cdot, y_j)$ be the approximated 1D stencil function associated to row y_j.

Assuming that we already have evaluated the stencil function p_{y_j} at a point x_i, we want to evaluate it at the next point $x_i + h$ as efficiently as possible. Since the grid points are equidistantly distributed, we can use a special case of the divided differences, called forward differences [18, page 126].

First, we need $q+1$ initial helper variables $\{\Delta_{x_0}^{(k,\ell)}\}$ with $\ell = 0$ and $k \in \{0, 1, \ldots, q\}$ which are defined in a preprocessing step as follows:

$$\Delta_{x_0}^{(0,0)} \coloneqq p_{y_j}(x_0)\,, \tag{8.3}$$

$$\Delta_{x_0}^{(k,0)} \coloneqq \Delta_{x_0+h}^{(k-1,0)} - \Delta_{x_0}^{(k-1,0)}, \quad k \in \{1, \ldots, q\}\,. \tag{8.4}$$

The values of $\Delta_{x_0+h}^{(k-1,0)}$ are defined in a similar fashion, and the value at position $p_{y_j}(x_0)$ is given by $\Delta_{x_0}^{(0,0)}$. In order to obtain the value at $p_{y_j}(x_0 + h)$ and to proceed

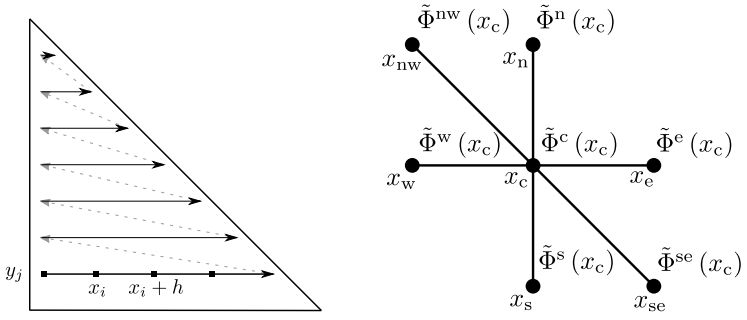


FIG. 4. *Illustration of a loop through the degrees of freedom in a macroelement. In each row of the loop, the two-dimensional stencil function may be interpreted as a 1D function (left). Seven stencil functions have to be evaluated in order to obtain the whole stencil for a degree of freedom (right).*

further in line, i.e., $\ell \to \ell+1$, one has to update all the helper variables in the following way:

$$\Delta_{x_0}^{(k,\ell)} := \Delta_{x_0}^{(k,\ell-1)} + \Delta_{x_0}^{(k+1,\ell-1)}, \quad k \in \{0,1,\dots,q\}, \ell > 0\,. \tag{8.5}$$

Since the second summand may not be defined for all cases, we set $\Delta_{x_0}^{(q,\ell)} = \Delta_{x_0}^{(q,0)}$ or equivalently $\Delta_{x_0}^{(k,\ell)} = 0$ for all $k > q$. After that, the value of $p_{y_j}(x_0 + h)$ is given by $\Delta_{x_0}^{(0,1)}$. Doing this recursively yields the approximated stencil function values at all mesh points on a single row using only $q+1$ helper variables and $q+1$ floating point additions. When iterating through a row, in general, seven polynomial evaluations, one for each direction, are required; cf. right of Figure 4. In the symmetric case this may be reduced to six polynomial evaluations, since the western stencil weight may be obtained from the previous eastern evaluation. Keep also in mind that in the symmetric case the polynomials of approximated stencil functions in opposite directions are the same but only evaluated at different positions; cf. Remark 8.2. Therefore, $6 \cdot (q+1)$ helper variables are required for a single row. For $q = 8$, these results in $54 \cdot 8$ bytes of memory which fits easily into a modern L1 CPU cache. When moving from one lattice point to another, $6 \cdot (q+1)$ vectorizable floating point additions have to be performed to obtain the updated polynomial evaluations. Furthermore, in our implementation, the polynomial degree is realized as a C++ template parameter; therefore, all loops concerning the evaluation of a polynomial of a certain degree may be optimized at compile time. Since our focus lies in the theoretical analysis of the surrogate approach, thorough performance studies employing performance models should be considered beyond the scope of this paper. Similar performance studies have been carefully completed in [9, 10]. Thus, in the next section, we only report on relative run times of the surrogate approach compared to the standard method, also implemented on HyTeG, using on-the-fly quadrature of the integrals stemming from the bilinear forms.

9. Numerical experiments. In this section, we numerically verify Theorems 7.3 and 7.4, both related to the variable coefficient Poisson equation. Additionally, we present proof-of-concept results for a linearized elasticity application and a simple p-Laplacian diffusion problem. While not covered by the theory, we include these latter examples to demonstrate the breadth of generality of the methodology.

All run time measurements in this sections were obtained on a machine equipped with two Intel® Xeon® Gold 6136 processors with a nominal base frequency of 3.0 GHz. Each processor has 12 physical cores which results in a total of 24 physical cores. The total available memory of 251 GB is split into two nonuniform memory access domains, one for each socket. We use version 7.3.0 of the GNU compiler collection and specify the following compiler arguments: `-O3 -march=native`. All the examples in this section were executed in parallel using all available 24 physical cores.

When comparing run times from the standard and the surrogate approaches, many factors are responsible for the relative speed-up of the surrogate approach. Increasing the polynomial order q of the surrogate stencil functions increases not only the run time of a multigrid iteration but also the time spent in the setup phase (i.e., computing each Φ_T^δ). The cost of the setup phase, however, is mostly dominated by the sampling level m_{LS}. Therefore, when the ratio of time spent in the iterative solver to the time spent in the setup phase(s) for solving a particular problem is large, the setup cost is almost negligible and we see the best performance. Since the problems in the following subsections differ in complexity and have different ratios

of solver to setup time, the observed relative speed-ups are not directly comparable. Nonetheless, the reported speed-ups for all tested examples range between factors of 14 and 20. Such significant speed-ups are in particular important in case of dynamic or stochastic applications. Most stochastic applications demand an enormous number of deterministic solves resulting quite often in extreme long run times. Having such a surrogate approach at hand can help to make stochastic approaches such as, e.g., multilevel Monte Carlo and its variants, more accessible for complex applications.

9.1. Quantitative benchmark problem. In this subsection, we examine the surrogate method for the variable coefficient Poisson equation which has been described and analyzed above. The strong form of the problem is

$$\begin{aligned} -\operatorname{div}(K\nabla u) &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega. \end{aligned} \tag{9.1}$$

We consider both the bilinear form coming from the scalar coefficient scenario (i.e., $K = k \cdot \mathrm{Id}$), introduced in (3.2), and the tensorial coefficient scenario, introduced in (3.13). In the scalar coefficient experiments, we use the unit-square domain $\Omega = (0,1)^2$. In the tensorial coefficient experiments, the domain Ω has a curvilinear boundary.

9.1.1. Scalar coefficient on unit square. In the first benchmark problem ($K = k \cdot \mathrm{Id}$), we take $\Omega = (0,1)^2$ and employ the scalar coefficient function

$$k(x,y) = \exp(xy) + \sin(3\pi xy) + \cos\left(\pi x^2 y\right) + 1 \tag{9.2}$$

in problem (9.1). The manufactured solution u is chosen as $u(x,y) = \sin(x)\sinh(y)$. The restriction of u to the boundary is chosen as Dirichlet datum g. The right-hand-side f is directly computed by inserting u into the equation. In this benchmark, we fix the finest mesh size h and report on the errors depending on H and q to show the proven $\mathcal{O}\left(H^{q+1}\right)$ estimate in the H^1 norm and $\mathcal{O}\left(H^{q+2}\right)$ in the L^2 norm. For this purpose, h is chosen to be very small in order for the error to be mostly dominated by the surrogate part.

The reference macromesh size is given by H_0, as illustrated in the left of Figure 5. All finer macromeshes, with associated mesh sizes $H < H_0$, stem from uniformly refining this reference mesh; see middle and right of Figure 5. The fine mesh, with associated mesh size $h \ll H$, is the 13 times uniformly refined reference macromesh, i.e., $h = 2^{-13}H_0$. This fine mesh has about $6.71 \cdot 10^7$ degrees of freedom. The approximation of the stencil functions through least-squares regression is done on

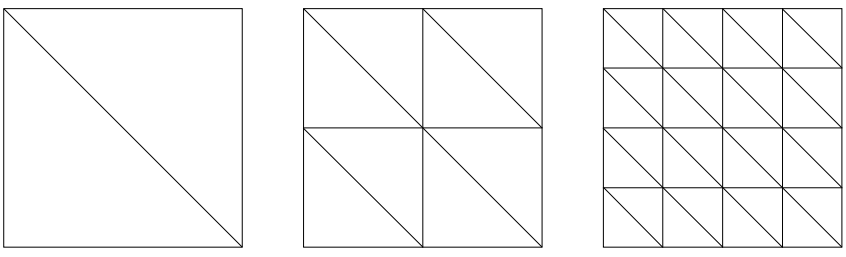


FIG. 5. *Coarse macromeshes of the unitsquare with mesh sizes $H = H_0$ (left), $H = H_0/2$ (middle), and $H = H_0/4$ (right). Meshes with a smaller H follow the same uniform refinement pattern.*

TABLE 1

Relative H^1 errors (rel. H^1 err.) and experimental orders of convergence (EOC) for fixed h and varying q and H in the case of problem (9.1) with the scalar coefficient (9.2). Here, the relative H^1 error with the classical Finite element method (FEM) is $1.23 \cdot 10^{-8}$.

$\frac{H}{H_0}$	$q=1$ Rel. H^1 err.	EOC	$q=2$ Rel. H^1 err.	EOC	$q=3$ Rel. H^1 err.	EOC	$q=4$ Rel. H^1 err.	EOC
2^{-1}	$4.58 \cdot 10^{-2}$	–	$2.24 \cdot 10^{-2}$	–	$4.52 \cdot 10^{-3}$	–	$1.68 \cdot 10^{-3}$	–
2^{-2}	$1.61 \cdot 10^{-2}$	1.50	$2.75 \cdot 10^{-3}$	3.02	$4.46 \cdot 10^{-4}$	3.34	$4.70 \cdot 10^{-5}$	5.16
2^{-3}	$4.43 \cdot 10^{-3}$	1.86	$3.74 \cdot 10^{-4}$	2.88	$2.88 \cdot 10^{-5}$	3.95	$1.59 \cdot 10^{-6}$	4.89
2^{-4}	$1.15 \cdot 10^{-3}$	1.95	$4.86 \cdot 10^{-5}$	2.94	$1.84 \cdot 10^{-6}$	3.97	$5.22 \cdot 10^{-8}$	4.93
2^{-5}	$2.90 \cdot 10^{-4}$	1.98	$6.17 \cdot 10^{-6}$	2.98	$1.17 \cdot 10^{-7}$	3.98	$1.23 \cdot 10^{-8}$	2.09

TABLE 2

Relative L^2 errors (rel. L^2 err.) and experimental orders of convergence for fixed h and varying q and H in the case of problem (9.1) with the scalar coefficient (9.2). Here, the relative L^2 error with the classical FEM is $4.10 \cdot 10^{-9}$.

$\frac{H}{H_0}$	$q=1$ Rel. L^2 err.	EOC	$q=2$ Rel. L^2 err.	EOC	$q=3$ Rel. L^2 err.	EOC	$q=4$ Rel. L^2 err.	EOC
2^{-1}	$5.75 \cdot 10^{-3}$	–	$1.92 \cdot 10^{-3}$	–	$2.90 \cdot 10^{-4}$	–	$9.46 \cdot 10^{-5}$	–
2^{-2}	$1.08 \cdot 10^{-3}$	2.42	$1.26 \cdot 10^{-4}$	3.93	$1.62 \cdot 10^{-5}$	4.16	$1.47 \cdot 10^{-6}$	6.01
2^{-3}	$1.60 \cdot 10^{-4}$	2.75	$8.84 \cdot 10^{-6}$	3.83	$5.63 \cdot 10^{-7}$	4.85	$2.47 \cdot 10^{-8}$	5.90
2^{-4}	$2.16 \cdot 10^{-5}$	2.89	$5.96 \cdot 10^{-7}$	3.89	$1.88 \cdot 10^{-8}$	4.91	$4.19 \cdot 10^{-9}$	2.56
2^{-5}	$2.80 \cdot 10^{-6}$	2.95	$3.87 \cdot 10^{-8}$	3.94	$4.10 \cdot 10^{-9}$	2.19	$4.05 \cdot 10^{-9}$	0.05

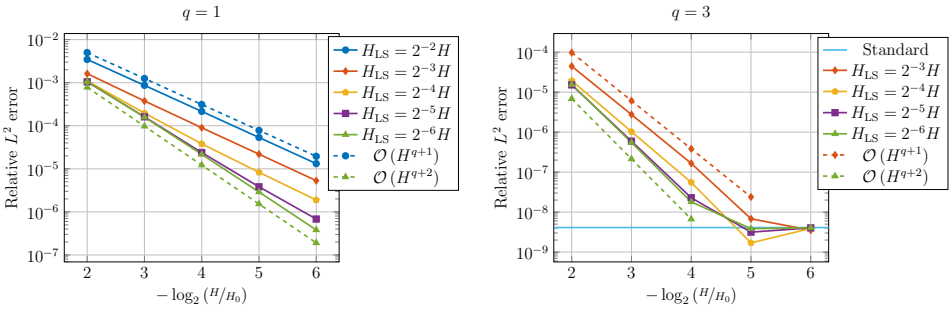


FIG. 6. *Relative L^2 errors for fixed $h = 2^{-13}H_0$, varying H_{LS}, $q = 1$ (left), and $q = 3$ (right) in the case of the variable coefficient Poisson equation on the unit-square with a scalar coefficient. For $q = 3$ the relative L^2 error obtained from the standard approach is included, since the discretization error is dominating the surrogate error on the meshes with $H \leq 2^{-5}H_0$.*

the mesh associated to mesh size $H_{\mathrm{LS}} = 2^{-8}H$. Note that this keeps the number of sampling points in each macroelement constant to 32 639. Each linear system is solved by applying geometric multigrid V(2,2) iterations until a relative residual of $1 \cdot 10^{-13}$ is obtained.

In Tables 1 and 2, the relative H^1 and L^2 errors for decreasing mesh sizes H are shown. Both tables show the expected convergence rates. In the case of the L^2 norm for $q = 3$ and $q = 4$, the convergence rate deteriorates for small macromesh sizes H because the discretization error is dominating the total error.

In order to show the dependence of the least-squares approach on the sampling level, we provide Figure 6. Here, we show two plots of the relative L^2 errors for fixed $q \in \{1, 3\}$, $h = 2^{-13}H_0$, and varying H_{LS}. From this figure, one can see that it is

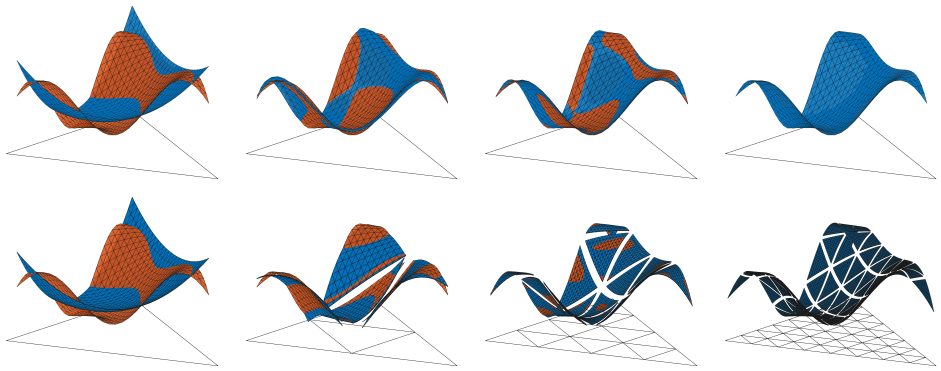


FIG. 7. *Plots of true stencil functions in orange and surrogate stencil functions in blue for* $\delta = 0$ *over the subdomain* $\{(x,y)^\top \in \Omega : x + y \geq 1\}$ *in the case of the variable coefficient Poisson equation. Top row: Fixed* $H = H_0$ *and varying* $q = 2$, 4, 6, *and* 8 *from left to right. Bottom row: Fixed* $q = 2$ *and varying* $H = H_0$, $H_0/2$, $H_0/4$, *and* $H_0/8$ *from left to right.*

crucial to tune the sampling level fine enough in order to achieve optimal $\mathcal{O}\left(H^{q+2}\right)$ convergence in the L^2-norm.

Remark 9.1. The choice of sampling level m_{LS} or, equivalently, H_{LS} is very important since the cost of the polynomial regression grows exponentially with m_{LS}. However, choosing a value for H_{LS} which is too large may violate the discrete L^2 projection property required in Theorem 7.4 in order to obtain an increased order of convergence. Therefore, it is crucial to choose a suitable H_{LS} for an optimal ratio between the accuracy of the solution and the run time of the stencil function approximation. In each of our experiments, we let $H_{\mathrm{LS}} = M_{\mathrm{LS}} \cdot h$, where $M_{\mathrm{LS}} \in \{2, 4\}$. This yielded satisfactory results with respect to accuracy and run time.

Furthermore, in Figure 7 we want to illustrate the dependence of the polynomial degree q and the macromesh size H within the surrogate approach. For this purpose, we plot the central true and surrogate stencil functions over the subdomain $\{(x,y)^\top \in \Omega : x + y \geq 1\}$ for different pairings of q and H. It can be observed that there is no visible difference of both functions when either the pairing $H = H_0$ and $q = 8$ or the pairing $H = H_0/8$ and $q = 2$ is chosen. Obviously, the quality of A can be improved by either increasing q or decreasing H. For smooth coefficients K, increasing q is the more efficient option, as in the *hp*-FEM context.

9.1.2. Tensor coefficient on domain with curved boundaries. In the second benchmark problem, we study problem (9.1) with the symmetric and positive definite tensor coefficient

$$K(x,y) = \begin{bmatrix} 3x^2 + 2y^2 + 1 & -x^2 - y^2 \\ -x^2 - y^2 & 4x^2 + 5y^2 + 1 \end{bmatrix}. \tag{9.3}$$

Moreover, we consider the domain Ω with the curved boundary illustrated in Figure 8. In the following scenarios, $a = 0.1$ is used as the amplitude of the boundary perturbation. The mapping from the reference unit-square to the perturbed domain is defined by φ in (9.4). To map the coefficient onto the perturbed domain, we replace the coefficient K in (9.1) by a new coefficient, K_0, induced by the domain transformation, viz.,

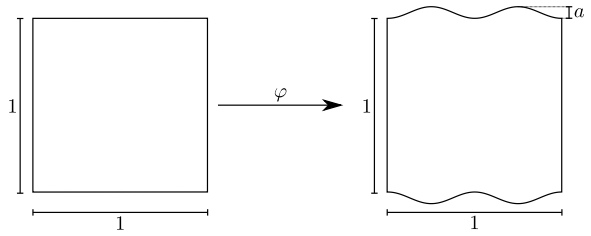


FIG. 8. *Illustration of the mapping φ from the unit-square to the perturbed unit-square. The top boundary is parametrized by $y = a \cdot \sin(2\pi x)^2 + 1$ and the bottom boundary by $y = -a \cdot \sin(2\pi x)^2$.*

TABLE 3

Relative H^1 errors and experimental orders of convergence for fixed h and varying q and H in the case of problem (9.1) *with the tensorial coefficient* (9.3) *and curved boundary. The relative H^1 error with the classical FEM is $1.59 \cdot 10^{-8}$.*

$\frac{H}{H_0}$	$q=1$ Rel. H^1 err.	EOC	$q=2$ Rel. H^1 err.	EOC	$q=3$ Rel. H^1 err.	EOC	$q=4$ Rel. H^1 err.	EOC
2^{-1}	$1.04 \cdot 10^{-1}$	–	$7.09 \cdot 10^{-2}$	–	$2.96 \cdot 10^{-2}$	–	$9.31 \cdot 10^{-3}$	–
2^{-2}	$5.41 \cdot 10^{-2}$	0.95	$1.12 \cdot 10^{-2}$	2.67	$4.31 \cdot 10^{-3}$	2.78	$1.07 \cdot 10^{-3}$	3.12
2^{-3}	$1.37 \cdot 10^{-2}$	1.98	$2.29 \cdot 10^{-3}$	2.29	$3.14 \cdot 10^{-4}$	3.78	$4.66 \cdot 10^{-5}$	4.52
2^{-4}	$3.72 \cdot 10^{-3}$	1.89	$2.94 \cdot 10^{-4}$	2.96	$2.25 \cdot 10^{-5}$	3.80	$1.64 \cdot 10^{-6}$	4.83
2^{-5}	$9.56 \cdot 10^{-4}$	1.96	$3.77 \cdot 10^{-5}$	2.96	$1.46 \cdot 10^{-6}$	3.94	$5.55 \cdot 10^{-8}$	4.89

TABLE 4

Relative L^2 errors and experimental orders of convergence for fixed h and varying q and H in the case of problem (9.1) *with the tensorial coefficient* (9.3) *and curved boundary. The relative L^2 error with the classical FEM is $4.56 \cdot 10^{-9}$.*

$\frac{H}{H_0}$	$q=1$ Rel. L^2 err.	EOC	$q=2$ Rel. L^2 err.	EOC	$q=3$ Rel. L^2 err.	EOC	$q=4$ Rel. L^2 err.	EOC
2^{-1}	$1.44 \cdot 10^{-2}$	–	$6.43 \cdot 10^{-3}$	–	$3.08 \cdot 10^{-3}$	–	$5.41 \cdot 10^{-4}$	–
2^{-2}	$3.74 \cdot 10^{-3}$	1.95	$6.60 \cdot 10^{-4}$	3.28	$1.39 \cdot 10^{-4}$	4.47	$3.54 \cdot 10^{-5}$	3.94
2^{-3}	$5.37 \cdot 10^{-4}$	2.80	$5.75 \cdot 10^{-5}$	3.52	$6.46 \cdot 10^{-6}$	4.43	$6.13 \cdot 10^{-7}$	5.85
2^{-4}	$7.35 \cdot 10^{-5}$	2.87	$3.77 \cdot 10^{-6}$	3.93	$2.09 \cdot 10^{-7}$	4.95	$1.28 \cdot 10^{-8}$	5.58
2^{-5}	$9.52 \cdot 10^{-6}$	2.95	$2.40 \cdot 10^{-7}$	3.98	$8.05 \cdot 10^{-9}$	4.70	$4.49 \cdot 10^{-9}$	1.51

(9.4)
$$K_0 = \frac{D\varphi^{-1}(K \circ \varphi) D\varphi^{-\top}}{|\det(D\varphi^{-1})|}, \qquad \text{where} \qquad \varphi(x,y) = \begin{bmatrix} x \\ (2ay - a)\sin^2(2\pi x) + y \end{bmatrix}.$$

The manufactured solution u is chosen to be $u(x,y) = \sin(\varphi_1(x,y))\sinh(\varphi_2(x,y))$. The restriction of u to the boundary is chosen as Dirichlet datum g, and the right-hand-side f is directly computed by inserting u into the strong form of the equation (9.1).

Here, we perform the same verification as in the previous subsection, that is, fixing h and varying q and H with the same meshes and solver settings. In Tables 3 and 4, the relative H^1 and L^2 errors for decreasing mesh sizes H are shown. Both tables show the expected convergence rates. In the case of $q = 4$, the convergence rate deteriorates for small macromesh sizes H because the discretization error is dominating.

Additionally, we present results for fixed $H = 2^{-3}H_0$ and varying h and q. Table 5 shows the relative L^2 errors and convergence rates of the standard approach and the surrogate approach with $q \in \{3, 5, 7\}$. Only for $q = 7$, the L^2 error coincides with the

TABLE 5

Degrees of Freedom (DoFs), relative L^2 errors, experimental orders of convergence, and relative time to solutions for fixed H and varying q and h in the case of problem (9.1) *with the tensorial coefficient* (9.3) *and curved boundary.*

			standard		$q=3$			$q=5$			$q=7$		
$\frac{h}{H_0}$	$\frac{H_{\mathrm{LS}}}{h}$	DoFs	Rel. L^2 err.	EOC	Rel. L^2 err.	EOC	RTTS	Rel. L^2 err.	EOC	RTTS	Rel. L^2 err.	EOC	RTTS
2^{-6}	2^0	$4.2 \cdot 10^3$	$7.14 \cdot 10^{-5}$	-	$7.15 \cdot 10^{-5}$	-	0.93	$7.14 \cdot 10^{-5}$	-	0.94	$7.14 \cdot 10^{-5}$	-	1.03
2^{-7}	2^0	$1.7 \cdot 10^4$	$1.81 \cdot 10^{-5}$	1.98	$1.87 \cdot 10^{-5}$	1.94	0.87	$1.81 \cdot 10^{-5}$	1.98	0.87	$1.81 \cdot 10^{-5}$	1.98	0.96
2^{-8}	2^1	$6.6 \cdot 10^4$	$4.55 \cdot 10^{-6}$	1.99	$6.76 \cdot 10^{-6}$	1.46	0.64	$4.55 \cdot 10^{-6}$	1.99	0.67	$4.55 \cdot 10^{-6}$	1.99	0.75
2^{-9}	2^1	$2.6 \cdot 10^5$	$1.14 \cdot 10^{-6}$	2.00	$5.70 \cdot 10^{-6}$	0.25	0.35	$1.14 \cdot 10^{-6}$	1.99	0.37	$1.14 \cdot 10^{-6}$	2.00	0.41
2^{-10}	2^1	$1.1 \cdot 10^6$	$2.85 \cdot 10^{-7}$	2.00	$6.11 \cdot 10^{-6}$	-0.10	0.15	$3.00 \cdot 10^{-7}$	1.93	0.16	$2.85 \cdot 10^{-7}$	2.00	0.19
2^{-11}	2^1	$4.2 \cdot 10^6$	$7.14 \cdot 10^{-8}$	2.00	$6.35 \cdot 10^{-6}$	-0.06	0.07	$1.16 \cdot 10^{-7}$	1.37	0.08	$7.14 \cdot 10^{-8}$	2.00	0.10
2^{-12}	2^1	$1.7 \cdot 10^7$	$1.79 \cdot 10^{-8}$	2.00	$6.47 \cdot 10^{-6}$	-0.03	0.05	$9.49 \cdot 10^{-8}$	0.29	0.06	$1.79 \cdot 10^{-8}$	2.00	0.07
2^{-13}	2^1	$6.7 \cdot 10^7$	$4.50 \cdot 10^{-9}$	1.99	$6.52 \cdot 10^{-6}$	-0.01	0.04	$9.43 \cdot 10^{-8}$	0.01	0.05	$4.54 \cdot 10^{-9}$	1.98	0.07

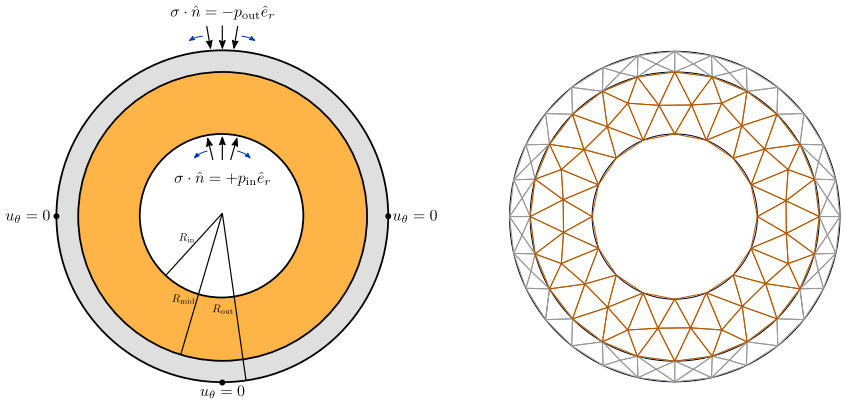


FIG. 9. *Linear elasticity problem setup (left) and initial macromesh $\mathcal{T}_H$ (right).*

errors from the standard approach for all h. The relative time to solution (RTTS) shown for the surrogate approaches is defined as the time to solution including the setup phase of the surrogate approach divided by the time to solution of the standard approach. In the case with the smallest h, the surrogate approach took at most only 7% of the time of the standard approach. That is, a speed-up by more than a factor of 14.

9.2. Linearized elasticity example. In this subsection, we compare a standard method with a surrogate method applied to the linearized elasticity problem presented in subsection 4.2. In our surrogate method, we employ the zero row sum property described in section 5. We choose an annular domain composed of two distinct and concentric materials under uniform pressure loading. The problem is inspired by a similar three-dimensional experiment documented in [27]. Let $\mathcal{B}_r \subseteq \mathbb{R}^2$ be the two-dimensional open ball of radius r with the midpoint at the origin. The computational domain is then defined as $\Omega = \mathcal{B}_{R_{\mathrm{out}}} \setminus \mathcal{B}_{R_{\mathrm{in}}}$. We split this domain into two disjoint sets $\Omega_I = \mathcal{B}_{R_{\mathrm{mid}}} \setminus \mathcal{B}_{R_{\mathrm{in}}}$ and $\Omega_O = \mathcal{B}_{R_{\mathrm{out}}} \setminus \mathcal{B}_{R_{\mathrm{mid}}}$ corresponding to each material. In our experiments, we fix $R_{\mathrm{in}} = 1\,\mathrm{cm}$, $R_{\mathrm{mid}} = 1.75\,\mathrm{cm}$, and $R_{\mathrm{out}} = 2\,\mathrm{cm}$. Refer to the leftmost diagram in Figure 9 for an illustration of the setup. Here, the macroelements adjacent to the boundary and the material interface are mapped to the physical geometry by using the transformation described in [41].

Let $r(x) = |x|$. The strong form of the problem is

$$
\begin{aligned}
-\mathrm{Div}\,(\sigma) &= \vec{f} && \text{in } \Omega,\\
u_\theta &= 0 && \text{on } \{(-R_{\mathrm{out}},0)^\top,(0,-R_{\mathrm{out}})^\top,(R_{\mathrm{out}},0)^\top\},\\
\sigma\cdot\hat{n} &= p_{\mathrm{in}}\hat{e}_r && \text{on } \{x\in\partial\Omega : r = R_{\mathrm{in}}\},\\
\sigma\cdot\hat{n} &= -p_{\mathrm{out}}\hat{e}_r && \text{on } \{x\in\partial\Omega : r = R_{\mathrm{out}}\}.
\end{aligned} \tag{9.5}
$$

Here, the stress tensor σ is given by Hooke's law for isotropic materials as defined in subsection 4.2. The unit vector in radial direction is denoted $\hat{e}_r$, and the outward pointing unit normal vector is denoted $\hat{n}$. We neglect body forces and therefore set $\vec{f} = \vec{0}$. The displacement is described in polar coordinates, where u_r is the radial displacement and u_θ is the tangential displacement. In order to make the system uniquely solvable, we enforce the tangential displacement u_θ to be zero at three points; see Figure 9. The materials are chosen to be cork in the inner domain Ω_I with an A36 steel layer in the outer domain Ω_O. Poisson's ratio and Young's modulus for this scenario are $E_I = 0.02\,\mathrm{GPa}$, $E_O = 200.0\,\mathrm{GPa}$, $\nu_I = 0$, and $\nu_O = 0.26$. The Lamé parameters are obtained by the expressions $\mu = \frac{E}{2(1+\nu)}$ and $\lambda = \frac{E\nu}{(1-2\nu)\cdot(1+\nu)}$. Note that while these expressions induce piecewise-constant Lamé parameters, $\lambda = \lambda(r)$ and $\mu = \mu(r)$, once the smooth domain is mapped to the computational domain depicted on the right of Figure 9, these parameters will not be piecewise-constant anymore due to the transformation. The pressure on the outer boundary is set to $p_{\mathrm{out}} = 0\,\mathrm{MPa}$, and on the inner boundary $p_{\mathrm{in}} = 1\,\mathrm{MPa}$ is prescribed.

In this particular scenario, there is an analytic solution available for the radial displacement u_r which has the form

$$
u_r(r) = \begin{cases} A\cdot r + B\cdot r^{-1} & \text{if } r\in[R_{\mathrm{in}},R_{\mathrm{mid}}]\,,\\ C\cdot r + D\cdot r^{-1} & \text{if } r\in(R_{\mathrm{mid}},R_{\mathrm{out}}]\,. \end{cases} \tag{9.6}
$$

The tangential displacement u_θ is zero everywhere due to the symmetry of the problem. In (9.6), the coefficients A, B, C, and D are uniquely determined by the following system of linear equations:

$$
\begin{bmatrix}
E_I R_{\mathrm{in}}^2 & E_I\,(2\nu_I-1) & 0 & 0\\
0 & 0 & E_O R_{\mathrm{out}}^2 & E_O\,(2\nu_O-1)\\
R_{\mathrm{mid}}^2 & 1 & -R_{\mathrm{mid}}^2 & -1\\
-E_I R_{\mathrm{mid}}^2 d_O & -d_O E_I\,(2\nu_I-1) & E_O R_{\mathrm{mid}}^2 d_I & d_I E_O\,(2\nu_O-1)
\end{bmatrix}
\begin{bmatrix} A\\ B\\ C\\ D \end{bmatrix}
=
\begin{bmatrix} p_{in} R_{\mathrm{in}}^2 d_I\\ p_{out} R_{\mathrm{out}}^2 d_O\\ 0\\ 0 \end{bmatrix}, \tag{9.7}
$$

where $d_I := 2\nu_I^2+\nu_I-1$ and $d_O := 2\nu_O^2+\nu_O-1$. This system is derived after deducing the continuity of the displacement and surface traction at the material interface, then by incorporating the prescribed external forces at the boundaries.

In order to verify the accuracy of the surrogate method, we select the polynomial degree $q = 7$ and the macromesh $\mathcal{T}_H$, illustrated on the right of Figure 9. Note that the discontinuity in the material parameters lies along the macroelement interfaces, and so $\lambda,\mu \in \prod_{T\in\mathcal{T}_H} C^\infty(T) \subsetneq W^{r+1,\infty}(\mathcal{T}_H)$ for any $r > 0$.

Each linear system is solved by applying geometric multigrid iterations with V(3,3) cycles until the relative residual is reduced by the factor $1\cdot 10^{-7}$. On the coarsest level used in the multigrid hierarchy, we employ MUMPS as a direct solver. In Table 6, we report on the results for varying h and present the relative L^2 errors for the standard and surrogate approach, respectively. On the finest mesh involving about $1.8\cdot 10^8$ degrees of freedom, the surrogate approach required only 5% of the

TABLE 6

Relative L^2 errors, experimental orders of convergence, and relative time to solutions for fixed H, $q = 7$, and varying h in the case of the linearized elasticity problem (9.5).

			Standard		$q = 7$		
$\frac{h}{H}$	$\frac{H_{\mathrm{LS}}}{h}$	DoFs	Rel. L^2 err.	EOC	Rel. L^2 err.	EOC	RTTS
2^{-3}	2^0	$1.1 \cdot 10^4$	$3.93 \cdot 10^{-4}$	-	$3.93 \cdot 10^{-4}$	-	0.57
2^{-4}	2^0	$4.5 \cdot 10^4$	$9.66 \cdot 10^{-5}$	2.02	$9.66 \cdot 10^{-5}$	2.02	0.38
2^{-5}	2^1	$1.8 \cdot 10^5$	$2.39 \cdot 10^{-5}$	2.02	$2.39 \cdot 10^{-5}$	2.02	0.31
2^{-6}	2^2	$7.1 \cdot 10^5$	$5.92 \cdot 10^{-6}$	2.01	$5.92 \cdot 10^{-6}$	2.01	0.19
2^{-7}	2^2	$2.8 \cdot 10^6$	$1.47 \cdot 10^{-6}$	2.01	$1.47 \cdot 10^{-6}$	2.01	0.12
2^{-8}	2^2	$1.1 \cdot 10^7$	$3.68 \cdot 10^{-7}$	2.00	$3.68 \cdot 10^{-7}$	2.00	0.08
2^{-9}	2^2	$4.5 \cdot 10^7$	$9.18 \cdot 10^{-8}$	2.00	$9.19 \cdot 10^{-8}$	2.00	0.06
2^{-10}	2^2	$1.8 \cdot 10^8$	$2.30 \cdot 10^{-8}$	2.00	$2.31 \cdot 10^{-8}$	1.99	0.05

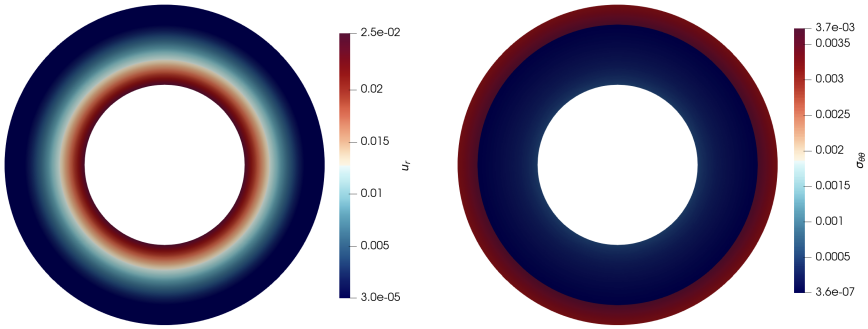


FIG. 10. *Plots of radial displacement u_r (left) and the tangential stress $\sigma_{\theta\theta}$ (right) computed on the fine mesh $\mathcal{S}^6(\mathcal{T}_H)$, corresponding to $h = 2^{-6}H$, with $q = 4$.*

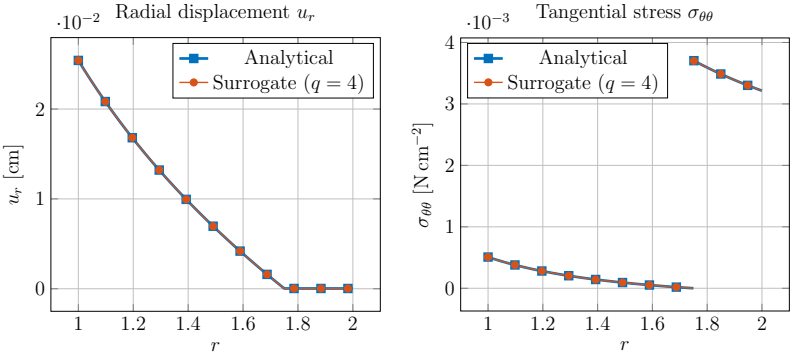


FIG. 11. *Plots over line of the radial displacement u_r (left) and the tangential stress $\sigma_{\theta\theta}$ (right) computed on the fine mesh $\mathcal{S}^6(\mathcal{T}_H)$, corresponding to $h = 2^{-6}H$, with $q = 4$.*

time required by the standard approach while having the same accuracy. That is a speed-up by a factor of 20. Figure 10 shows the radial displacement u_r and the tangential stress $\sigma_{\theta\theta}$ computed with the surrogate approach on the fine mesh $\mathcal{S}^6(\mathcal{T}_H)$, corresponding to $h = 2^{-6}H$, with $q = 4$. This is illustrated further by the plots in Figure 11 which allow a visual comparison between u_r and $\sigma_{\theta\theta}$ in the surrogate and analytical solutions.

9.3. *p*-Laplacian diffusion example. In this subsection, we consider the time-dependent example introduced in subsection 4.3. Here, we solve the nonlinear p-Laplacian diffusion problem (9.8), given in strong form as

$$\begin{aligned} \frac{\partial u}{\partial t} - \operatorname{div}\left(|\nabla u|^{p-2}\cdot u\right) &= f \quad && \text{in } \Omega\times(0,T]\,,\\ u &= 0 && \text{on } \partial\Omega\times(0,T]\,,\\ u &= u_0 && \text{in } \Omega\times\{0\}\,. \end{aligned} \tag{9.8}$$

The computational domain is set to the unit disk, i.e., $\Omega := \mathcal{B}_1$ and the right-hand side is set to the constant function $f(x) = 2\cdot(p')^{p/p'}$, where $p' = \frac{p}{p-1}$ is the Hölder conjugate of p. The initial solution is set to $u_0(x) = 0.1\cdot\left(1-|x|^2\right)$. For this particular problem, the stationary limit u_∞ has unit magnitude; namely, $u_\infty(x) = 1-|x|^{p'}$ [6, Example 3.1].

Our discretization follows a standard approach where a mass matrix $\mathsf{M}_{ij} = \int_\Omega \phi_i\phi_j\,\mathrm{d}x$ and a stiffness matrix $\mathsf{A}_{ij}(\widetilde{\mathsf{u}}) = \int_\Omega |\nabla\widetilde{u}_h|^{p-2}\nabla\phi_j\cdot\nabla\phi_i\,\mathrm{d}x$ are introduced. At this point, $\widetilde{\mathsf{u}}$ is the coefficient vector, in the $\{\phi_i\}$ basis, of an arbitrary discrete function $\widetilde{u}_h$. The time derivative is discretized by a backward Euler scheme, and the non-linearity in each time step is resolved by Picard fixed-point iterations. Let u_k^l be the coefficient vector of the discrete solution at the kth time step and lth fixed point iteration. Employing the bilinear form (4.3) and fixing a time step size $dt > 0$, the discrete problem in each time step $k > 0$ and fixed point iteration $l > 0$ reads as follows:

$$\left(\mathsf{M} + dt\mathsf{A}\left(\mathsf{u}_k^{l-1}\right)\right)\mathsf{u}_k^l = \mathsf{M}\mathsf{u}_{k-1} + dt\mathsf{M}\mathsf{f}\,, \tag{9.9}$$

where u_{k-1} is the final coefficient vector from the previous time step. In each time step, this system is solved multiple times (once for each fixed-point iteration) by the application of five V(2,2) multigrid cycles. The fixed-point iterations continue until the relative increment $\frac{\|\mathsf{u}_k^l-\mathsf{u}_k^{l-1}\|_2}{\|\mathsf{u}_k^l\|_2}$ is smaller than the fixed tolerance $1\cdot10^{-3}$. Then k is incremented and a new $\mathsf{u}_{k-1} = \mathsf{u}_{k-1}^l$ is defined.

In our surrogate method, the stencil functions of the stiffness matrix $\mathsf{A}\left(\mathsf{u}^{k-1}\right)$ are approximated by solving the least-squares problems after every fixed-point iteration, all the while enforcing the zero row sum property (cf. subsection 9.1). Meanwhile, the stencil function of the mass matrix M is only approximated once in a preprocessing step because it does not depend on any free variables in the computation. The time step surrogate polynomials of both operators are then simply summed together to obtain the time step matrix $\mathsf{M}+dt\mathsf{A}\left(\mathsf{u}^{k-1}\right)$. This particular splitting of the surrogate matrices, which reproduces the zero row sum property in the stiffness matrix, allows for faster reapproximation of the time step matrix stencil function and appears to improve the stability of the method. In this example, we did not enforce the symmetry condition featured in subsection 3.5. Instead, whenever a vertex $x_i \in \mathbb{X}_m$ was on the boundary of a macroelement ∂T_m, we set the surrogate stiffness matrix to the exact value stiffness matrix $\widetilde{\mathsf{A}}_{ij} = \mathsf{A}_{ij}$. This minor asymmetry is more amenable to computation because there is less data transfer and it did not affect the behavior of our multigrid solver. In fact, this choice improved our results with this problem, which we believe is due to better accuracy in the surrogate near the singularity in the coefficient, i.e., at the origin $x = (0,0)$. The success of this approach suggests that the definition

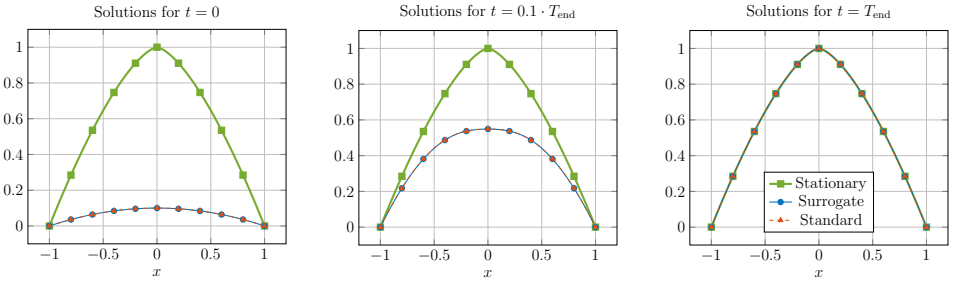


FIG. 12. *Plots of standard, surrogate, and stationary analytical solution over the line* $[0,1]\times\{0\}$ *for different times* $t = 0$, $t = 0.1 \cdot T$, *and* $t = T_{\text{end}}$.

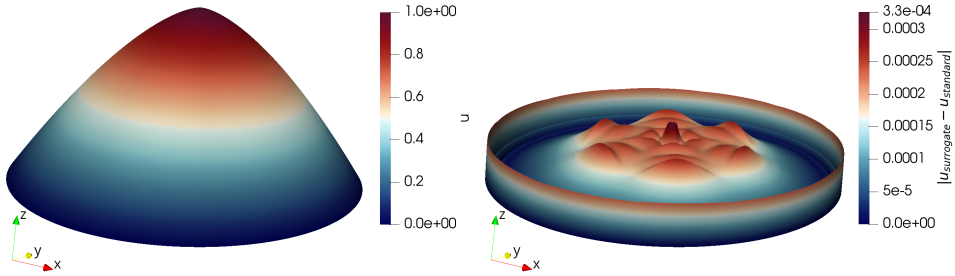


FIG. 13. *Surface plot of nonstationary p-Laplacian surrogate solution for* $t = T_{\text{end}}$ *with* $p = 3$ *and* $q = 6$ *(left). Absolute difference between the discrete standard and surrogate solution at the time* $t = T_{\text{end}}$ *(right).*

given in (3.17) may be relaxed in other applications as well.[1] In the proximity of this singularity and for $p > 2$, the coefficient depending on the solution of the previous fixed-point iteration is getting very close to zero which serves as a challenge for the approximated off-diagonal stencil functions. Depending on the polynomial degree q, they might erroneously take on positive values due to overshoots which possibly results in a loss of positive definiteness of the surrogate matrix. However, this drawback could not be observed in the scenario considered in the following example.

The unit-disk is discretized by the macromesh $\mathcal{T}_H$ featured on the right of Figure 2. Note that the vertices of the central macroelements meet at the origin $x = (0,0)$, i.e., exactly where the singularity occurs in the stationary limit u_∞. The simulations are conducted on the mesh $\mathcal{S}^9(\mathcal{T}_H)$, which involves about $4.72 \cdot 10^6$ degrees of freedom. The macroelements adjacent to the boundary are mapped to the physical geometry by using the mapping described in [41]. Furthermore, the least-squares regressions are carried out on the mesh corresponding to $H_{\text{LS}} = 4h$, and the polynomial degree of the approximated stencil functions is fixed to $q = 6$. In this scenario, we consider the p-Laplacian operator with $p = 3$, fix the time step size $\Delta t = 1 \cdot 10^{-2}$, and solve until time $T_{\text{end}} = 1$. Figure 12 illustrates the standard and surrogate solutions plotted over the line $[0,1] \times \{0\}$ for different times t. In the left of Figure 13, the surface plot of the surrogate solution is depicted. Since the difference of the solutions is very small, we added in the right of Figure 13 a surface plot of the absolute difference of the surrogate and standard solution at the final time $t = T_{\text{end}}$. The surrogate approach

[1] See [10] for further evidence.

required only 5.4% of the time required by the standard approach, that is, a speed-up by more than a factor of 18.

Appendix A. Proofs.

Proof of Proposition 6.1. By the min-max theorem [21], the kth eigenvalue of M is

$$\lambda_k(\mathsf{M}) = \min_{W \subseteq \mathbb{R}^N} \left\{ \max_{\|\mathsf{x}\|_2=1} \left\{ \mathsf{x}^\top \mathsf{M}\mathsf{x} : \mathsf{x} \in W \right\} : \dim W = k \right\} \tag{A.1}$$

$$= \max_{W \subseteq \mathbb{R}^N} \left\{ \min_{\|\mathsf{x}\|_2=1} \left\{ \mathsf{x}^\top \mathsf{M}\mathsf{x} : \mathsf{x} \in W \right\} : \dim W = N-k+1 \right\}. \tag{A.2}$$

Define $\mathsf{D} = \mathsf{M} - \mathsf{N}$. We first show that $\lambda_1(\mathsf{D}) \le \lambda_k(\mathsf{M}) - \lambda_k(\mathsf{N}) \le \lambda_N(\mathsf{D})$. Indeed,

$$\lambda_k(\mathsf{M}) \le \min_{W \subseteq \mathbb{R}^N} \left\{ \max_{\|\mathsf{x}\|_2=1} \left\{ \mathsf{x}^\top \mathsf{N}\mathsf{x} : \mathsf{x} \in W \right\} + \max_{\|\mathsf{x}\|_2=1} \left\{ \mathsf{x}^\top \mathsf{D}\mathsf{x} : \mathsf{x} \in W \right\} : \dim W = k \right\} \tag{A.3}$$

$$\le \min_{W \subseteq \mathbb{R}^N} \left\{ \max_{\|\mathsf{x}\|_2=1} \left\{ \mathsf{x}^\top \mathsf{N}\mathsf{x} : \mathsf{x} \in W \right\} : \dim W = k \right\} + \max_{\|\mathsf{x}\|_2=1} \left\{ \mathsf{x}^\top \mathsf{D}\mathsf{x} : \mathsf{x} \in \mathbb{R}^N \right\} \tag{A.4}$$

$$= \lambda_k(\mathsf{N}) + \lambda_N(\mathsf{D}) \tag{A.5}$$

and, likewise,

$$\lambda_k(\mathsf{M}) \ge \max_{W \subseteq \mathbb{R}^N} \left\{ \min_{\|\mathsf{x}\|_2=1} \left\{ \mathsf{x}^\top \mathsf{N}\mathsf{x} : \mathsf{x} \in W \right\} + \min_{\|\mathsf{x}\|_2=1} \left\{ \mathsf{x}^\top \mathsf{D}\mathsf{x} : \mathsf{x} \in W \right\} : \dim W = N-k+1 \right\} \tag{A.6}$$

$$\ge \max_{W \subseteq \mathbb{R}^N} \left\{ \min_{\|\mathsf{x}\|_2=1} \left\{ \mathsf{x}^\top \mathsf{N}\mathsf{x} : \mathsf{x} \in W \right\} : \dim W = N-k+1 \right\} + \min_{\|\mathsf{x}\|_2=1} \left\{ \mathsf{x}^\top \mathsf{D}\mathsf{x} : \mathsf{x} \in \mathbb{R}^N \right\} \tag{A.7}$$

$$= \lambda_k(\mathsf{N}) + \lambda_1(\mathsf{D}). \tag{A.8}$$

This immediately leads us to the inequality $|\lambda_k(\mathsf{M}) - \lambda_k(\mathsf{N})| \le \max\{|\lambda_1(\mathsf{D})|, |\lambda_N(\mathsf{D})|\}$. Now, for at least one i, $|\lambda_N(\mathsf{D})| - |\mathsf{D}_{ii}| \le |\lambda_N(\mathsf{D}) - \mathsf{D}_{ii}| \le \sum_{j \ne i} |\mathsf{D}_{ij}|$, by the Gershgorin circle theorem. Therefore, $|\lambda_N(\mathsf{D})| \le \sum_j |\mathsf{D}_{ij}| \le \|\mathsf{D}\|_\infty$. Similarly, $|\lambda_1(\mathsf{D})| \le \|\mathsf{D}\|_\infty$. ∎

Acknowledgments. The authors would like to thank Marcus Mohr and each of the referees for the detailed feedback they gave during the review process. Their insights significantly improved the quality of the final manuscript.

REFERENCES

[1] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 15–41.

[2] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet, *Hybrid scheduling for the parallel solution of linear systems*, Parallel Comput., 32 (2006), pp. 136–156.

[3] P. Arbenz, G. H. van Lenthe, U. Mennel, R. Müller, and M. Sala, *A scalable multi-level preconditioner for matrix-free μ-finite element analysis of human bone structures*, Internat. J. Numer. Methods Engrg., 73 (2008), pp. 927–947.

[4] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc Users Manual*, Tech. Rep. ANL-95/11 - Revision 3.9, Argonne National Laboratory, Lemont, IL, 2018.

[5] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser, Basel, Switzerland, 1997, pp. 163–202.

[6] J. W. Barrett and W. B. Liu, *Finite element approximation of the p-Laplacian*, Math. Comp., 61 (1993), pp. 523–537.

[7] S. Bauer, D. Drzisga, M. Mohr, U. Rüde, C. Waluga, and B. Wohlmuth, *A stencil scaling approach for accelerating matrix-free finite element implementations*, SIAM J. Sci. Comput., 40 (2018), pp. C748–C778.

[8] S. Bauer, M. Huber, S. Ghelichkhan, M. Mohr, U. Rüde, and B. Wohlmuth, *Large-scale simulation of mantle convection based on a new matrix-free approach*, J. Comput. Sci., 31 (2019), pp. 60–76.

[9] S. Bauer, M. Huber, M. Mohr, U. Rüde, and B. Wohlmuth, *A new matrix-free approach for large-scale geodynamic simulations and its performance*, in Computational Science–ICCS 2018, Y. Shi, H. Fu, Y. Tian, V. V. Krzhizhanovskaya, M. H. Lees, F. Dongerra, and P. M. A. Sloot, eds., Springer, Cham, Switzerland, 2018, pp. 17–30.

[10] S. Bauer, M. Mohr, U. Rüde, J. Weismüller, M. Wittmann, and B. Wohlmuth, *A two-scale approach for efficient on-the-fly operator assembly in massively parallel high performance multigrid codes*, Appl. Numer. Math., 122 (2017), pp. 14–38.

[11] B. Bergen, *Hierarchical Hybrid Grids: Data Structures and Core Algorithms for Efficient Finite Element Simulations on Supercomputers: Hierarchische Hybride Gitter: Datenstrukturen und Algorithmen Zur Effizienten Simulation Mit Finiten Elementen Auf Höchstleistungsrechnern*, SCS Publishing House, San Diego, CA, 2005.

[12] B. Bergen and F. Hülsemann, *Hierarchical hybrid grids: Data structures and core algorithms for multigrid*, Numer. Linear Algebra Appl., 11 (2004), pp. 279–291.

[13] B. Bergen, G. Wellein, F. Hülsemann, and U. Rüde, *Hierarchical hybrid grids: Achieving TERAFLOP performance on large scale finite element simulations*, Internat. J. Parallel Emergent Distrib. Syst., 22 (2007), pp. 311–329.

[14] J. Bey, *Tetrahedral grid refinement*, Computing, 55 (1995), pp. 355–378.

[15] J. Bielak, O. Ghattas, and E.-J. Kim, *Parallel octree-based finite element method for large-scale earthquake ground motion simulation*, CMES Comput. Model. Eng. Sci., 10 (2005), pp. 99–112.

[16] S. Brenner and R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer Science & Business Media, New York, NY, 2007.

[17] J. Brown, *Efficient nonlinear solvers for nodal high-order finite elements in 3D*, J. Sci. Comput., 45 (2010), pp. 48–63.

[18] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical Analysis*, Cengage Learning, Boston, MA, 2015.

[19] G. F. Carey and B.-N. Jiang, *Element-by-element linear and nonlinear solution schemes*, Commun. Appl. Numer. Methods, 2 (1986), pp. 145–153.

[20] P. G. Ciarlet, *Three-Dimensional Elasticity*, Elsevier, New York, NY, 1988.

[21] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, Vol. 1, Wiley, Hoboken, NJ, 1953.

[22] D. Drzisga, B. Keith, and B. Wohlmuth, *The Surrogate Matrix Methodology: Low-Cost Assembly for Isogeometric Analysis*, arXiv preprint arXiv:1904.06971, 2019.

[23] D. Drzisga, U. Rüde, and B. Wohlmuth, *Stencil Scaling for Vector-Valued PDEs with Applications to Generalized Newtonian Fluids*, arXiv preprint arXiv:1908.08666, 2019.

[24] C. Engwer, R. D. Falgout, and U. M. Yang, *Stencil computations for PDE-based applications with examples from DUNE and Hypre*, Concurrency Comput. Practice Experience, 29 (2017), e4097.

[25] A. Ern and J.-L. Guermond, *Theory and Practice of Finite Elements*, Springer Science & Business Media, New York, NY, 2013.

[26] C. Flaig and P. Arbenz, *A highly scalable matrix-free multigrid solver for μFE analysis based on a pointer-less octree*, in Large-Scale Scientific Computing: 8th International Conference, LSSC 2011, Sozopol, Bulgaria, June 6-10, 2011, Revised Selected Papers, I. Lirkov, S. Margenov, and J. Waśniewski, eds., Springer, Berlin, 2012, pp. 498–506.

[27] F. Fuentes, B. Keith, L. Demkowicz, and P. Le Tallec, *Coupled variational formulations of linear elasticity and the DPG methodology*, J. Comput. Phys., 348 (2017), pp. 715–731.
[28] B. Gmeiner, U. Rüde, H. Stengel, C. Waluga, and B. Wohlmuth, *Towards textbook efficiency for parallel multigrid*, Numer. Math. Theory Methods Appl., 8 (2015), pp. 22–46.
[29] P. Grisvard, *Elliptic Problems in Nonsmooth Domains*, Pitman, Boston, MA, 1985.
[30] G. Guennebaud, B. Jacob, et al., *Eigen,* v3, http://eigen.tuxfamily.org, 2010.
[31] N. Kohl, D. Thönnes, D. Drzisga, D. Bartuschat, and U. Rüde, *The HyTeG finite-element software framework for scalable multigrid solvers*, Internat. J. Parallel Emergent Distrib. Syst., 34 (2019), pp. 477–496.
[32] M. Kronbichler and K. Kormann, *A generic interface for parallel cell-based finite element operator application*, Comput. & Fluids, 63 (2012), pp. 135–147.
[33] K. Ljungkvist, *Matrix-free finite-element computations on graphics processors with adaptively refined unstructured meshes*, in Proceedings of the 25th High Performance Computing Symposium, HPC '17, Society for Computer Simulation International, 2017, pp. 1:1–1:12.
[34] K. Ljungkvist and M. Kronbichler, *Multigrid for Matrix-Free Finite Element Computations on Graphics Processors*, Tech. Rep. 2017-006, Department of Information Technology, Uppsala University, Uppsala, Sweden, 2017.
[35] J. Loffeld and J. Hittinger, *On the arithmetic intensity of high-order finite-volume discretizations for hyperbolic systems of conservation laws*, Internat. J. High Performance Comput. Appl., 33 (2019), pp. 25–52.
[36] D. A. May, J. Brown, and L. L. Pourhiet, *A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow*, Comput. Methods Appl. Mech. Engrg., 290 (2015), pp. 496–523.
[37] D. A. May, P. Sanan, K. Rupp, M. G. Knepley, and B. F. Smith, *Extreme-scale multigrid components within PETSc*, in Proceedings of the Platform for Advanced Scientific Computing Conference, PASC '16, New York, NY, 2016, ACM, pp. 5:1–5:12.
[38] L. R. Scott and S. Zhang, *Finite element interpolation of nonsmooth functions satisfying boundary conditions*, Math. Comp., 54 (1990), pp. 483–493.
[39] G. Strang, *Variational crimes in the finite element method*, in The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations, A. K. Aziz, ed., Elsevier, New York, NY, 1972, pp. 689–710.
[40] B. van Rietbergen, H. Weinans, R. Huiskes, and B. Polman, *Computational strategies for iterative solutions of large FEM applications employing voxel data*, Internat. J. Numer. Methods Engrg., 39 (1996), pp. 2743–2767.
[41] M. Zlámal, *Curved elements in the finite element method.* I, SIAM J. Numer. Anal., 10 (1973), pp. 229–240.

A.2. The surrogate matrix methodology: Low-cost assembly for isogeometric analysis

The surrogate matrix methodology: Low-cost assembly for isogeometric analysis

Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth

Following the successful application and analysis of the surrogate matrix methodology for a low-order matrix-free finite element method in core article I (Appendix A.1), we investigate its applicability to IGA. The main idea is again based on the two-scale approach which was introduced in the context of first-order finite elements by Bauer et al. in [13]. It turns out that using uniform knot vectors in IGA has some beneficial properties which can be exploited to efficiently construct surrogate matrices. In this article, we present a methodology to avoid over-computation in the assembly of matrices in IGA. It is based on performing integration for only a small subset of the trial and test basis function interactions while the rest is approximated through interpolation. The majority of entries in the resulting sparse matrices are computed without using any integration at all. Our methodology is independent of the integration rule used at the element or basis functions level. Therefore, it may be used in conjunction with one of the state of the art integration schemes based on, e.g., reduced or weighted integration. Many techniques accelerating the formation and assembly of matrices in IGA demonstrate their strength only when high-order approximations are used, but with this approach, performance usually grows with h-refinement. For demonstration purposes, we implemented our surrogate methods by modifying the assembly routines in the open-source library GeoPDEs [51, 102]. A more detailed description of these modifications is provided in the companion article [41] which includes references to the code in order to allow reproducibility.

In Section 2, we set up the majority of mathematical notation used in the remainder of the paper. In Section 3, we introduce the notion of a stencil function in the IGA context. In Section 4, we investigate the accuracy of B-spline interpolations with regard to the approximation of stencil functions. In Section 5, we use these interpolants of the stencil functions to define surrogate matrices for IGA. Section 6 contains a brief description of our software implementation. In Section 7, we consider the surrogate matrix methodology for Poisson's problem and provide a priori error estimates which are verified by numerical experiments. In Section 8, we consider the surrogate IGA method for the analysis of transverse vibrations of an isotropic elastic membrane and provide an a priori analysis for the eigenvalue errors. In Sections 9 and 10, we examine surrogate IGA methods for plate bending and Stokes flow problems, respectively. The appendix is included to substantiate some of the claims made in Section 3 and to support some of the analysis conducted in Section 4.

I was significantly involved in finding the ideas and primarily responsible for setting up the mathematical framework and carrying out the scientific work presented in this article. Furthermore, I was in charge of writing the article with the exception of the appendix. The co-authors contributed by making corrective changes.

Permission to include:

Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth
The surrogate matrix methodology: Low-cost assembly for isogeometric analysis
Computer Methods in Applied Mechanics and Engineering 361 (2020): 112776
(see also article [43] in the bibliography)

The following pages on copyright are excerpts from copies of the website

https://www.elsevier.com/about/policies/copyright#Author-rights

retrieved on 22 March 2020.

Copyright

Describes the rights related to the publication and distribution of research. It governs how authors (as well as their employers or funders), publishers and the wider general public can use, publish and distribute articles or books.

Journal author rights Government employees Elsevier's rights Protecting author rights Open access

Journal author rights

In order for Elsevier to publish and disseminate research articles, we need publishing rights. This is determined by a publishing agreement between the author and Elsevier. This agreement deals with the transfer or license of the copyright to Elsevier and authors retain significant rights to use and share their own published articles. Elsevier supports the need for authors to share, disseminate and maximize the impact of their research and these rights, in Elsevier proprietary journals* are defined below:

For subscription articles	For open access articles
Authors transfer copyright to the publisher as part of a journal publishing agreement, but have the right to: • Share their article for Personal Use, Internal Institutional Use and Scholarly Sharing purposes, with a DOI link to the version of record on ScienceDirect (and with the Creative Commons CC-BY-NC- ND license for author manuscript versions) • Retain patent, trademark and other intellectual property rights (including research data). • Proper attribution and credit for the published work.	Authors sign an exclusive license agreement, where authors have copyright but license exclusive rights in their article to the publisher**. In this case authors have the right to: • Share their article in the same ways permitted to third parties under the relevant user license (together with Personal Use rights) so long as it contains a CrossMark logo, the end user license, and a DOI link to the version of record on ScienceDirect. • Retain patent, trademark and other intellectual property rights (including research data). • Proper attribution and credit for the published work.

***Please note that society or third party owned journals may have different publishing agreements. Please see the journal's guide for authors for journal specific copyright information.**

****This includes the right for the publisher to make and authorize commercial use, please see "Rights granted to Elsevier" for more details.**

Help and Support

- Download a sample publishing agreement for subscription articles in ↓ English and ↓ French.
- Download a sample publishing agreement for open access articles for authors choosing a ↓ commercial user license and ↓ non-commercial user license.
- For authors who wish to self-archive see our sharing guidelines
- See our author pages for further details about how to promote your article.
- For use of Elsevier material not defined below please see our permissions page or visit the Permissions Support Center ↗ .

For Elsevier proprietary journals the following steps apply:

1. Authors sign a publishing agreement where they will have copyright but grant broad publishing and distribution rights to the publisher, including the right to publish the article on Elsevier's online platforms.
2. The author chooses an end user license under which readers can use and share the article.
3. The publisher makes the article available online with the author's choice of end user license.

Quick definitions

Personal use

Authors can use their articles, in full or in part, for a wide range of scholarly, non-commercial purposes as outlined below:

- Use by an author in the author's classroom teaching (including distribution of copies, paper or electronic)
- Distribution of copies (including through e-mail) to known research colleagues for their personal use (but not for Commercial Use)
- Inclusion in a thesis or dissertation (provided that this is not to be published commercially)
- Use in a subsequent compilation of the author's works
- Extending the Article to book-length form
- Preparation of other derivative works (but not for Commercial Use)
- Otherwise using or re-using portions or excerpts in other works

These rights apply for all Elsevier authors who publish their article as either a subscription article or an open access article. In all cases we require that all Elsevier authors always include a full acknowledgement and, if appropriate, a link to the final published version hosted on Science Direct.

Commercial use

This is defined as the use or posting of articles:

- **For commercial gain without a formal agreement with the publisher.**
  - For example by associating advertising with the full-text of the article, by providing hosting services to other repositories or to other organizations (including where an otherwise non-commercial site or repository provides a service to other organizations or agencies), or charging fees for document delivery or access
- **To substitute for the services provided directly by the journal.**
  - For example article aggregation, systematic distribution of articles via e-mail lists or share buttons, posting, indexing, or linking for promotional/marketing activities, by commercial companies for use by customers and intended target audiences of such companies (e.g. pharmaceutical companies and healthcare professionals/physician-prescribers).

If you would like information on how to obtain permission for such uses click here or if you would like to make commercial use of the article please visit the Permissions Support Center ↗ .

Internal institutional use

- Use by the author's institution for classroom teaching at the institution and for internal training purposes (including distribution of copies, paper or electronic, and use in coursepacks and courseware programs, but not in Massive Open Online Courses)
- Inclusion of the Article in applications for grant funding
- For authors employed by companies, the use by that company for internal training purposes

Notice of publication and copyright

First Published in "The surrogate matrix methodology: Low-cost assembly for isogeometric analysis" in Computer Methods in Applied Mechanics and Engineering 361 (2020), published by Elsevier B.V.

DOI: https://doi.org/10.1016/j.cma.2019.112776

Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 361 (2020) 112776
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

The surrogate matrix methodology: Low-cost assembly for isogeometric analysis

Daniel Drzisga*, Brendan Keith, Barbara Wohlmuth

Lehrstuhl für Numerische Mathematik, Fakultät für Mathematik (M2), Technische Universität München, Garching bei München, Germany

Received 12 April 2019; received in revised form 22 November 2019; accepted 26 November 2019
Available online xxxx

Abstract

A new methodology in isogeometric analysis (IGA) is presented. This methodology delivers low-cost variable-scale approximations (surrogates) of the matrices which IGA conventionally requires to be computed from element-scale quadrature formulas. To generate surrogate matrices, quadrature must only be performed on certain elements in the computational domain. This, in turn, determines only a subset of the entries in the final matrix. The remaining matrix entries are computed by a simple B-spline interpolation procedure. Poisson's equation, membrane vibration, plate bending, and Stokes' flow problems are studied. In these problems, the use of surrogate matrices has a negligible impact on solution accuracy. Because only a small fraction of the original quadrature must be performed, we are able to report beyond a fifty-fold reduction in overall assembly time in the same software. The capacity for even further speed-ups is clearly demonstrated. The implementation used here was achieved by a small number of modifications to the open-source IGA software library GeoPDEs. Similar modifications could be made to other present-day software libraries.

Keywords: Assembly; Surrogate numerical methods; Isogeometric analysis; A priori error analysis

1. Introduction

Avoiding unnecessary work is of utmost importance when computing at the frontiers of contemporary research. To frame a workable definition, recall that practical simulations in science and engineering involve a large number of possible sources of error. For instance, we highlight the categories of modeling error, numerical error, and data error, each of which have many subcategories. The total error in a simulation is controlled by the aggregate of each relevant source of error. In this paper, "unnecessary work" — or, more precisely, *over-computation* — is any machine expense used to drive one source of error in a problem far below the total error. It cannot be overstated that removing sources of over-computation can have an outsized influence on the computational cost of getting an accurate solution.

In some instances, circumventing over-computation is the simplest way to accelerate a numerical algorithm. For example, in the use of iterative methods, for both linear and non-linear problems, it has long been acknowledged that *over-solving* a discretized problem is a negligent expense. Relaxing iterative solver errors usually reduces to just

* Corresponding author.
E-mail addresses: drzisga@ma.tum.de (D. Drzisga), keith@ma.tum.de (B. Keith), wohlmuth@ma.tum.de (B. Wohlmuth).

https://doi.org/10.1016/j.cma.2019.112776

adjusting the tolerances naturally built into established algorithms. In other instances, sources of over-computation are less conspicuous and avoiding them requires the development of new algorithms. For example, in the field of uncertainty quantification, it has recently come to light that sampling error can be relaxed — and, in turn, computational cost can be significantly reduced — by the use of a tunable *surrogate response surface* [1,2].

The focus of this article is the Galerkin form of isogeometric analysis (IGA) [3,4]. At first sight, in view of the long list of computer methods which rose beforehand, the Galerkin isogeometric method may be seen as a rather paradigmatic approach to the discretization of partial differential equations (PDEs). Indeed, Galerkin IGA methods are little more than finite element methods which employ non-uniform rational B-spline (NURBS) bases [5]. Although it was immediately shown by Hughes et al. [3] that the use of such a basis improves the interoperability between computer-aided design (CAD) and PDE analysis, many other benefits of the IGA approach were also demonstrated early on in the IGA literature. Of particular note, the arbitrary smoothness of NURBS bases generally improves the accuracy per degree of freedom and lends itself to convenient techniques for the discretization of high-order PDEs [5,6]. It is these and other serendipitous features of IGA which have attributing to its truly meteoric success in modern computational science and engineering research.

It is well-established that traditional isogeometric methods face a great computational burden at the point of matrix assembly. This is due, in part, to the large support of the basis functions. Although many other common concerns are naturally alleviated by the IGA paradigm, this particular challenge is clearly evidenced by the expansive literature on quadrature rules and accelerated assembly algorithms [7–19]. Indeed, we may further accentuate this remark with the following quote from the 2014 review article [20]:

> "... at the moment the assembly of the matrix is the most time-consuming part of isogeometric codes. The development of optimal assembly procedures is an important task required to render isogeometric methods a competitive technology."

In this article, we present a simple methodology to avoid *over-assembling* matrices in IGA. Roughly speaking, it requires performing quadrature for only a small fraction of the trial and test basis function interactions and then approximating the rest through, for example, interpolation. This leads to a large sparse matrix where the majority of entries have not been computed using any quadrature at all. Usually, such matrices will not coincide with the ones generated by performing quadrature for every non-zero entry (cf. Section 5.4), but they can be interpreted as surrogates for those matrices.

The main idea used here was first introduced in the context of first-order finite elements by Bauer et al. in [21]. Thereafter, applications to peta-scale geodynamical simulations were presented in [22,23] and a theoretical analysis was given in [24]. In the massively parallel applications [21–23], it was natural to work with so-called "macro-meshes" as well as a piecewise polynomial space for resolving the surrogate matrices. This choice was motivated by a low communication cost across the faces of the macro-elements, a convenient cache-aware implementation, and the fact that a hybrid mesh structure allowed for extremely fast evaluation of the three-dimensional polynomials; see [22,23] for further details. In contrast, the surrogate matrices in this paper are computed using a B-spline interpolation space. With this particular strategy, we demonstrate that the cost of matrix assembly in conventional IGA codes can be reduced by an order of magnitude.

Our approach bears some similarities to the integration by interpolation and lookup (IIL) approach proposed in [13,14]. In those two works, an integrand factor from the weak form, composed of both the coefficients of the underlying PDE as well as the geometry mapping, is approximated. In this work, the actual entries of the final matrix are shown to be related to a small number of smooth so-called *stencil functions*; instead of a factor in the integrand, it is these stencil functions which are approximated.

An advantage of the IIL approach is that, in theory, it does not require a uniform knot vector assumption (cf. Section 2). However, in practice, this assumption is necessary in order to obtain compact lookup tables [13]. On the other hand, one advantage of our approach is that it can be easily implemented using existing IGA assembly paradigms. Another advantage is that the implementation is identical whether using a B-spline or a NURBS basis.

Before moving on, some other important remarks deserve to be emphasized:

- The methodology we propose for IGA applications is essentially independent of the quadrature rule used at the individual element or basis function level. This lays bare the possibility for it to be used in conjunction with many other cutting edge techniques for accelerated IGA assembly.

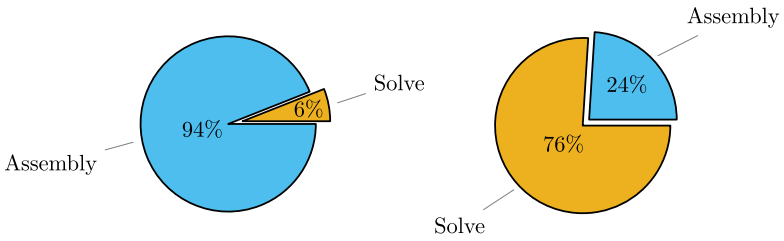


Fig. 1.1. Left: Distribution of computational cost with standard IGA. Right: Distribution of computational cost with a surrogate IGA strategy. (Timings taken from the experiment presented in Figs. 7.5 and 7.6 for $c = 3$. Both experiments ran on a single compute core and the default MATLAB backslash operator has been used as a solver.)

- For our new surrogate methods, it would be most efficient if the matrix entries which require quadrature were to be computed based on individual basis function interactions. This leaves out standard element-by-element assembly strategies, but would work well with the control point pairwise method proposed in [25] or, ideally, with a row-by-row approach; e.g., [9,15,19]. Nevertheless, one should still expect to see significant speed-ups with surrogate methods in standard element-by-element codes, at least for moderate polynomial orders. In order to underscore this fact, we did not develop a stand-alone code. Instead, we implemented our surrogate methods by simply modifying the assembly routines in the open-source library GeoPDEs [26,27], leaving ever other aspect of the code fixed. For illustration, the reader may refer to the left and right sides of Fig. 1.1 to compare the relative timings before and after some relatively minor changes were made to this software (cf. Section 6 and [28]). In both cases, the differences in solving time and solution accuracy were negligible. We expect that most other element-by-element IGA codes should be easy to modify in a similar manner.
- Many efficient assembly strategies for IGA see their performance advantage only in the high polynomial order regime. Here, the performance usually grows with each h-refinement. Indeed, at just over one million degrees of freedom, our experiments demonstrate assembly speed-ups beyond ***fifty times***, in the exact same code, with a simple second-order NURBS basis (see Section 7.3.3).

In our experiments, we analyze surrogate IGA methods for Poisson's equation, membrane vibration, plate bending, and Stokes' flow problems. The Poisson case is analyzed in detail and the additional experiments are provided in order to motivate further study. It is our eventual goal to adapt our methods to a matrix-free framework, similar to what has been used recently in low-order settings [21–24]. This would certainly be helpful in order to reach the full potential of IGA in extreme scale computations.

In the next section, we take stock of the majority of mathematical notation used in the remainder of the paper. In Section 3, we introduce the notion of a stencil function in the IGA context. In Section 4, we investigate the accuracy of B-spline interpolation with regard to stencil functions. In Section 5, we use interpolants of the stencil functions (i.e., surrogate stencil functions) to define surrogate matrices for IGA. Section 6 consists of a brief description of our software implementation. A more complete description is provided in [28]. In Sections 7–10, we examine surrogate IGA methods for Poisson's equation, membrane vibration, plate bending, and Stokes' flow problems, respectively. The Appendix is included to support some of the analysis carried out in Section 4.

2. Preliminaries

In this section, we lay out the principal mathematical focus and notation of the paper.

2.1. Model problems and notation

Let $\Omega \subseteq \mathbb{R}^n$ be a domain, $n = 2, 3$. Let $V = V(\Omega)$ be a Hilbert space over $\mathbb{R}$, the field of real numbers, and let V^* denote its topological dual. For historical reasons, we proceed by adopting notation from the h-version of the finite element method and thus let V_h denote a finite-dimensional subspace of V. Although we also deal with

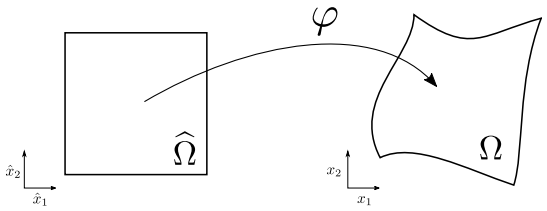


Fig. 2.1. Illustration of a smooth transformation from a parametric domain $\widehat{\Omega} \subseteq \mathbb{R}^2$ to a physical domain $\Omega \subseteq \mathbb{R}^2$.

a number of important alternatives (see, e.g., Sections 8 and 10), we are chiefly interested in the following three problems:

$$\text{Find } u \in V \text{ satisfying} \qquad a(u, v) = F(v) \quad \text{for all } v \in V\,. \tag{2.1a}$$

$$\text{Find } u_h \in V_h \text{ satisfying} \qquad a(u_h, v_h) = F(v_h) \quad \text{for all } v_h \in V_h\,. \tag{2.1b}$$

$$\text{Find } \widetilde{u}_h \in V_h \text{ satisfying} \qquad \widetilde{a}(\widetilde{u}_h, v_h) = F(v_h) \quad \text{for all } v_h \in V_h\,. \tag{2.1c}$$

Here and throughout, $a : V \times V \to \mathbb{R}$ is a continuous and coercive bilinear form, $\widetilde{a} : V_h \times V_h \to \mathbb{R}$ is an approximation of $a|_{V_h \times V_h}$, hereby deemed the *surrogate* for $a(\cdot, \cdot)$, and $F \in V^*$ is a bounded linear functional.

To keep the exposition simple and to the point, we will assume that the physical domain of every problem $\Omega \subseteq \mathbb{R}^n$ is defined as the image of a single parametric domain $\widehat{\Omega} = (0, 1)^n$. This leaves all of our analysis in the single patch geometry setting, $\Omega = \boldsymbol{\varphi}(\widehat{\Omega})$, for some diffeomorphism $\boldsymbol{\varphi} : \widehat{\Omega} \to \mathbb{R}^n$ of sufficient regularity; see Fig. 2.1. The single patch setting is by no means a necessary assumption. The entirety of the analysis considered here can easily be generalized to the multi-patch setting (cf. Section 3.5). However, in order to stay in the isogeometric setting, we assume that $\boldsymbol{\varphi}(\widehat{\boldsymbol{x}}) = \sum_i \mathsf{c}_i \widehat{N}_i(\widehat{\boldsymbol{x}})$, where each $\mathsf{c}_i \in \mathbb{R}^n$ is a control point vector and each $\widehat{N}_i(\widehat{\boldsymbol{x}})$ is a NURBS basis function on the parametric domain $\widehat{\Omega}$. Here, NURBS basis functions are defined in the standard way, as described in Section 2.2.

For matrices $\mathsf{M} \in \mathbb{R}^{l \times m}$, define the max-norm, $\|\mathsf{M}\|_{\max} = \max_{i,j} |\mathsf{M}_{ij}|$. For any function $v : \Omega \to \mathbb{R}$, we will use the notation, $\|v\|_0$, $\|v\|_1$, and $\|v\|_2$, for the canonical $L^2(\Omega)$-, $H^1(\Omega)$-, and $H^2(\Omega)$-norms, respectively. Moreover, if v is smooth, we define its support as $\operatorname{supp}(v) = \{\boldsymbol{x} \in \Omega : v(\boldsymbol{x}) \neq 0\}$. When dealing with a domain $\mathcal{D} \subseteq \Omega$, denote the related $L^2(\mathcal{D})$, $H^1(\mathcal{D})$, and $H^2(\mathcal{D})$ norms by $\|v\|_{0,\mathcal{D}}$, $\|v\|_{1,\mathcal{D}}$, and $\|v\|_{2,\mathcal{D}}$, respectively. Denote the space of univariate polynomials of degree at most q by $\mathcal{P}_q$. Likewise, denote the space of multivariate polynomials of degree at most q, in each Cartesian direction $\boldsymbol{e}_i$, by $\mathcal{Q}_q = [\mathcal{P}_q]^n$ and denote $\mathcal{Q}_q(\mathcal{D}) = \{f|_{\mathcal{D}} : f \in \mathcal{Q}_q\}$. We will often deal with Cartesian subdomains $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$. In this case, it is natural to deal with Cartesian-Sobolev seminorms; e.g., $[f]_{W^{r,\infty}(\mathcal{D})} = \sum_{i=1}^n \|D^{r \cdot \boldsymbol{e}_i} f\|_{L^\infty(\mathcal{D})}$. Note that $[f]_{W^{r,\infty}(\mathcal{D})} \leq |f|_{W^{r,\infty}(\mathcal{D})} = \sum_{|\boldsymbol{\alpha}|=r} \|D^{\boldsymbol{\alpha}} f\|_{L^\infty(\mathcal{D})}$. All remaining notation will be defined as it arises.

2.2. Cardinal B-splines and NURBS

Let $m \geq 2p+1$ and $\{b_k\}_{k=1}^m$ be an order p B-spline basis on the unit interval $(0, 1)$. Let $N = m^n$ and let $\{\widehat{N}_i\}_{i=1}^N$ be a corresponding NURBS basis on $\widehat{\Omega}$. Namely,

$$\widehat{N}_i(\widehat{\boldsymbol{x}}) = \frac{w_i \widehat{B}_i(\widehat{\boldsymbol{x}})}{\sum_j w_j \widehat{B}_j(\widehat{\boldsymbol{x}})}\,, \qquad \widehat{\boldsymbol{x}} = (\widehat{x}_1, \ldots, \widehat{x}_n) \in \widehat{\Omega}\,, \tag{2.2}$$

where each $\widehat{B}_i(\widehat{\boldsymbol{x}}) = b_{i_1}(\widehat{x}_1) \cdots b_{i_n}(\widehat{x}_n)$ is a multivariate B-spline of uniform order p and each $w_i > 0$ is a fixed weight parameter. Here and from now on, we identify every global index $1 \leq i \leq N$ with a multi-index $\boldsymbol{i} = (i_1, \ldots, i_n)$, $1 \leq i_k \leq m$, through the colexicographical relationship $i = i_1 + (i_2 - 1)m + \cdots + (i_n - 1)m^{n-1}$.

Generally, a univariate B-spline basis $\{b_k\}_{k=1}^m$ is defined by an ordered multiset, or *knot vector*, $\Xi = \{\xi_1, \ldots, \xi_{m+p+1}\}$. In this paper, we deal only with *open uniform knot vectors*; i.e., $\xi_1, \ldots, \xi_{p+1} = 0$, $\xi_{m+1}, \ldots, \xi_{m+p+1} = 1$, and $\xi_{k+1} - \xi_k = \frac{1}{m-p}$, otherwise. The quantity $h = \max_{1 \leq k \leq m-1} |\xi_{k+1} - \xi_k| = \frac{1}{m-p}$ will be an important parameter for us, which we hereby refer to as the *mesh size*. Clearly, we could consider NURBS spaces with

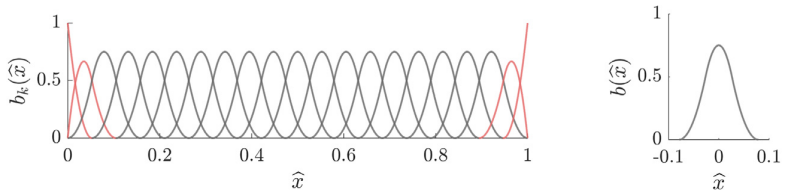


Fig. 2.2. Left: 1D B-spline basis functions $\{b_k\}_{k=1}^{21}$ generated by the open uniform knot vector $\Xi^{(2)}$, defined in (2.3). Right: Each gray basis function is equivalent, up to translation, to the function $b(\widehat{x})$. The red functions are obviously not. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

different orders $p_1, \ldots, p_n$ in each Cartesian direction [3,29]. In order to simplify the exposition, we avoid this complication.

For an explicit example of an open uniform knot vector, consider

$$\Xi^{(p)} = \{\underbrace{0, \ldots, 0}_{p \text{ times}}\} \cup \left\{\frac{k}{19}\right\}_{k=0}^{19} \cup \{\underbrace{1, \ldots, 1}_{p \text{ times}}\}. \tag{2.3}$$

The corresponding $p = 2$ B-spline basis is depicted in Fig. 2.2. Observe that all but four of the basis functions (highlighted in red) are identical, up to an equally spaced set of translations. These functions are called *cardinal B-splines* [30–32].

Let $\widetilde{x}^{(k)} = (k - \frac{p+1}{2}) \cdot h$, for each $k = p+1, \ldots, m-p$. In general, there are always $m - 2p$ univariate cardinal B-spline basis functions which can each be expressed $b_k(\widehat{x}) = b(\widehat{x} - \widetilde{x}^{(k)})$, for some function, $b(\widehat{x})$, centered at the origin with support in $(-\frac{p+1}{2} \cdot h, \frac{p+1}{2} \cdot h)$ (see, e.g., Fig. 2.2). The $\widetilde{x}^{(k)}$ correspond to the midpoints of each function b_k. Likewise, there are $(m - 2p)^n$ multivariate cardinal B-splines. That is $\widehat{B}_i(\widehat{\boldsymbol{x}}) = \widehat{B}(\widehat{\boldsymbol{x}} - \widetilde{\boldsymbol{x}}_i)$, where $\widetilde{\boldsymbol{x}}_i = \left(\widetilde{x}^{(i_1)}, \ldots, \widetilde{x}^{(i_n)}\right)$ and $\widehat{B}(\widehat{\boldsymbol{x}}) = b(\widehat{x}_1) \cdots b(\widehat{x}_n)$. For future reference, define the set of all such $\widetilde{\boldsymbol{x}}_i$ as $\widetilde{\mathbb{X}}$. Also, notice that the ratio of cardinal B-splines basis functions to total B-spline basis functions quickly tends to unity, $\left(\frac{m-2p}{m}\right)^n \to 1$, as m increases.

3. Surrogate matrices: exploiting basis structure

Eqs. (2.1b) and (2.1c), respectively, induce two related matrix equations,

$$\mathsf{A}\mathsf{u} = \mathsf{f} \qquad \text{and} \qquad \widetilde{\mathsf{A}}\widetilde{\mathsf{u}} = \mathsf{f}, \tag{3.1}$$

for basis function coefficients $\mathsf{u}, \widetilde{\mathsf{u}} \in \mathbb{R}^N$. As mentioned previously, the key idea in this paper is constructing the majority of the surrogate stiffness matrix $\widetilde{\mathsf{A}}$ via *interpolation* of the true stiffness matrix A. In this section, we first describe exactly what is meant by this statement and then demonstrate how isogeometric analysis makes it possible.

3.1. Stencil functions

Recall (2.1b). Generally, every function $v_h \in V_h$ can be identified with a unique function on the domain $\widehat{\Omega}$ through a suitable pushforward operator $\boldsymbol{\varphi}_*$. Namely, $v_h = \boldsymbol{\varphi}_* \widehat{v}_h$. Define $\widehat{V}_h$ be the set of all such $\widehat{v}_h$, which is a discrete space in the parametric domain $\widehat{\Omega}$. Accordingly, the bilinear form $a : V_h \times V_h \to \mathbb{R}$ can be identified with a parametric domain bilinear form $\widehat{a} : \widehat{V}_h \times \widehat{V}_h \to \mathbb{R}$ in such a way that $a(w_h, v_h) = \widehat{a}(\widehat{w}_h, \widehat{v}_h)$, for all $\widehat{w}_h, \widehat{v}_h \in \widehat{V}_h$.

Let $\{\phi_i\} = \{\boldsymbol{\varphi}_* \widehat{\phi}_i\}$ be a basis for V_h with $\{\widehat{\phi}_i\}$ the corresponding basis for $\widehat{V}_h$. The fundamental observation in the surrogate matrix methodology now follows. If, $\widehat{\phi}$ is some fixed reference function and, for a set of indices i, j, $\widehat{\phi}_i(\widehat{\boldsymbol{x}}) = \widehat{\phi}(\widehat{\boldsymbol{x}} - \widetilde{\boldsymbol{x}}_i)$ and $\widehat{\phi}_j(\widehat{\boldsymbol{x}}) = \widehat{\phi}(\widehat{\boldsymbol{x}} - \widetilde{\boldsymbol{x}}_j)$, then

$$\widehat{a}(\widehat{\phi}_j, \widehat{\phi}_i) = \widehat{a}(\widehat{\phi}(\cdot - \widetilde{\boldsymbol{x}}_j), \widehat{\phi}(\cdot - \widetilde{\boldsymbol{x}}_i)) := \Phi(\widetilde{\boldsymbol{x}}_j, \widetilde{\boldsymbol{x}}_i). \tag{3.2}$$

Here, in the rightmost equality, the definition of a new scalar-valued function $\Phi(\cdot, \cdot)$ has been made, wherein any dependence on the mesh size h has been implicitly assumed. This function, $\Phi(\widetilde{\boldsymbol{x}}_j, \widetilde{\boldsymbol{x}}_i)$, may also be expressed in

terms of $\widetilde{\boldsymbol{x}}_i$ and a translation $\boldsymbol{\delta} = \widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{x}}_i$. In this alternative characterization, after denoting $\mathsf{A}_{ij} = a(\phi_j, \phi_i)$, we may write

$$\mathsf{A}_{ij} = \Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i) \tag{3.3}$$

and $\boldsymbol{\delta}$ may be treated as a parameter. We define $\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i) = \Phi(\widetilde{\boldsymbol{x}}_i + \boldsymbol{\delta}, \widetilde{\boldsymbol{x}}_i) = \Phi(\widetilde{\boldsymbol{x}}_j, \widetilde{\boldsymbol{x}}_i)$.

For a fixed number of translations $\boldsymbol{\delta}$, these so-called *stencil functions*, $\Phi_{\boldsymbol{\delta}}(\cdot)$, can be identified with the majority of entries in many IGA stiffness matrices. In many circumstance, each $\Phi_{\boldsymbol{\delta}}(\cdot)$ is smooth and may, therefore, be interpolated after only being evaluated at small number of points in the parametric domain $\widetilde{\boldsymbol{x}}_i \in \widetilde{\Omega}$. After denoting the interpolants — i.e., the *surrogate stencil functions* — by $\widetilde{\Phi}_{\boldsymbol{\delta}}(\cdot)$, we simply define

$$\widetilde{\mathsf{A}}_{ij} = \widetilde{\Phi}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i)\,. \tag{3.4}$$

Remark 3.1. In some cases, the stencil functions $\Phi_{\boldsymbol{\delta}}$ are themselves polynomials (see, e.g., Proposition 5.1 and Corollaries 8.1 and 10.1). Therefore, if polynomial interpolation of sufficiently high order is used, the true stiffness matrix be generated exactly — i.e., $\widetilde{\Phi}_{\boldsymbol{\delta}} = \Phi_{\boldsymbol{\delta}}$ and thus $\widetilde{\mathsf{A}} = \mathsf{A}$, up to round-off error — in significantly less time than with a traditional assembly algorithm. Otherwise, in many scenarios, a sufficiently accurate approximation of the stiffness matrix $\widetilde{\mathsf{A}} \approx \mathsf{A}$ will be generated.

3.2. B-spline basis functions

Fix $V = H^1(\Omega)$, $\boldsymbol{\varphi} : \widehat{\Omega} \to \Omega$, $\widehat{V}_h = \operatorname{span}\{\widehat{B}_i\}$, and, accordingly, $V_h = \operatorname{span}\{B_i\}$, where each $B_i = \widehat{B}_i \circ \boldsymbol{\varphi}^{-1}$. Consider the bilinear form $a(u, v) = \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x$. It is easy to verify that $a(\cdot,\cdot)$ pulls back to

$$\widehat{a}(\widehat{w}, \widehat{v}) = \int_{\widehat{\Omega}} \widehat{\nabla}\widehat{w}(\widehat{\boldsymbol{x}})^\top K(\widehat{\boldsymbol{x}})\, \widehat{\nabla}\widehat{v}(\widehat{\boldsymbol{x}})\,\mathrm{d}\widehat{\boldsymbol{x}}\,, \qquad \text{where} \qquad K = \frac{D\boldsymbol{\varphi}^{-1}\, D\boldsymbol{\varphi}^{-\top}}{|\det\left(D\boldsymbol{\varphi}^{-1}\right)|}\,, \tag{3.5}$$

with arguments $\widehat{w}, \widehat{v} \in \widehat{V} = H^1(\widehat{\Omega})$.

Recall Section 2.2. Assume that $\{\widehat{B}_i\}$ is generated by an open uniform knot vector Ξ with $p > 0$ fixed. Obviously, $\mathsf{A}_{ij} = \widehat{a}(\widehat{B}_j, \widehat{B}_i)$. In the cardinal B-spline setting, $\widehat{B}_i(\widehat{\boldsymbol{x}}) = \widehat{B}(\widehat{\boldsymbol{x}} - \widetilde{\boldsymbol{x}}_i)$ and $\widehat{B}_j(\widehat{\boldsymbol{x}}) = \widehat{B}(\widehat{\boldsymbol{x}} - \widetilde{\boldsymbol{x}}_j)$. Therefore, by a simple change of variables,

$$\mathsf{A}_{ij} = \int_{\widehat{\Omega}} \widehat{\nabla}\widehat{B}(\widehat{\boldsymbol{x}} - \widetilde{\boldsymbol{x}}_i)^\top K(\widehat{\boldsymbol{x}})\, \widehat{\nabla}\widehat{B}(\widehat{\boldsymbol{x}} - \widetilde{\boldsymbol{x}}_j)\,\mathrm{d}\widehat{\boldsymbol{x}} = \int_{\widehat{\omega}_{\boldsymbol{\delta}}} \widehat{\nabla}\widehat{B}(\widehat{\boldsymbol{y}})^\top K(\widetilde{\boldsymbol{x}}_i + \widehat{\boldsymbol{y}})\, \widehat{\nabla}\widehat{B}_{\boldsymbol{\delta}}(\widehat{\boldsymbol{y}})\,\mathrm{d}\widehat{\boldsymbol{y}}\,, \tag{3.6}$$

where $\boldsymbol{\delta} = \widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{x}}_i$, $\widehat{B}_{\boldsymbol{\delta}}(\widehat{\boldsymbol{y}}) = \widehat{B}(\widehat{\boldsymbol{y}} - \boldsymbol{\delta})$, and $\widehat{\omega}_{\boldsymbol{\delta}} = \operatorname{supp}(\widehat{B}) \cap \operatorname{supp}(\widehat{B}_{\boldsymbol{\delta}})$.

There is a natural correspondence between the density of the matrix A and the set of translations $\boldsymbol{\delta}$ such that $\widehat{\omega}_{\boldsymbol{\delta}} \neq \emptyset$. Consequently, the cardinality of the set of relevant translations, $\mathcal{D} = \{\boldsymbol{\delta} = \widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{x}}_i \,:\, \widehat{\omega}_{\boldsymbol{\delta}} \neq \emptyset\}$, is fixed for all sufficiently large m. Namely, $|\mathcal{D}| = (2p+1)^n$. Now, for each $\boldsymbol{\delta} \in \mathcal{D}$, we may define the stencil function

$$\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}) = \int_{\widehat{\omega}_{\boldsymbol{\delta}}} \widehat{\nabla}\widehat{B}(\widehat{\boldsymbol{y}})^\top K(\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}})\, \widehat{\nabla}\widehat{B}_{\boldsymbol{\delta}}(\widehat{\boldsymbol{y}})\,\mathrm{d}\widehat{\boldsymbol{y}}\,. \tag{3.7}$$

Let $\operatorname{conv}(\widetilde{\mathbb{X}})$ denote the convex hull of $\widetilde{\mathbb{X}}$. Such functions are defined at any point $\widetilde{\boldsymbol{x}} \in \operatorname{conv}(\widetilde{\mathbb{X}})$ where $\widetilde{\boldsymbol{x}} + \boldsymbol{\delta} \in \operatorname{conv}(\widetilde{\mathbb{X}})$. Ultimately, this means that the domain of $\Phi_{\boldsymbol{\delta}}$, which we will denote by $\widetilde{\Omega}_{\boldsymbol{\delta}}$, depends on $\boldsymbol{\delta}$ (see, e.g., Fig. 3.1). Clearly, we always have $\mathbf{0} \in \mathcal{D}$. The reader may easily verify that $\widetilde{\Omega}_{\mathbf{0}} = \operatorname{conv}(\widetilde{\mathbb{X}}) = \bigcup_{\boldsymbol{\delta}\in\mathcal{D}} \widetilde{\Omega}_{\boldsymbol{\delta}}$ and $\widetilde{\Omega}_{\boldsymbol{\delta}} + \boldsymbol{\delta} \subseteq \widetilde{\Omega}_{\mathbf{0}}$, for each $\boldsymbol{\delta} \in \mathcal{D}$.

3.3. NURBS basis functions

The principal difference between the treatment of a NURBS basis $\{\widehat{N}_i\}$ and the related B-spline basis $\{\widehat{B}_i\}$ is that a NURBS basis cannot be assumed to have the translation invariance property which leads directly to (3.2). Fortunately, as we now demonstrate, this property is not entirely necessary to define a useful stencil function.

Define $W(\widehat{\boldsymbol{x}}) = \sum_j w_j \widehat{B}_j(\widehat{\boldsymbol{x}})$, where $\{w_j\}$ are the weight parameters appearing in (2.2). It is known that $W(\widehat{\boldsymbol{x}})$ is unchanged under mesh refinements. Therefore, employing a similar change of variables argument as used in (3.6), it holds that

$$\mathsf{A}_{ij} = \widehat{a}(\widehat{N}_j, \widehat{N}_i) = w_i w_j \int_{\widehat{\omega}_{\boldsymbol{\delta}}} \widehat{\nabla}\left(\frac{\widehat{B}(\widehat{\boldsymbol{y}})}{W(\widetilde{\boldsymbol{x}}_i + \widehat{\boldsymbol{y}})}\right)^\top K(\widetilde{\boldsymbol{x}}_i + \widehat{\boldsymbol{y}})\, \widehat{\nabla}\left(\frac{\widehat{B}_{\boldsymbol{\delta}}(\widehat{\boldsymbol{y}})}{W(\widetilde{\boldsymbol{x}}_i + \widehat{\boldsymbol{y}})}\right)\mathrm{d}\widehat{\boldsymbol{y}}\,, \tag{3.8}$$

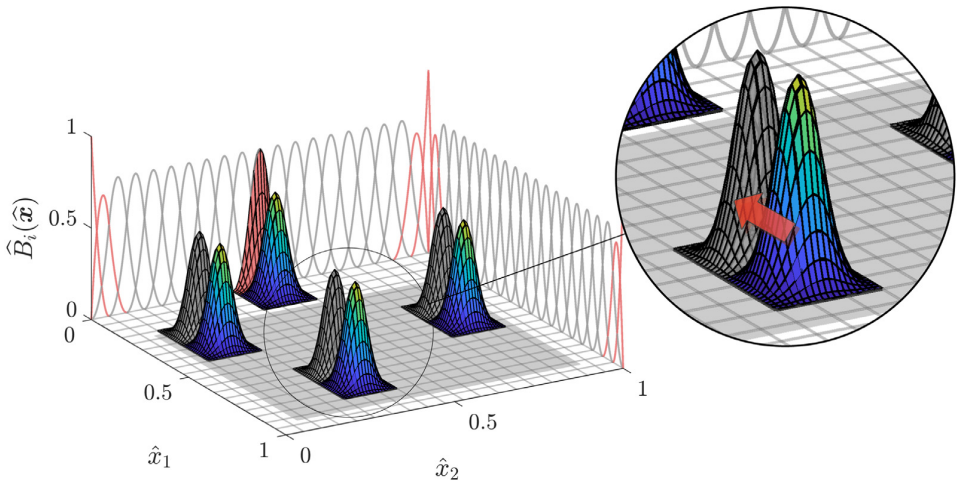


Fig. 3.1. Second-order cardinal B-spline basis functions $\widehat{B}_i(\widehat{\boldsymbol{x}}) = \widehat{B}(\widehat{\boldsymbol{x}} - \widetilde{\boldsymbol{x}}_i)$ (color gradient). After translation by $\boldsymbol{\delta} = \binom{-2h}{0}$ (represented by the red arrow), most of these basis functions are equal to another cardinal basis function $\widehat{B}_j(\widehat{\boldsymbol{x}}) = \widehat{B}_i(\widehat{\boldsymbol{x}} - \boldsymbol{\delta})$ (gray). For every such function, $\widetilde{\boldsymbol{x}}_i \in \widetilde{\Omega}_{\boldsymbol{\delta}}$ (this subset is shaded in gray on the mesh). Clearly, this property does not hold for every cardinal basis function, as indicated by the cardinal B-spline neighboring the boundary and the nearby (non-cardinal) basis function (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where, as in (3.6), $\boldsymbol{\delta} = \widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{x}}_i$. At this point, it may be natural to divide by $w_i w_j$ and define the stencil function $\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}})$ from the resulting expression on the right-hand side of (3.8). Instead, we pause to consider the regularity of $W(\widetilde{\boldsymbol{x}})$.

Recall (3.4). Since each $\widehat{B}_i(\widehat{\boldsymbol{x}})$ is only piecewise polynomial, it is clear that, in general, $W \notin C^q(\widehat{\Omega})$, for any $q \geq p$. This fact could significantly limit the accuracy of an interpolant $\widetilde{\Phi}_{\boldsymbol{\delta}} = \Pi\Phi_{\boldsymbol{\delta}}$ which we may wish to construct. Therefore, we restrict our attention to $W \in C^q(\widehat{\Omega})$, where $q \geq p$. It turns out that this set of functions, $\operatorname{span}\{\widehat{B}_i\} \cap C^q(\widehat{\Omega})$, is equal to the polynomial space $\mathcal{Q}_p(\widehat{\Omega})$. Moreover, restricting to the subset $\widetilde{\Omega}_{\mathbf{0}}$, each weight parameter can be expressed as $w_i = w(\widetilde{\boldsymbol{x}}_i)$, where $w(\widehat{\boldsymbol{x}})$ is a polynomial in $\mathcal{Q}_p(\widetilde{\Omega}_{\mathbf{0}})$. (See the Appendix for details.) Therefore, for any $W \in \mathcal{Q}_p(\widehat{\Omega})$, we may define

$$\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}) = w(\widetilde{\boldsymbol{x}}) w(\widetilde{\boldsymbol{x}} + \boldsymbol{\delta}) \int_{\widehat{\omega}_{\boldsymbol{\delta}}} \widehat{\nabla}\left(\frac{\widehat{B}(\widehat{\boldsymbol{y}})}{W(\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}})}\right)^{\top} K(\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}})\, \widehat{\nabla}\left(\frac{\widehat{B}_{\boldsymbol{\delta}}(\widehat{\boldsymbol{y}})}{W(\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}})}\right) \mathrm{d}\widehat{\boldsymbol{y}}, \tag{3.9}$$

for each $\widetilde{\boldsymbol{x}} \in \widetilde{\Omega}_{\boldsymbol{\delta}}$ and $\boldsymbol{\delta} \in \mathcal{D}$. Clearly, $\mathsf{A}_{ij} = \Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i)$ for each corresponding $\boldsymbol{\delta} \in \mathcal{D}$. An illustration of a stencil function coming from an IGA discretization with a NURBS basis is presented in Fig. 3.2.

Remark 3.2. Notably, when $W = 1$, then $w = 1$ also. Therefore, (3.9) is consistent with (3.7). Obviously, in practice, neither of these expressions needs to be used in order to evaluate $\Phi_{\boldsymbol{\delta}}(\cdot)$ at any point $\widetilde{\boldsymbol{x}}_i$. Indeed, since $\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i) = \mathsf{A}_{ij}$, any existing IGA code already has a quadrature mechanism to compute $\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i)$; cf. Section 6. Nevertheless, these expressions are important for analysis.

3.4. Symmetric bilinear forms

When $a(w, v) = a(v, w)$, for all $w, v \in V$, a translational symmetry is induced on the set of corresponding stencil functions. Indeed, it is simple to see that $\Phi(\widetilde{\boldsymbol{x}}_j, \widetilde{\boldsymbol{x}}_i) = \Phi(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j)$ and, therefore,

$$\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i) = \Phi(\widetilde{\boldsymbol{x}}_i + \boldsymbol{\delta}, \widetilde{\boldsymbol{x}}_i) = \Phi(\widetilde{\boldsymbol{x}}_j - \boldsymbol{\delta}, \widetilde{\boldsymbol{x}}_j) = \Phi_{-\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_j)\,.$$

A similar conclusion can be drawn in the NURBS scenario above. Fig. 3.3 presents a visual comparison of two stencil functions, $\Phi_{\boldsymbol{\delta}}$ and $\Phi_{-\boldsymbol{\delta}}$, generated by an isogeometric NURBS basis and the corresponding symmetric bilinear form (3.5).

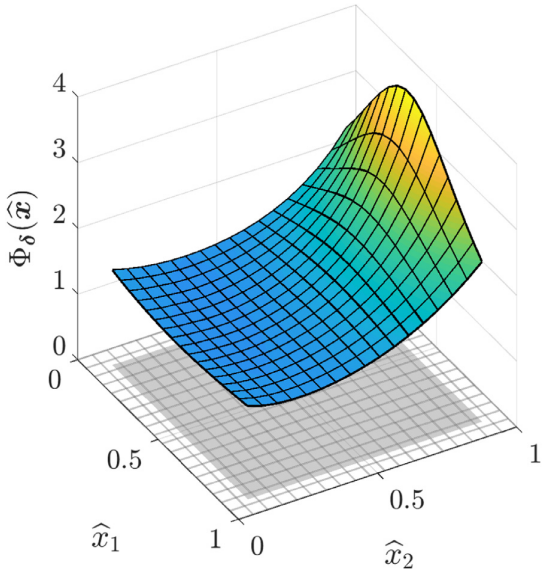


Fig. 3.2. The graph of a stencil function $\Phi_\delta : \widetilde{\Omega}_\delta \to \mathbb{R}$ defined by (3.9). The corresponding physical geometry Ω is depicted in Fig. 7.1(a). In this case, $\delta = \mathbf{0} \cdot h$ and $p = 2$. The domain $\widetilde{\Omega}_\mathbf{0}$ is shaded gray. For further details, see Section 7.3.

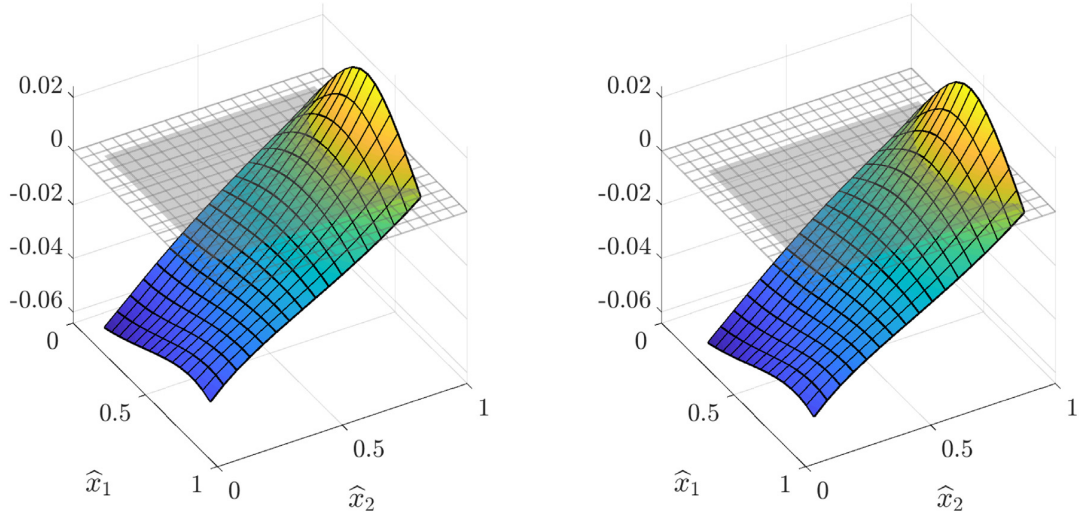


Fig. 3.3. Left and right, respectively: Graphs of stencil functions $\Phi_\delta(\widetilde{x})$ for $\delta = \binom{2h}{0}$ and $\delta = \binom{-2h}{0}$. The associated domains $\widetilde{\Omega}_\delta$ are shaded in gray (cf. Fig. 3.1). Notice the translational symmetry $\Phi_\delta(\widetilde{x}) = \Phi_{-\delta}(\widetilde{x} + \delta)$.

3.5. The multi-patch setting

In the multi-patch setting, the physical domain Ω is partitioned into a finite number of disjoint subdomains $\overline{\Omega} = \bigcup_{k=1}^{L} \overline{\Omega^{(k)}}$. We may assume that each such domain or *patch* $\Omega^{(k)}$, as they are usually called, can be identified with a common parametric domain $\widehat{\Omega}$, through a unique NURBS mapping $\Omega^{(k)} = \boldsymbol{\varphi}^{(k)}(\widehat{\Omega})$. In this case, one may define a separate set of stencil functions $\Phi_\delta^{(k)}(\widetilde{x}) : \widetilde{\Omega}_\delta \to \mathbb{R}$, for each index k. The forthcoming analysis is immediately applicable to this setting, but treating it outright would require unnecessarily complicated notation. We also did not consider this setting in our numerical experiments.

3.6. Edge-based and face-based stencil functions

It is also possible to define stencil functions, in addition to those above, which exploit lower-dimensional translational symmetries. For instance, many functions in the basis $\{\widehat{N}_i\}$ are only equivalent under translations parallel to a given edge in the parametric domain $\widehat{\Omega}$. Likewise, a new family of stencil functions could be constructed for each edge or face in a patch $\Omega^{(k)}$. Edge-/face-based stencil functions are not considered in this work.

4. Surrogate matrices: interpolation of stencil functions

Recall (3.3) and (3.4). A surrogate matrix $\widetilde{\mathsf{A}} \approx \mathsf{A}$ will be useful to us if: (1) each stencil function $\Phi_{\boldsymbol{\delta}}$ has high regularity and, therefore, the high-order approximation $\Pi\Phi_{\boldsymbol{\delta}} \approx \Phi_{\boldsymbol{\delta}}$ will have high-order accuracy; and (2) each $\widetilde{\Phi}_{\boldsymbol{\delta}} = \Pi\Phi_{\boldsymbol{\delta}}$ can be computed and evaluated fast. In this section, we focus on the former of these two requirements. Particulars on our implementation are withheld until Section 6.

As a simplifying accommodation, we perform all of our analysis on the largest subset of $\widehat{\Omega}$ where every stencil function is defined. That is, $\widetilde{\Omega} = \bigcap_{\boldsymbol{\delta}\in\mathcal{D}} \widetilde{\Omega}_{\boldsymbol{\delta}}$. A simple computation shows that

$$\widetilde{\Omega} = \left[\frac{3p+1}{2(m-p)}, 1 - \frac{3p+1}{2(m-p)}\right]^n \subsetneq \widehat{\Omega}\,.$$

All of the results in this section can be reformulated with $\widetilde{\Omega}$ replaced by $\widetilde{\Omega}_{\boldsymbol{\delta}}$. However, this generalization also requires a different operator Π for each $\boldsymbol{\delta} \in \mathcal{D}$; see, e.g., [24].

4.1. B-spline interpolation

Let $\{\widetilde{B}_j\}$ be a degree $q \geq 0$ multivariate B-spline basis on $\widetilde{\Omega}$ with the quasi-uniform knot vector $\widetilde{\boldsymbol{\Xi}} = \widetilde{\Xi}_1 \times \cdots \times \widetilde{\Xi}_n$ and define $S_q(\widetilde{\boldsymbol{\Xi}}) = \operatorname{span}\{\widetilde{B}_j\}$. We will refer to each $\widetilde{\boldsymbol{\xi}}_{\boldsymbol{j}} \in \widetilde{\boldsymbol{\Xi}}$, $\boldsymbol{j} = (j_1, \ldots, j_n)$, as a *sampling point* and define a new length scale parameter, hereby referred to as the *sampling length*, $H = \max_{|\boldsymbol{j}|=1,\boldsymbol{i}} \{\|\widetilde{\boldsymbol{\xi}}_{\boldsymbol{i}+\boldsymbol{j}} - \widetilde{\boldsymbol{\xi}}_{\boldsymbol{i}}\|_{\max} : \widetilde{\boldsymbol{\xi}}_{\boldsymbol{i}+\boldsymbol{j}} \in \widetilde{\boldsymbol{\Xi}}\}$.

In isogeometric analysis, the geometry determines the basis used in PDE discretization. When constructing the surrogate stencil functions $\widetilde{\Phi}_{\boldsymbol{\delta}}$, we are primarily interested in using a basis of higher order q than the underlying spatial discretization p. Generally, such a choice $q > p$ is desirable because it will allow us to guarantee that the discretization error in a standard IGA method will dominate the error actually attributed to using a surrogate (cf. Section 7.3.2).

In this paper, we only consider constructing B-spline interpolants $\widetilde{\Phi}_{\boldsymbol{\delta}} = \Pi_H \Phi_{\boldsymbol{\delta}}$, where Π_H is a stable local interpolation operator onto the space $S_q(\widetilde{\boldsymbol{\Xi}})$. Various global interpolants could also be considered, as well as sparse grid interpolants [33] and least-squares projections (cf. [24]). We see no benefit in using a NURBS basis to approximate $\Phi_{\boldsymbol{\delta}}$, even when a NURBS mapping $\boldsymbol{\varphi} : \widehat{\Omega} \to \Omega$ defines the physical domain. The following lemma follows directly from [34, Theorem 4.2].

Lemma 4.1. *For every bounded projection $\Pi_H \colon C^0(\widetilde{\Omega}) \to S_q(\widetilde{\boldsymbol{\Xi}})$, with $\|\Pi_H\| < C_0$ for some H-independent constant C_0, it holds that*

$$\|f - \Pi_H f\|_{L^\infty(\widetilde{\Omega})} \leq C_1 H^{q+1} [f]_{W^{q+1,\infty}(\widetilde{\Omega})}\,, \quad \text{for all } f \in W^{q+1,\infty}(\widetilde{\Omega})\,,$$

where C_1 is a constant depending only on q, $\widetilde{\Omega}$, and $\|\Pi_H\|$.

Fig. 4.1 presents graphs of stencil functions, $\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}})$ and their surrogates $\widetilde{\Phi}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}})$ obtained by B-spline interpolation for various $\boldsymbol{\delta}$. The stencil functions are generated by an isogeometric NURBS basis and the symmetric bilinear form (3.5).

Remark 4.1. The norm $\|\Pi_H\|$ can be greatly influenced by the distribution of the sampling points $\widetilde{\boldsymbol{\xi}}_{\boldsymbol{j}} \in \widetilde{\boldsymbol{\Xi}}$. In all of our experiments, we kept $\widetilde{\boldsymbol{\Xi}} \subseteq \widetilde{\Omega} \cap \widetilde{\mathbb{X}}$. This convenient choice delivered good results. When we wish to underscore the convention $\widetilde{\boldsymbol{\Xi}} \subseteq \widetilde{\Omega} \cap \widetilde{\mathbb{X}}$, we will denote the sampling points in $\widetilde{\boldsymbol{\Xi}}$ by $\widetilde{\boldsymbol{x}}_i^{\mathrm{s}}$. Moreover, from now on, dependence on the subset $\widetilde{\Omega}$ will not be stated since $\widetilde{\Omega} \to \widehat{\Omega}$ as $h \to 0$ and the parametric domain $\widehat{\Omega} = (0,1)^n$ is always fixed.

4.2. Regularity of the stencil functions

Define $\widetilde{\Phi}_{\boldsymbol{\delta}} = \Pi_H \Phi_{\boldsymbol{\delta}}$, where Π_H is any projection satisfying the assumptions of Lemma 4.1. In this subsection, we present an essential theorem on the error in the class of surrogate stencil functions defined in (3.9). In order to expedite our presentation, we only prove Theorem 4.2 here under an assumption which directly relates to the B-spline basis scenario (3.7). The general proof is given in the Appendix.

Theorem 4.2. *Let $\Phi_{\boldsymbol{\delta}} : \widetilde{\Omega} \to \mathbb{R}$ be defined by (3.9) and assume that $\boldsymbol{\varphi} : \widehat{\Omega} \to \Omega$ induces a coefficient tensor $K \in \left[W^{q+1,\infty}(\widehat{\Omega})\right]^{n\times n}$. If $W \in \mathcal{Q}_p(\widehat{\Omega})$, then there exists a constant C_2, depending only on p, q, $\|\Pi_H\|$, and $\boldsymbol{\varphi}$, such that*

$$\left\| \Phi_{\boldsymbol{\delta}} - \widetilde{\Phi}_{\boldsymbol{\delta}} \right\|_{L^\infty(\widetilde{\Omega})} \leq C_2 h^{n-2} H^{q+1} \quad \text{for each } \boldsymbol{\delta} \in \mathcal{D}\,.$$

Moreover, if $W(\widehat{\boldsymbol{x}}) = 1$, then $C_2 \leq C\, C_1\, [K]_{W^{q+1,\infty}(\widehat{\Omega})}$, for some C depending only on p.

Proof of Theorem 4.2, under the assumption $W(\widehat{\boldsymbol{x}}) = 1$. For every multi-index $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$, where each $0 \leq \alpha_i \leq q+1$, it holds that

$$D^{\boldsymbol{\alpha}} \Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}) = \int_{\widehat{\omega}_{\boldsymbol{\delta}}} \widehat{\nabla} \widehat{B}(\widehat{\boldsymbol{y}})^\top D^{\boldsymbol{\alpha}} K(\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}})\, \widehat{\nabla} \widehat{B}_{\boldsymbol{\delta}}(\widehat{\boldsymbol{y}})\, \mathrm{d}\widehat{\boldsymbol{y}}.$$

Therefore, $\|D^{\boldsymbol{\alpha}} \Phi_{\boldsymbol{\delta}}\|_{L^\infty(\widetilde{\Omega})} \leq \|D^{\boldsymbol{\alpha}} K\|_{L^\infty(\widehat{\Omega})} \|\widehat{\nabla} \widehat{B} \cdot \widehat{\nabla} \widehat{B}_{\boldsymbol{\delta}}\|_{L^1(\widehat{\omega}_{\boldsymbol{\delta}})} \leq C h^{n-2} \|D^{\boldsymbol{\alpha}} K\|_{L^\infty(\widehat{\Omega})}$ and, moreover, $\Phi_{\boldsymbol{\delta}} \in W^{q+1,\infty}(\widetilde{\Omega})$. The result follows from Lemma 4.1 using $f = \Phi_{\boldsymbol{\delta}}$. □

Remark 4.2. Observe that $[K]_{W^{q+1,\infty}(\widehat{\Omega})} = 0$ iff $K \in [\mathcal{Q}_q(\widehat{\Omega})]^{n\times n}$. Generally, due to the definition of K appearing in (3.5), this assumption can only be expected to be satisfied when the geometry map $\boldsymbol{\varphi}$ is affine. Nevertheless, if $W = 1$ as well, then Theorem 4.2 would imply that $\widetilde{\Phi}_{\boldsymbol{\delta}} = \Phi_{\boldsymbol{\delta}} \in \mathcal{Q}_q(\widetilde{\Omega})$. This a useful reproduction property which can help to verify the implementation. It also manifests in stencil functions defined from more complicated bilinear forms than (3.5). The reproduction of stencil functions is described in a general scenario in Section 5.4.

5. Surrogate matrices: preserving structure

In this section, we discuss a number of different surrogate matrix definitions. Under both definitions, we also show that the surrogate matrices may actually fully reproduce the very matrices they approximate. Note that these definitions may also be employed for general matrices and not only for a stiffness matrix. If not stated otherwise, in the following subsections we assume that the matrix A emanates from a discretization of a general bilinear form.

5.1. General surrogate matrices

In Section 3.1, the definition $\widetilde{\mathsf{A}}_{ij} = \widetilde{\Phi}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i)$ was presented, with $\boldsymbol{\delta} = \widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{x}}_i$. However, this is valid exclusively for the special indices $1 \leq i, j \leq N$ with a cardinal B-spline associated to them. Of course, we may always compute the remaining components of the surrogate matrix $\widetilde{\mathsf{A}}$ directly, using element-wise quadrature, as done in standard IGA assembly algorithms. Therefore, we propose the following definition:

$$\widetilde{\mathsf{A}}_{ij} = \begin{cases} \widetilde{\Phi}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i) & \text{if } \widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j \in \widetilde{\Omega}, \\ \mathsf{A}_{ij} & \text{otherwise.} \end{cases} \tag{5.1a}$$

This is a convenient definition which essentially can be used for constructing a surrogate matrix from any general bilinear form $a(\cdot,\cdot)$. Typically, this definition is used for general non-symmetric matrices, e.g., for the divergence matrices B arising in the discretization of Stokes' flow (cf. Section 10). However, (5.1a) can easily be improved when the bilinear form is symmetric or has a well-known kernel.

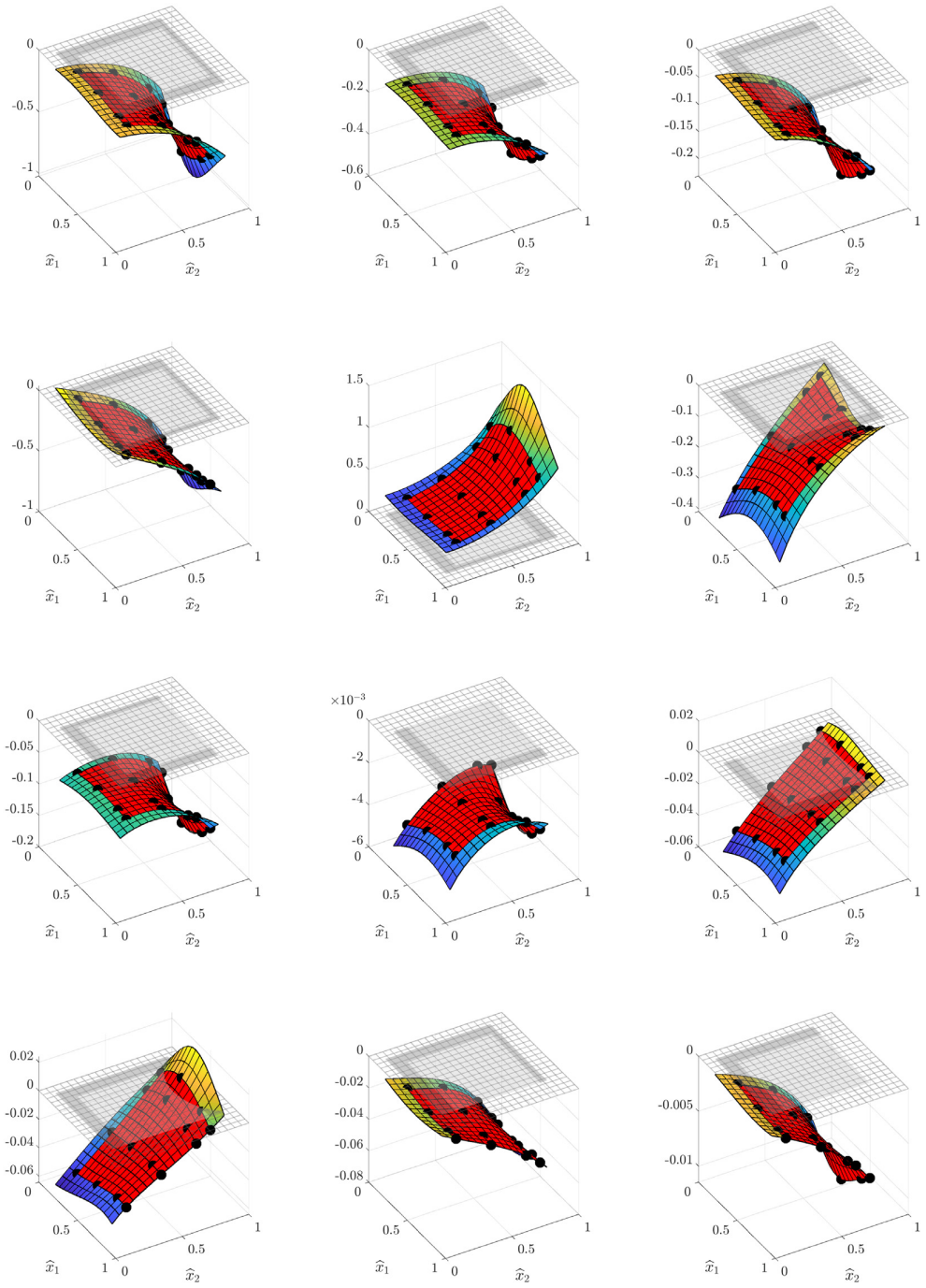


Fig. 4.1. Graphs of the stencil functions $\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}})$ (color gradient) and their surrogates $\widetilde{\Phi}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}})$ (red) for the various $\boldsymbol{\delta}$ required in populating the upper diagonal of the stiffness matrix A_{ij} defined in (5.1). The sampling points $\widetilde{\boldsymbol{x}}_i^{\mathrm{s}} \in \widetilde{\boldsymbol{\Xi}} \subsetneq \widetilde{\Omega} \cap \widetilde{\mathbb{X}}$ used to construct the interpolant are depicted with black dots. The subsets $\widetilde{\Omega} \subseteq \widetilde{\Omega}_{\boldsymbol{\delta}}$ are highlighted in light gray and the remaining subsets $\widetilde{\Omega}_{\boldsymbol{\delta}} \setminus \widetilde{\Omega}$ are depicted in a darker gray. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.2. Symmetry

If the bilinear form is symmetric, (5.1a) does not guarantee that the corresponding surrogate stiffness matrix $\widetilde{\mathsf{A}}$ will be symmetric. In order to enforce symmetry, it is convenient to just include the action of copying A_{ij} into A_{ji}, for all $i > j$. Therefore, we propose the following symmetric surrogate matrix definition:

$$\widetilde{\mathsf{A}}_{ij} = \begin{cases} \widetilde{\Phi}_\delta(\widetilde{\boldsymbol{x}}_i) & \text{if } \widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j \in \widetilde{\Omega} \text{ and } i \le j, \\ \widetilde{\mathsf{A}}_{ji} & \text{if } \widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j \in \widetilde{\Omega} \text{ and } i > j, \\ \mathsf{A}_{ij} & \text{otherwise.} \end{cases} \tag{5.1b}$$

With this definition, note that only $\frac{(2p+1)^n+1}{2}$ surrogate stencil functions need to be computed. We employ this definition in the construction of surrogate mass matrices M which are symmetric but do not have a kernel (cf. Section 8).

5.3. Preserving the kernel

Recall that $\sum N_i(\boldsymbol{x}) = 1$. In the situation $a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x$, we have that $a(1, w) = a(w, 1) = 0$, for all $w \in H^1(\Omega)$. Therefore, $1 \in V_h = \operatorname{span}\{N_i\}$ and, moreover, $\mathsf{A}\mathsf{v}_1 = 0$, where $\mathsf{v}_1 = (1, \dots, 1)^\top$. Clearly, neither definition (5.1a) nor definition (5.1b), will guarantee that $\widetilde{\mathsf{A}}\mathsf{v}_1 = 0$. Therefore, we will pose the following symmetric kernel-preserving definition:

$$\widetilde{\mathsf{A}}_{ij} = \begin{cases} \widetilde{\Phi}_\delta(\widetilde{\boldsymbol{x}}_i) & \text{if } \widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j \in \widetilde{\Omega} \text{ and } i < j, \\ \widetilde{\mathsf{A}}_{ji} & \text{if } \widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j \in \widetilde{\Omega} \text{ and } i > j, \\ \mathsf{A}_{ij} & \text{in all other cases where } i \neq j \\ -\sum_{k \neq i} \widetilde{\mathsf{A}}_{ik} & \text{if } i = j, \end{cases} \tag{5.1c}$$

With this definition, the reader may readily verify that $\widetilde{\mathsf{A}}\mathsf{v}_1 = 0$ and $\widetilde{\mathsf{A}} = \widetilde{\mathsf{A}}^\top$.

Remark 5.1. Three important comments are in order. First, in a matrix-free setting, where memory copying cannot be performed efficiently, one can actually design a set of surrogate stencil functions $\widetilde{\Phi}_\delta$ which preserves the symmetry of the stiffness matrix (see, e.g., [24, Remark 3.4]). Second, in general, it is difficult to generalize the row sum trick used in (5.1c), in a way which preserves symmetry, when the bilinear form $a(\cdot, \cdot)$ has a multi-dimensional kernel; cf. Section 9. Third, mass matrices in isogeometric analysis conserve mass in the sense that $\sum_{i,j} \mathsf{M}_{ij} = \operatorname{vol}(\Omega)$. Under definition (5.1a), there is no reason to expect that the same property holds for the corresponding surrogate mass matrices $\widetilde{M}_{ij}$. Although such a property would be desirable, it does not appear to significantly affect accuracy; cf. Section 8.

5.4. Polynomial reproduction

Until now, we have focused, almost entirely, on the analysis of surrogate stiffness matrices which derive from the bilinear form generated by Poisson's equation. Clearly, the methodology presented above can be applied to other settings as well. The general scenario we are interested in is when $a(\cdot, \cdot)$ in (2.1a) can be expressed in the parametric domain as

$$\widehat{a}(\widehat{w}, \widehat{v}) = \int_{\widehat{\Omega}} G(\widehat{\boldsymbol{x}}, \widehat{w}(\widehat{\boldsymbol{x}}), \widehat{v}(\widehat{\boldsymbol{x}})) \, \mathrm{d}\widehat{\boldsymbol{x}} \qquad \text{for all } \widehat{w}, \widehat{v} \in \widehat{V}, \tag{5.2a}$$

where, for all smooth $\widehat{w}, \widehat{v}$,

$$G(\widehat{\boldsymbol{x}}, \widehat{w}(\widehat{\boldsymbol{y}}), \widehat{v}(\widehat{\boldsymbol{y}})) = 0, \qquad \text{whenever } \widehat{\boldsymbol{y}} \notin \operatorname{supp}(\widehat{w}) \cap \operatorname{supp}(\widehat{v}) \, . \tag{5.2b}$$

We now consider the general coefficient matrix A and a cardinal B-spline basis $\{\widehat{B}_i\}$ (cf. Section 3.2). Invoking (5.2b), a simple change of variables leads us to

$$\mathsf{A}_{ij} = \widehat{a}(\widehat{B}_j, \widehat{B}_i) = \int_{\widehat{\omega}_\delta} G(\widetilde{\boldsymbol{x}}_i + \widehat{\boldsymbol{y}}, \widehat{B}_\delta(\widehat{\boldsymbol{y}}), \widehat{B}(\widehat{\boldsymbol{y}})) \, \mathrm{d}\widehat{\boldsymbol{y}} \, ,$$

where, as before, $\boldsymbol{\delta} = \widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{x}}_i$ and $\widehat{\omega}_{\boldsymbol{\delta}} = \operatorname{supp}(\widehat{B}) \cap \operatorname{supp}(\widehat{B}_{\boldsymbol{\delta}})$. Using the techniques put forth in Section 3.3, this expression can easily be generalized for cardinal NURBS bases with polynomial weight functions $W(\widehat{\boldsymbol{x}})$. However, considering only the case of a cardinal B-spline basis, if $G(\cdot,\cdot,\cdot)$ is a $\mathcal{Q}_p(\widehat{\Omega})$ polynomial in its first argument, we have the following reproduction property (cf. Remark 4.2).

Proposition 5.1. *Assume that (5.2a) and (5.2b) hold. For all $\widehat{\boldsymbol{x}} \in \widetilde{\Omega}$, define*

$$\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}) = \int_{\widehat{\omega}_{\boldsymbol{\delta}}} G(\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}}, \widehat{B}_{\boldsymbol{\delta}}(\widehat{\boldsymbol{y}}), \widehat{B}(\widehat{\boldsymbol{y}}))\, \mathrm{d}\widehat{\boldsymbol{y}}. \tag{5.3}$$

If $G(\cdot, \widehat{\boldsymbol{y}}, \widehat{\boldsymbol{y}}) \in \mathcal{Q}_p(\widehat{\Omega})$, for every $\widehat{\boldsymbol{y}} \in \widehat{\Omega}$, then $\Phi_{\boldsymbol{\delta}} \in \mathcal{Q}_q(\widetilde{\Omega})$. Moreover, taking (5.1a) as the definition of the surrogate $\widetilde{\mathsf{A}}$, it holds that $\widetilde{\mathsf{A}} = \mathsf{A}$.

Proof. It suffices to show that $\widetilde{\Phi}_{\boldsymbol{\delta}} = \Phi_{\boldsymbol{\delta}}$, for all $\boldsymbol{\delta} \in \mathcal{D}$. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ be a multi-index, $\widetilde{\boldsymbol{x}}^{\boldsymbol{\alpha}} = \widetilde{x}_1^{\alpha_1} \cdots \widetilde{x}_n^{\alpha_n}$. By assumption, we may express $G(\widetilde{\boldsymbol{x}}, \widehat{B}_{\boldsymbol{\delta}}(\widehat{\boldsymbol{y}}), \widehat{B}(\widehat{\boldsymbol{y}})) = \sum_{i=1}^{n} \sum_{\alpha_i \le p} c_{\boldsymbol{\alpha}}(\widehat{\boldsymbol{y}}) \widetilde{\boldsymbol{x}}^{\boldsymbol{\alpha}}$, where each coefficient function $c_{\boldsymbol{\alpha}}(\widehat{\boldsymbol{y}})$ has support only in $\widehat{\omega}_{\boldsymbol{\delta}}$. Moreover, if $\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}} \in \widehat{\Omega}$, then

$$G(\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}}, \widehat{B}_{\boldsymbol{\delta}}(\widehat{\boldsymbol{y}}), \widehat{B}(\widehat{\boldsymbol{y}})) = \sum_{i=1}^{n} \sum_{\alpha_i \le p} c_{\boldsymbol{\alpha}}(\widehat{\boldsymbol{y}}) (\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}})^{\boldsymbol{\alpha}}$$

is clearly an equal degree polynomial in the $\widetilde{\boldsymbol{x}}$-variable. The proof is completed noting that the integral in (5.3) is performed in the $\widehat{\boldsymbol{y}}$-variable over the set $\widehat{\omega}_{\boldsymbol{\delta}}$ and $\widehat{\omega}_{\boldsymbol{\delta}} \subseteq \widehat{\Omega}$, for every $\boldsymbol{\delta} \in \mathcal{D}$. □

6. Surrogate matrices: faster assembly with existing software

All the examples in this paper were implemented using the GeoPDEs package for Isogeometric Analysis in MATLAB and Octave [26,27]. This package provides a framework for implementing and testing new isogeometric methods for the solution of partial differential equations. We reused most of the original low-level functions and only had to make changes to some high-level assembly functions. A detailed explanation of the code modifications as well as a reference implementation with example code is provided in [28]. Nonetheless, we give here a short explanation of the implementation in GeoPDEs. In particular, for the Poisson problem, we modified `op_gradu_gradv_tp` using the following strategy:

First, the sampling length H and the sampling points need to be specified. For this purpose, we introduce the sampling parameter $M \in \mathbb{N}$ which relates the small scale h to the coarse scale H via $H = M \cdot h$. Starting from the first point $\widetilde{\boldsymbol{x}}_i \in \widetilde{\Omega} \cap \widetilde{\mathbb{X}}$, let $\widetilde{\boldsymbol{\Xi}}$ be the lattice containing every Mth point $\widetilde{\boldsymbol{x}}_i \in \widetilde{\Omega} \cap \widetilde{\mathbb{X}}$, in each Cartesian direction. When M does not evenly divide these points in any given Cartesian direction, include the Mth endpoints $\widetilde{\boldsymbol{x}}_i \in \partial\widetilde{\Omega} \cap \widetilde{\mathbb{X}}$ as well; see, e.g., the black dots in Fig. 4.1. The stencil functions $\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}})$ are evaluated at all points $\widetilde{\boldsymbol{x}}_i^{\mathrm{s}} \in \widetilde{\boldsymbol{\Xi}}$ and these values are used as the support points of the ensuing B-spline interpolant $\widetilde{\Phi}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}})$.

An additional benefit of this choice is seen in that $\widetilde{\Phi}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i^{\mathrm{s}}) = \Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i^{\mathrm{s}})$, at each point $\widetilde{\boldsymbol{x}}_i^{\mathrm{s}} \in \widetilde{\boldsymbol{\Xi}}$. This leads to an increased point-wise accuracy and lower potential cost, since each entry $\widetilde{\mathsf{A}}_{ij} = \widetilde{\Phi}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i^{\mathrm{s}})$ in the surrogate stiffness matrix is equal to the correct entry, $\mathsf{A}_{ij} = \Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i^{\mathrm{s}})$. Here, it is appropriate to point out that when $M = 1$ every point $\widetilde{\boldsymbol{x}}_i$ is sampled, $\widetilde{\boldsymbol{\Xi}} = \widetilde{\Omega} \cap \widetilde{\mathbb{X}}$. In this case, $H = h$ and there is no difference from the surrogate $\widetilde{\mathsf{A}}$ and the true A.

In order to evaluate $\Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i^{\mathrm{s}})$, we identify the matrix rows which correspond to the sampling points $\widetilde{\boldsymbol{x}}_i^{\mathrm{s}} \in \widetilde{\boldsymbol{\Xi}}$. Additionally, we include the rows which correspond to basis functions near the domain boundary. Each of these rows needs to be assembled using quadrature formulas; see the red and green points in Fig. 6.1. The number of interior rows depends on M, whereas the number of rows corresponding to the boundary depends on the order of the basis functions p. After that, we identify all of the active elements which need to be assembled to compute the estimated rows, cf. Fig. 6.2. Note that the number of required elements for each sample point depends on the order p. In order to assemble these elements, we employ the `op_gradu_gradv` function, but skip the elements which are not active.

To construct the interpolated stencil functions $\widetilde{\Phi}_{\boldsymbol{\delta}}$, it is possible to use the builtin MATLAB functions `interp2` and `interp3`. However, in 2D, we use the `RectBivariateSpline` function provided by the SciPy Python package [35], which supports spline interpolation up to order 5. These interpolated stencil functions are then evaluated in order to retrieve the remaining values of $\widetilde{\mathsf{A}}$; cf. the blue off-diagonal entries in Fig. 6.1. The assembly

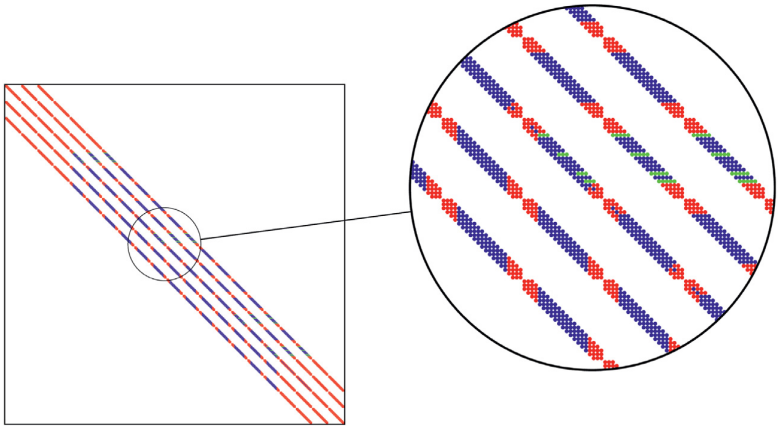


Fig. 6.1. Sparsity pattern of the surrogate stiffness matrix $\widetilde{\mathsf{A}}$. The red and green points indicate the entries of the stiffness matrix which are evaluated in the standard way. The blue points indicate the entries which are obtained by evaluating the interpolated stencil functions. The red points correspond to the basis functions near the boundaries and the green entries are used as supporting points for the interpolation. Some of the diagonal entries are drawn in blue due to the modification of preserving the kernel. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

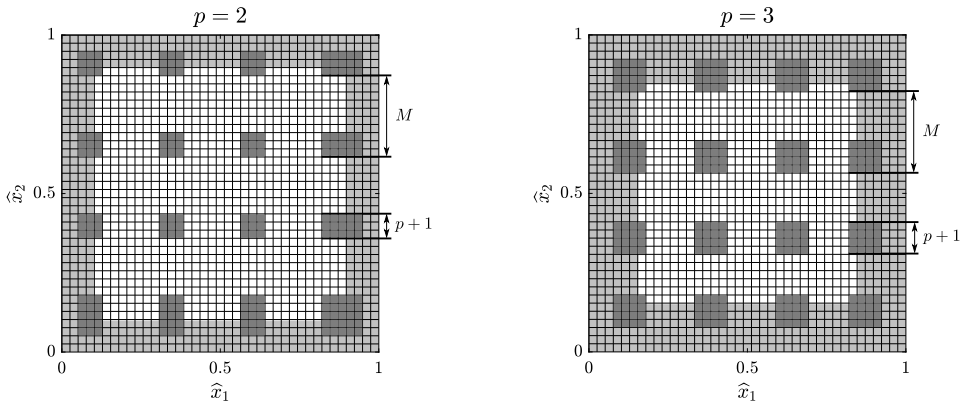


Fig. 6.2. The active elements (shown in gray) involved in the surrogate assembly for $M = 10$ with forty knots in each Cartesian direction. The light gray elements correspond to the active boundary elements and the dark gray elements correspond to the inner active elements required for the sampling of the stencil functions.

functions for the mass matrix, the biharmonic equation, and the Stokes problem were modified in a similar way. For symmetric matrices, only the upper-diagonal entries are interpolated and copied to the lower-diagonal entries (cf. (5.1b)). In the Poisson and biharmonic case, we additionally enforce the zero-row sum property by changing the diagonal entries for all rows which include at least one interpolated value (cf. (5.1c)).

In order to show that the surrogate approach may be easily applied to other IGA frameworks, we tried to keep the modifications as simple as possible. However, in an IGA implementation tailored to the surrogate approach, even more properties may be exploited to achieve better performance. For example, in the current implementation, the complete local stiffness matrices of the active elements are computed via quadrature, but in practice only a single row of the local matrix is required. Exploiting this fact would save a significant amount of unnecessary computation, especially as p grows, but such an implementation in GeoPDEs would also involve the modification of low-level functions.

Remark 6.1. Dirichlet boundary conditions are enforced for surrogate methods in the standard way; that is, by eliminating dofs from the original linear system. As usual, let $\widetilde{\mathsf{A}}$ be the full matrix without any consideration for boundary conditions. Without loss of generality, we may assume that all the smallest global indices correspond to the set of interior dofs, denoted by I, and all of the largest correspond to the set of Dirichlet dofs, denoted by D. The unconstrained linear system, $\widetilde{\mathsf{A}}\widetilde{\mathsf{u}} = \mathsf{f}$, may then be written in block form as follows:

$$\begin{bmatrix} \widetilde{\mathsf{A}}_{II} & \widetilde{\mathsf{A}}_{ID} \\ \widetilde{\mathsf{A}}_{DI} & \widetilde{\mathsf{A}}_{DD} \end{bmatrix} \begin{bmatrix} \widetilde{\mathsf{u}}_I \\ \widetilde{\mathsf{u}}_D \end{bmatrix} = \begin{bmatrix} \mathsf{f}_I \\ \mathsf{f}_D \end{bmatrix}. \tag{6.1}$$

Recall that all cardinal basis functions vanish at the domain boundary $\partial\Omega$. Therefore, following definition (5.1c), $\widetilde{\mathsf{A}}_{ID} = \mathsf{A}_{ID}$ and $\widetilde{\mathsf{A}}_{DD} = \mathsf{A}_{DD}$, where A_{ID} and A_{DD} are the corresponding submatrices of the standard stiffness matrix A. Furthermore, because the values of $\widetilde{\mathsf{u}}_D$ are prescribed, (6.1) may be reduced to the linear system $\widetilde{\mathsf{A}}_{II}\widetilde{\mathsf{u}}_I = \mathsf{f}_I - \mathsf{A}_{ID}\widetilde{\mathsf{u}}_D$. This reduced system may be solved to determine all the unprescribed solution coefficients $\widetilde{\mathsf{u}}_I$.

7. Poisson's equation

In this section, we analyze a surrogate IGA discretization of Poisson's equation on the domain $\Omega = \boldsymbol{\varphi}(\widehat{\Omega})$. Here, as well as in the forthcoming problems, we restrict our attention to Dirichlet boundary conditions. This simplifies the analysis while also retaining all of its interesting features. Given a function $f \in L^2(\Omega)$, the corresponding weak form is the following:

$$\text{Find } u \in H_0^1(\Omega) \text{ satisfying} \quad a(u, v) = F(v) \quad \text{for all } v \in H_0^1(\Omega), \tag{7.1}$$

where $a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x$ and $F(v) = \int_\Omega f v \, \mathrm{d}x$. At this point, it has been made well-understood that the bilinear form $a(\cdot, \cdot)$ can be rewritten on the parametric domain $\widehat{\Omega}$, using the expression (3.5). Recalling this detail, we continue on with the simplifying assumption $K \in \left[W^{q+1,\infty}(\widehat{\Omega})\right]^{n\times n}$.

7.1. Inconsistency

Recall (2.1) and (3.1) and take $V_h = \operatorname{span}\{N_i\}$, where every $N_i = \widehat{N}_i \circ \boldsymbol{\varphi}^{-1}$. Analysis of surrogate methods best proceeds using the surrogate bilinear form $\widetilde{a} : V_h \times V_h \to \mathbb{R}$ inherent to the surrogate matrix $\widetilde{\mathsf{A}}$. Explicitly,

$$\widetilde{a}(w_h, v_h) = \mathsf{v}^\top \widetilde{\mathsf{A}} \mathsf{w},$$

for all $w_h = \sum_i \mathsf{w}_i N_i$, $v_h = \sum_i \mathsf{v}_i N_i \in V_h$. Here and throughout, we shall use definition (5.1c) in constructing the surrogate stiffness matrix $\widetilde{\mathsf{A}}$ and its associated surrogate bilinear form $\widetilde{a}(\cdot, \cdot)$.

In the proceeding analysis, Theorem 7.2 is of fundamental importance. Its proof is a simple consequence of Theorem 4.2 and Lemma 7.1. From now on, for any matrix N, we use the notation $|\mathsf{N}|_{\max} = \max_{i\neq j} |\mathsf{N}_{ij}|$.

Lemma 7.1. *For all $v_h, w_h \in V_h$, the following upper bound holds:*

$$|a(v_h, w_h) - \widetilde{a}(v_h, w_h)| \leq C_3 h^{2-n} |\mathsf{A} - \widetilde{\mathsf{A}}|_{\max} \|\nabla v_h\|_0 \|\nabla w_h\|_0 , \tag{7.2}$$

where C_3 is a constant depending only on $\boldsymbol{\varphi}$ and p.

Proof. In this proof, we will use the symbols "$\lesssim$" and "$\eqsim$" to denote upper bounds and equivalence, respectively, up to constants depending at most on p and $\boldsymbol{\varphi}$. For each $i = 1, \ldots, N$, let $\mathcal{I}(i)$ be the set of indices j such that $\operatorname{supp}(N_i) \cap \operatorname{supp}(N_j) \neq \emptyset$ and notice that $\mathcal{I}(i) \leq |\mathcal{D}| = (2p+1)^n$. We begin with the observation that $\sum_i (\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}) = \sum_j (\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}) = 0$ due to definition (5.1c). With these two identities at hand, we find that

$$\begin{aligned} a(v_h, w_h) - \widetilde{a}(v_h, w_h) &= -\frac{1}{2} \sum_{i,j} (\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij})(\mathsf{v}_i - \mathsf{v}_j)(\mathsf{w}_i - \mathsf{w}_j) \\ &\leq |\mathsf{A} - \widetilde{\mathsf{A}}|_{\max} \sum_i \sum_{j\in\mathcal{I}(i)} |\mathsf{v}_i - \mathsf{v}_j| |\mathsf{w}_i - \mathsf{w}_j| \\ &\leq |\mathsf{A} - \widetilde{\mathsf{A}}|_{\max} \sum_i \left(\sum_{j\in\mathcal{I}(i)} |\mathsf{v}_i - \mathsf{v}_j|^2 \right)^{1/2} \left(\sum_{j\in\mathcal{I}(i)} |\mathsf{w}_i - \mathsf{w}_j|^2 \right)^{1/2}. \end{aligned} \tag{7.3}$$

For the time being, fix the index $i = 1, \dots, N$. For each coefficient vector $\mathsf{v} \in \mathbb{R}^N$, define $|\mathsf{v}|_{\mathcal{I}(i)} = \left(\sum_{j\in\mathcal{I}(i)} |\mathsf{v}_i - \mathsf{v}_j|^2\right)^{1/2}$. One may easily check that $|\cdot|_{\mathcal{I}(i)}$ is a seminorm on $\mathbb{R}^N$. In order to identify its kernel, simply observe that $|\mathsf{v}|_{\mathcal{I}(i)} = 0$ iff $\mathsf{v}_j = \mathsf{v}_i$ for each coefficient $j \in \mathcal{I}(i)$. Since $i \in \mathcal{I}(i)$, an equivalent way of stating this condition is that there exists some constant $C \in \mathbb{R}$ such that $\mathsf{v}_j = C$ for each $j \in \mathcal{I}(i)$. Now, recall that $v_h(\boldsymbol{x}) = \sum_i \mathsf{v}_i N_i(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$ and consider the following alternative seminorm:

$$|\mathsf{v}|_{N_i} = \|\nabla v_h\|_{0,\mathrm{supp}(N_i)}\,.$$

Notice that $|\mathsf{v}|_{N_i} = 0$ iff $v_h(\boldsymbol{x})$ is equal to a constant on $\mathrm{supp}(N_i)$, say $C \in \mathbb{R}$. From the partition of unity property inherent to all NURBS bases, it must hold that $C = v_h(\boldsymbol{x}) = \sum_{j\in\mathcal{I}(i)} C N_j(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathrm{supp}(N_i)$. In other words, since $\{N_i|_{\mathrm{supp}(N_i)}\}_{j\in\mathcal{I}(i)}$ is a linearly independent set, $|\mathsf{v}|_{N_i} = 0$ iff $\mathsf{v}_j = C$ for each $j \in \mathcal{I}(i)$.

In the previous paragraph, we showed that the kernels of $|\mathsf{v}|_{N_i}$ and $|\mathsf{v}|_{\mathcal{I}(i)}$ are identical; namely, $|\mathsf{v}|_{N_i} = 0$ iff $|\mathsf{v}|_{\mathcal{I}(i)} = 0$ iff $\mathsf{v} \in Q_i$, where

$$Q_i = \{\mathsf{v} \in \mathbb{R}^N : \mathsf{v}_j = \mathsf{v}_i \text{ for each } j \in \mathcal{I}(i)\}\,.$$

Clearly, $|\cdot|_{N_i}$ and $|\cdot|_{\mathcal{I}(i)}$ induce norms on the quotient space $\mathbb{R}^N/Q_i$. The next important observation is that $|Q_i| = N + 1 - |\mathcal{I}(i)|$, which may be witnessed by inspection. Because $\dim(\mathbb{R}^N/Q_i) = |\mathcal{I}(i)| - 1 \le (2p+1)^n - 1$ is finite and depends only on p, it follows from the well-known equivalence of norms on finite dimensional vector spaces (e.g., the vector space $\mathbb{R}^N/Q_i$) that the seminorms $|\cdot|_{N_i}$ and $|\cdot|_{\mathcal{I}(i)}$ are equivalent. Of course, the corresponding equivalence constants will depend on h, $\boldsymbol{\varphi}$, and p. Nevertheless, a standard scaling argument is all that is required to see that $|\mathsf{v}|_{\mathcal{I}(i)} \eqsim h^{2-n}|\mathsf{v}|_{N_i}$. Therefore, employing (7.3), we may simply write

$$|a(v_h, w_h) - \widetilde{a}(v_h, w_h)| \lesssim h^{2-n}|\mathsf{A} - \widetilde{\mathsf{A}}|_{\max} \sum_i \left\|\nabla v_h\right\|_{0,\mathrm{supp}(N_i)} \left\|\nabla w_h\right\|_{0,\mathrm{supp}(N_i)}\,.$$

The proof is completed by applying the discrete Cauchy–Schwarz inequality to the right-hand side of the inequality above and employing the fact that $\left(\sum_i \|f\|^2_{0,\mathrm{supp}(N_i)}\right)^{1/2} \eqsim \|f\|_0$, for all $f \in L^2(\Omega)$. □

Theorem 7.2. *Let $C_4 = C_2 \cdot C_3$. For all $v_h, w_h \in V_h$, the following upper bound holds:*

$$|a(v_h, w_h) - \widetilde{a}(v_h, w_h)| \le C_4 H^{q+1} \|\nabla v_h\|_0 \|\nabla w_h\|_0\,.$$

Proof. Obviously, we are done if $a(\cdot,\cdot) = \widetilde{a}(\cdot,\cdot)$. Therefore, assume $a(\cdot,\cdot) \neq \widetilde{a}(\cdot,\cdot)$ and let $i < j$ be the indices of the maximal value $|\mathsf{A}_{ij} - \widetilde{\mathsf{A}}_{ij}| = \left|\mathsf{A} - \widetilde{\mathsf{A}}\right|_{\max} > 0$. Since $\widetilde{\mathsf{A}}$ is defined by (5.1c), Theorem 4.2 leads us to the inequality

$$\left|\mathsf{A} - \widetilde{\mathsf{A}}\right|_{\max} = |\Phi_\delta(\widetilde{\boldsymbol{x}}_i) - \widetilde{\Phi}_\delta(\widetilde{\boldsymbol{x}}_i)| \le C_2 h^{n-2} H^{q+1}\,.$$

The proof is completed using Lemma 7.1. □

Remark 7.1. In Lemma 7.1, it is important that $\widetilde{\mathsf{A}}$ be defined using (5.1c). Indeed, in (7.3), it is the symmetry and the zero row sum property preserved in this definition which allows $|a(v_h, w_h) - \widetilde{a}(v_h, w_h)|$ to be bounded by products of differences in the coefficients v_i and w_j. If we had used definition (5.1a) or (5.1b), one would have to directly work with the upper bound

$$|a(v_h, w_h) - \widetilde{a}(v_h, w_h)| \le \|\mathsf{A} - \widetilde{\mathsf{A}}\|_{\max} \sum_i \sum_{j\in\mathcal{I}(i)} |\mathsf{v}_i||\mathsf{w}_j|\,.$$

This, in turn, can only be finessed to arrive at an inequality like

$$|a(v_h, w_h) - \widetilde{a}(v_h, w_h)| \le C_3' h^{-n} \|\mathsf{A} - \widetilde{\mathsf{A}}\|_{\max} \|v_h\|_0 \|w_h\|_0\,, \tag{7.4}$$

for some other constant C_3', depending only on $\boldsymbol{\varphi}$ and p. Notice the loss of an h^2 scaling factor when comparing (7.2) and (7.4). This difference could greatly affect solution accuracy.

7.2. A priori error estimation

Define $V_{h,0} = V_h \cap H_0^1(\Omega)$. The following lemma is identical in spirit to [24, Theorem 7.1]. By the Lax–Milgram theorem, this lemma allows us to conclude that there exists a unique surrogate solution corresponding to (7.1), namely $\widetilde{u}_h \in V_{h,0}$, for every sufficiently small $H > 0$.

Lemma 7.3. *Let $\alpha = (1 + C_P)^{-1}$, where C_P is the Poincaré constant for the domain Ω. If $H^{q+1} < \alpha \cdot C_4^{-1}$, then the surrogate bilinear form $\widetilde{a} : V_h \times V_h \to \mathbb{R}$ is coercive on $V_{h,0}$. Letting $\widetilde{\alpha} > 0$ be the associated coercivity constant, it also holds that $\widetilde{\alpha} \to \alpha$, as $H \to 0$.*

Proof. Let $S = \{v \in H_0^1(\Omega) : \|v\|_1 = 1\}$ be the surface of the unit ball in $H_0^1(\Omega)$. Notice that $\alpha \leq a(v_h, v_h) \leq \widetilde{a}(v_h, v_h) + |a(v_h, v_h) - \widetilde{a}(v_h, v_h)|$ for all $v_h \in V_h \cap S$ and, therefore,

$$\alpha - |a(v_h, v_h) - \widetilde{a}(v_h, v_h)| \leq \widetilde{a}(v_h, v_h) \quad \text{for all } v_h \in V_h \cap S\,.$$

Invoking Lemma 7.1, the second term on the left may be bounded from above as follows:

$$|a(v_h, v_h) - \widetilde{a}(v_h, v_h)| \leq C_4 H^{q+1}\,.$$

Clearly, if $C_4 H^{q+1} < \alpha$, then $0 < \alpha - |a(v_h, v_h) - \widetilde{a}(v_h, v_h)| \leq \widetilde{a}(v_h, v_h)$, as necessary. □

Theorem 7.4. *Let $\theta > 1$. If $u \in H^{p+1}(\Omega)$, then there exists a constant C_5, depending only on p and $\boldsymbol{\varphi}$, such that*

$$\|u - \widetilde{u}_h\|_1 \leq C_5 h^p \|u\|_{p+1} + \theta \cdot \alpha^{-1} C_4 H^{q+1} \|\nabla u\|_0\,, \tag{7.5a}$$

for every sufficiently small $H > 0$. Additionally, if $\Omega \subseteq \mathbb{R}^n$ is convex, then there exists a constant C_6, depending only on p and $\boldsymbol{\varphi}$, such that

$$\|u - \widetilde{u}_h\|_0 \leq C_6 h^{p+1} \|u\|_{p+1} + \theta \cdot C_P C_4 H^{q+1} \|\nabla u\|_0\,, \tag{7.5b}$$

for every sufficiently small $H > 0$.

Proof. We first prove (7.5a). Let $u_h \in V_{h,0}$ be the solution of the standard IGA discretization (2.1b) associated to (7.1). Clearly, $\|u - \widetilde{u}_h\|_1 \leq \|u - u_h\|_1 + \|u_h - \widetilde{u}_h\|_1$. Moreover, by [20, Theorem 6.1], $\|u - u_h\|_1 \leq C_5 h^p \|u\|_{p+1}$. Recalling Lemma 7.3, we find that

$$\widetilde{\alpha}\|u_h - \widetilde{u}_h\|_1^2 \leq \widetilde{a}(u_h - \widetilde{u}_h, u_h - \widetilde{u}_h) = \widetilde{a}(u_h - \widetilde{u}_h, u_h) - a(u_h - \widetilde{u}_h, u_h)\,.$$

After invoking Theorem 7.2, it now readily follows that $\|u_h - \widetilde{u}_h\|_1 \leq \widetilde{\alpha}^{-1} C_4 H^{q+1} \|\nabla u\|_0$. Since $\widetilde{\alpha} \to \alpha$, in the limit $H \to 0$, it also holds that $\|u_h - \widetilde{u}_h\|_1 \leq \theta \cdot \alpha^{-1} C_4 H^{q+1} \|\nabla u\|_0$, for all sufficiently small $H > 0$.

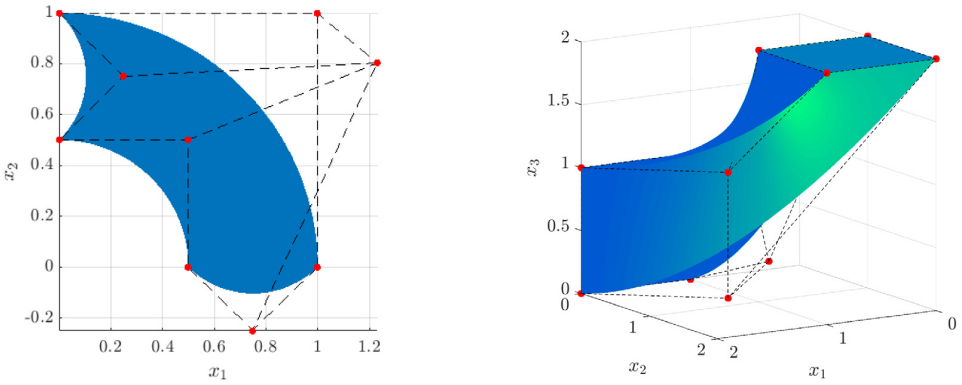
Our proof of (7.5b), also involves the triangle inequality: $\|u - \widetilde{u}_h\|_0 \leq \|u - u_h\|_0 + \|u_h - \widetilde{u}_h\|_0$. If Ω is convex, then $\|u - u_h\|_0 \leq C_6 h^{p+1} \|u\|_{p+1}$. Next, find $w_h \in V_{h,0}$ satisfying $a(w_h, v_h) = (u_h - \widetilde{u}_h, v_h)_\Omega$, for all $v_h \in V_{h,0}$. It holds that $\|\nabla w_h\|_0 \leq C_P \|u_h - \widetilde{u}_h\|_0$. Now, observe that $\|u_h - \widetilde{u}_h\|_0^2 = a(w_h, u_h - \widetilde{u}_h) = \widetilde{a}(w_h, \widetilde{u}_h) - a(w_h, \widetilde{u}_h)$. Finally, invoke Theorem 7.2 to arrive at

$$\|u_h - \widetilde{u}_h\|_0^2 \leq C_4 H^{q+1} \|\nabla \widetilde{u}_h\|_0 \|\nabla w_h\|_0 \leq C_P C_4 H^{q+1} \|\nabla \widetilde{u}_h\|_0 \|u_h - \widetilde{u}_h\|_0\,,$$

which works to deliver the stated estimate, since $\|\nabla \widetilde{u}_h\|_0 \to \|\nabla u_h\|_0 \leq \|\nabla u\|_0$, as $H \to 0$. □

7.3. Numerical experiments

The two bounds (7.5a) and (7.5b) need to be experimentally verified. Because both estimates depend on the two scales h and H, in order to be overtly thorough, we should provide verification in both scales, independently. That is, first holding the h-scale fixed and varying H and, alternatively, holding the H-scale fixed and varying h. We forgo this verification step and instead refer the reader to similar studies with low-order finite elements in [21–24]. In this section, therefore, we verify the bounds above under the assumption $H = H(h)$. We expect that this would be the typical use case.



(a) 2D domain (b) 3D domain

Fig. 7.1. Two domains Ω for problem (7.1).

7.3.1. Overview

In our experiments with Poisson's equation (7.1), we considered a 2D and a 3D domain Ω; see Fig. 7.1. The 2D domain is given by a second-order NURBS parameterization and the 3D domain is parameterized by second-order B-splines. This particular 2D domain (Fig. 7.1(a)) was chosen instead of the 2D quarter annulus featured later (Fig. 8.1) in order to break some symmetries which would otherwise appear in its stencil functions (Fig. 4.1). (Additional stencil function symmetries can appear if $\boldsymbol{\varphi}\colon \widehat{\Omega} \to \Omega$ is a conformal mapping.) We also note that, after fixing the NURBS parameterizations for the edges on the boundary of the 2D domain, the parameterization $\boldsymbol{\varphi}\colon \widehat{\Omega} \to \Omega$ was generated as a Coons patch [29]. Because each of the four edges had polynomial weight functions in their parameterizations, the resulting Coons patch NURBS parameterization, $\boldsymbol{\varphi}\colon \widehat{\Omega} \to \Omega$, contained a global polynomial weight function $W(\widehat{\boldsymbol{x}}) = \sum_j w_j \widehat{B}_j(\widehat{\boldsymbol{x}})$. Consequently, the assumptions of Theorem 4.2 were satisfied in all of our experiments.

In constructing $\widetilde{\mathsf{A}}$ (cf. Section 6) and, thus, the surrogate stencil functions $\widetilde{\Phi}_{\boldsymbol{\delta}} = \Pi_H \Phi_{\boldsymbol{\delta}}$, we tried local B-spline interpolants $\Pi_H\colon C^0(\widetilde{\Omega}) \to S_q(\widetilde{\boldsymbol{\Xi}})$ of various order $1 \le q \le 5$. As stated previously (see Remark 4.1), we used the convention $\widetilde{\boldsymbol{\Xi}} \subseteq \widetilde{\Omega} \cap \widetilde{\mathbb{X}}$. In our 2D experiments, we explored both $p = 2$ or $p = 3$ NURBS bases and the full range of possible q. In our 3D experiments, we only explored the case $p = 2$ using $q = 1, 3$. This limitation of our 3D experiments was due to the restraints of the MATLAB B-spline interpolation routine we were using.

In the first set of experiments (Figs. 7.2–7.4), the sampling length H was set to $H = Mh$, with the constant $M = 5$. In the second set of experiments (Figs. 7.5 and 7.6), we used a mesh-dependent sampling length, defined below.

7.3.2. Constant sampling lengths

For the constant sampling strategy, with $M = 5$, convergence plots of the errors in surrogate solutions are presented in Figs. 7.2–7.4. After inspecting these figures, the reader should observe that there is a noticeable difference in the accuracy of the surrogate solutions $\widetilde{u}_h$, dependent on the load f. For instance, with the "low-frequency" manufactured solution, $u(\boldsymbol{x}) = \sin(\pi\, x_1)\, \sin(\pi\, x_2)$, the errors in the surrogate solutions appear to converge to the error generated by the (standard) non-surrogate IGA solution u_h asymptotically, at various rates. However, for the "high-frequency" manufactured solution, $u(\boldsymbol{x}) = \sin(20\,\pi\, x_1)\, \sin(20\,\pi\, x_2)$, each of the surrogate errors and the corresponding standard IGA error are nearly indistinguishable (except, sometimes, in the case $q = 1$).

This difference can be explained in a simple way. Observe that the h-dependent terms in (7.5a) and (7.5b) are multiplied by the high-order norm $\|u\|_{p+1}$ and, meanwhile, the H-dependent terms are only multiplied by the lower-order seminorm $\|\nabla u\|_0$. Let us call the first term in each bound the *discretization error* and the second term the *consistency error*. Because of the presence of the high-order norm $\|u\|_{p+1}$, the discretization error is much more sensitive to irregularities or oscillations in the solution. Another important detail not to overlook is that C_4

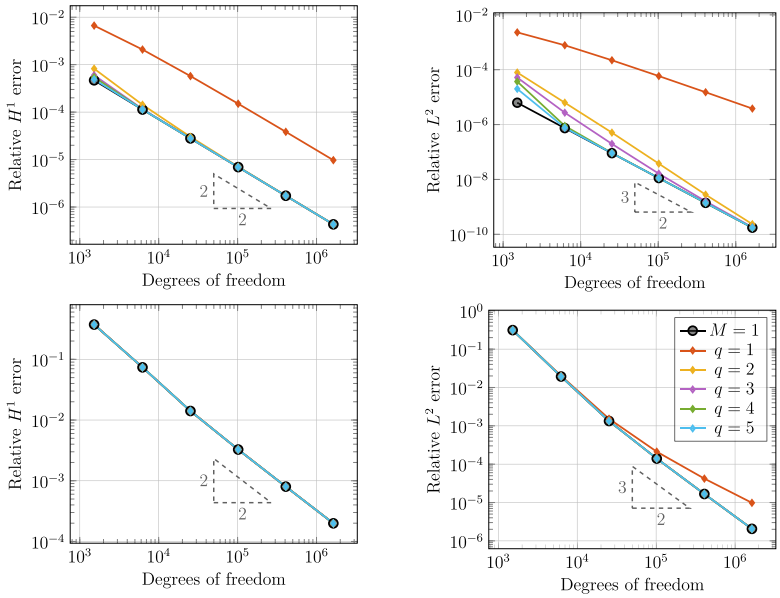


Fig. 7.2. 2D domain. Relative errors for $p = 2$, $M = 5$, and two different manufactured solutions. Top row: $u(\boldsymbol{x}) = \sin(\pi\, x_1)\, \sin(\pi\, x_2)$. Bottom row: $u(\boldsymbol{x}) = \sin(20\,\pi\, x_1)\, \sin(20\,\pi\, x_2)$.

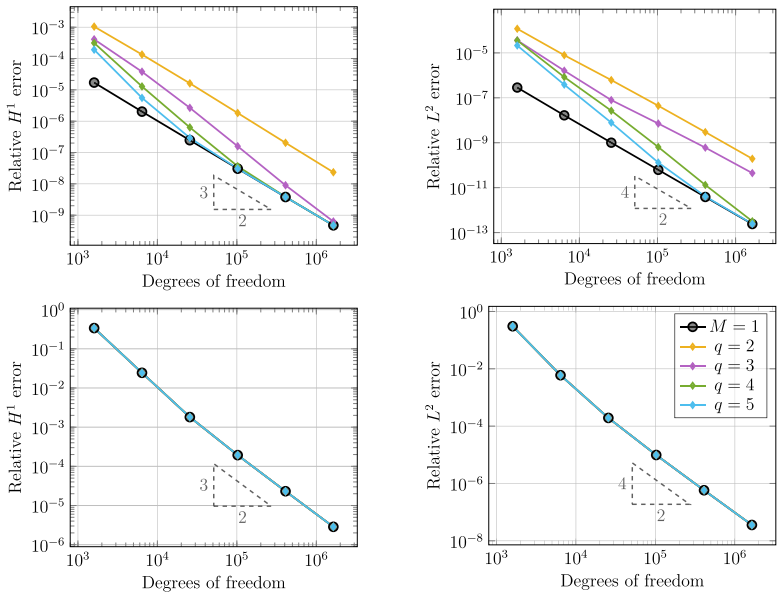


Fig. 7.3. 2D domain. Relative errors for $p = 3$, $M = 5$, and two different manufactured solutions. Top row: $u(\boldsymbol{x}) = \sin(\pi\, x_1)\, \sin(\pi\, x_2)$. Bottom row: $u(\boldsymbol{x}) = \sin(20\,\pi\, x_1)\, \sin(20\,\pi\, x_2)$.

ultimately involves high-order norms of the tensor coefficient $K(\widehat{\boldsymbol{x}})$. Meanwhile, C_5 and C_6 depend (through Ω), at most, on the H^2-norm of $K(\widehat{\boldsymbol{x}})$.

These observations can lead in many interesting directions, but the conclusion which the reader should ultimately arrive at is that the assembly of the stiffness matrix should only be carried out to the accuracy required by the problem at hand. Therefore, the correct surrogate assembly strategy must take into account properties of the problem

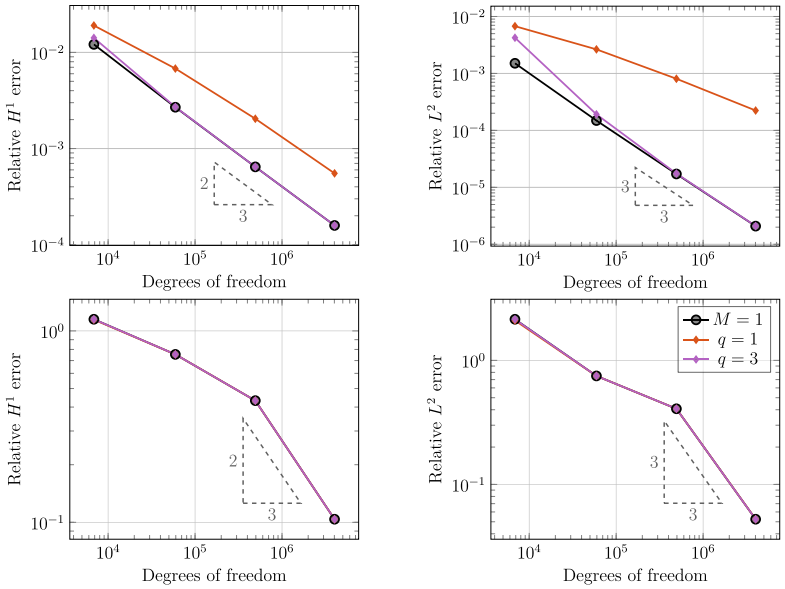


Fig. 7.4. 3D domain. Relative errors for $p = 2$, $M = 5$, and two different manufactured solutions. Top row: $u(\boldsymbol{x}) = \sin(\pi\, x_1)\, \sin(\pi\, x_2)\, \sin(\pi\, x_3)$. Bottom row: $u(\boldsymbol{x}) = \sin(20\,\pi\, x_1)\, \sin(20\,\pi\, x_2)\, \sin(20\,\pi\, x_3)$.

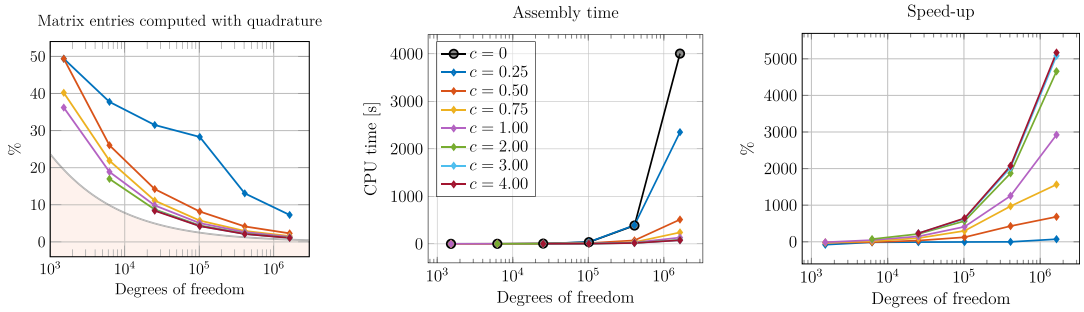


Fig. 7.5. 2D domain with $p = 2$ and $q = 5$. Assembly times with the mesh-dependent sampling strategy $M(h) = \max\left\{1, \lfloor c \cdot h^{\frac{p-q+\frac{1}{2}}{q+1}} \rfloor\right\}$. On the left, note the percentage of matrix entries computed with quadrature. Here, the percentage of entries involving non-cardinal basis functions (roughly $1 - (\frac{m-2p}{m})^n$) is shaded out to indicate the theoretical lower bound.

geometry and the given loads, as well as each of the parameters h, p, and q. A simple way of doing this is to use mesh-dependent sampling lengths.

Remark 7.2. In [24], it was documented that the L^2 error converged like $h^{p+1} + H^{q+2}$ when Π_H resembles an L^2 projection. However, when using an interpolation instead, like it is done in this work, the error converges like $h^{p+1} + H^{q+2}$ whenever the interpolation order q is even and like $h^{p+1} + H^{q+1}$ otherwise. This parity effect can also be observed in results of this work; see, e.g., the top right plot in Fig. 7.3. We do not prove this improved convergence rate but refer to proofs of quadrature formulas where the same parity effect can be observed; see, e.g., [36, Chapter 2.5].

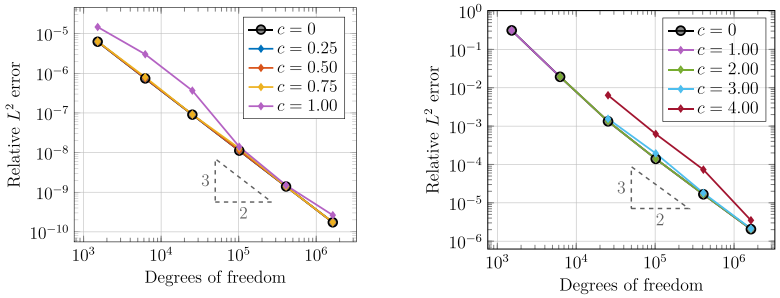


Fig. 7.6. 2D domain with $p = 2$ and $q = 5$. Relative L^2 errors for $M(h) = \max\left\{1, \lfloor c \cdot h^{\frac{p-q+\frac{1}{2}}{q+1}} \rfloor\right\}$, with two different manufactured solutions. Left: $u(\boldsymbol{x}) = \sin(\pi\, x_1)\, \sin(\pi\, x_2)$. Right: $u(\boldsymbol{x}) = \sin(20\,\pi\, x_1)\, \sin(20\,\pi\, x_2)$.

7.3.3. Mesh-dependent sampling lengths

In a successful surrogate method, the discretization error should always dominate the consistency error. Of course, however, this domination should not be exercised to such an extent that overall efficiency is compromised. As a rule of thumb, keeping the consistency error at or below 5% of the discretization error is often acceptable. Even within this somewhat conservative threshold, balancing the two errors appropriately can lead to very significant performance advantages.

Let $q > p$ and consider the mesh-dependent sampling length $M(h) = \max\left\{1, \lfloor c \cdot h^{\frac{p-q+\beta}{q+1}} \rfloor\right\}$, where both $c, \beta \geq 0$ are tuneable parameters. Returning the definition $H = M \cdot h$, we now find that, for any $c > 0$,

$$H^{q+1} \leq (M \cdot h)^{q+1} = c^{q+1} h^{p+1+\beta}, \quad \text{as } h \to 0\,.$$

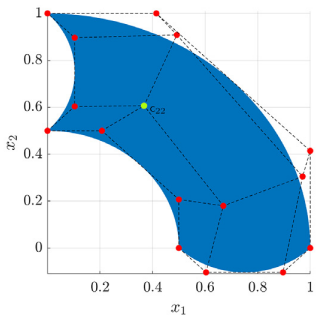
Here, for any $\beta > 0$, both the H^1 and L^2 discretization error will eventually dominate their associated consistency error. Therefore, this parameter exists to maintain the dominance of the discretization error, throughout mesh refinements. We found $\beta = 1/2$ to be a suitable choice for our purposes. The parameter c, on the other hand, must be calibrated to the problem at hand.

All run-time measurements in this sections were obtained on a machine equipped with two Intel® Xeon® Gold 6136 processors with a nominal base frequency of 3.0 GHz. Each processor has 12 physical cores which results in a total of 24 physical cores, but only single core was used to run the following experiments. The total available memory of 251 GB is split into two NUMA domains, one for each socket. For compatibility reasons with using SciPy, we employed the BLAS library in version 3.7.1 provided by the operating system Ubuntu 18.04.2, but using other optimized libraries might improve the performance of the MATLAB solver.

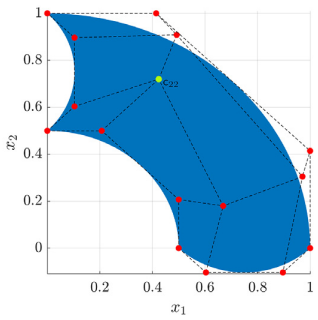
In Fig. 7.5 we show the assembly times our implementation accrued for various values of c, when $p = 2$ and $q = 5$. Inspection of Fig. 7.6 clearly shows that $c = 0.75$ is suitable for $u(\boldsymbol{x}) = \sin(\pi\, x_1)\, \sin(\pi\, x_2)$ and $c = 3$ is suitable for $u(\boldsymbol{x}) = \sin(20\,\pi\, x_1)\, \sin(20\,\pi\, x_2)$. Let t_{std} be the time required to assemble the matrix with the standard approach and t_{surr} the time using the surrogate approach. The speed-up from using the surrogate approach is then defined as $\frac{t_{\text{std}}}{t_{\text{surr}}} - 1$. When compared to the non-surrogate assembly strategy at just over one million degrees of freedom, note that the first choice gives an assembly speed-up of around 1500% and the second choice gives a speed-up of more than 5000%. These enormous speed-ups are in fact not that surprising after inspecting the percentage of matrix entries which must be computed using traditional quadrature formulas; see, again, Fig. 7.5.

7.3.4. Constant sampling lengths with non-smooth geometry maps

In this subsection, we investigate the impact on the surrogate method above if the assumption $W \in \mathcal{Q}_p(\widehat{\Omega})$ in Theorem 4.2 is violated. For this purpose, we consider the 2D benchmark from Section 7.3.2 but with a different geometry transformation $\boldsymbol{\varphi}$. We start with the domain illustrated in Fig. 7.1(a) and perform knot insertion, resulting in the control net shown in Fig. 7.7(a). The control net shown in Fig. 7.7(b) is constructed from Fig. 7.7(a) by slightly moving the internal control point marked in green. Precisely, the control point $\mathtt{c}_{22}$ is shifted by the vector $\alpha_h \cdot (1, 2, 0)^\top$ with $\alpha_h = 0.04$. Note that the map described by the control net in Fig. 7.7(a) results in a geometry map of C^∞ regularity whereas the control net from Fig. 7.7(b) results in a map of $C^{p-1} = C^1$ regularity if $\alpha_h \neq 0$.



(a) 2D domain with added internal knots. The resulting map is still of C^∞ regularity.



(b) 2D domain with added internal knots where the green control point has been moved. The resulting map is of C^1 regularity.

Fig. 7.7. 2D domains equivalent to the domain in Fig. 7.1(a) generated by geometry maps of different regularity..

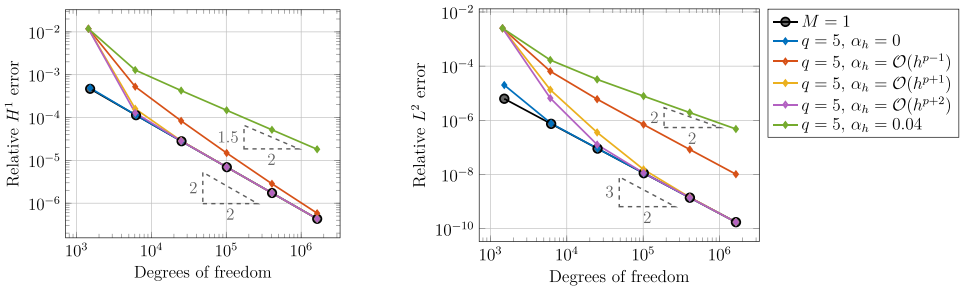


Fig. 7.8. 2D domain. Relative errors for $p = 2$, $M = 5$, and manufactured solution $u(\boldsymbol{x}) = \sin(\pi\, x_1)\, \sin(\pi\, x_2)$.

In the following, we perform a set of experiments with different choices of α_h. In the first experiment, we keep α_h fixed to $\alpha_h = 0.04$, resulting in the control net shown in Fig. 7.7(b). In the other experiments, we choose $\alpha_h = \mathcal{O}(h^{p+\gamma})$ for $\gamma \in \{-1, 1, 2\}$, i.e., $\alpha_h = 0.04 \cdot 40^{p+\gamma} \cdot h^{p+\gamma}$. This particular choice of α_h enforces that both maps in the coarsest problem are the same. For any sufficiently small $h > 0$, the resulting geometry maps are still of C^1 regularity but converge to the smoother map as $h \to 0$. A similar case of approximately smooth maps has already been investigated for the low-order finite element surrogate context in [21,22].

We perform the same computations as in Section 7.3.2 but consider only the surrogate interpolation degree $q = 5$ and compare the effect on the errors depending on the choice of α_h. The results are presented in Fig. 7.8. It may be observed that for $\alpha_h = 0.04$, the errors in the L^2-norm using the surrogate method converge with a lower rate than for $\alpha_h = 0$. For $\alpha_h = \mathcal{O}(h^{p-1})$, the optimal convergence rate is recovered, but the constant of the dominating error term is larger. For $\alpha_h = \mathcal{O}(h^{p+\gamma})$, with $\gamma > -1$, the reference discretization errors and rates are recovered asymptotically. This indicates that the assumption $W \in \mathcal{Q}_p(\widehat{\Omega})$ may be mildly violated without losing the convergence rates predicted by Theorem 7.4.

Remark 7.3. We sketch two approaches to improve the convergence rate of the surrogate method in cases where it is not optimal because of irregular geometry maps. The geometry transformation $\boldsymbol{\varphi}$ described by a general control net is built up of regions where the map is locally smooth and the lower regularity appears only at the interfaces of these regions. These locally smooth regions may be interpreted as patches in a multi-patch IGA setting where the surrogate method may be directly applied as described in Section 3.5. The matrix entries corresponding to the boundaries of these patches can be obtained by performing standard numerical quadrature. An alternative possibility is to chose the interpolation spaces in such a way that the basis functions capture the irregularity of the geometry

map. This can be done by a careful choice of the B-spline interpolation space since the locations of the irregularities are known beforehand in a fixed control net.

8. Transverse vibrations of an isotropic membrane

One of the great advantages of the IGA paradigm is its superior accuracy in structural vibration problems. Any worthwhile surrogate method should maintain this advantage. Therefore, in this section, we extend the analysis of the previous section to the analysis of transverse vibrations of a two-dimensional elastic membrane. The corresponding weak form is the following:

$$\begin{cases} \text{Find all pairs } u \in H_0^1(\Omega) \setminus \{0\} \text{ and } \lambda \in \mathbb{R} \text{ satisfying} \\ a(u, v) = \lambda\, m(u, v) \quad \text{for all } v \in H_0^1(\Omega)\,, \end{cases} \tag{8.1}$$

where $a(w, v) = \int_\Omega \nabla w \cdot \nabla v \,\mathrm{d}x$ and $m(w, v) = \int_\Omega w v \,\mathrm{d}x$. It is well-known that there are countably many solutions to this problem, $\{(u^{(k)}, \lambda^{(k)})\}_{j=1}^{\infty}$, where each $\lambda^{(k)} > 0$. From here on, we make the ordering assumption $\lambda^{(j)} \leq \lambda^{(k)}$, for every $j < k$.

8.1. Surrogate mass matrices

So far, we have only rigorously analyzed surrogates of the elliptic bilinear form $a(\cdot, \cdot)$, written above. Using the techniques developed thus far, results similar to Theorem 7.2 can be proven for other bilinear forms, such as $m(\cdot, \cdot)$. Let the corresponding mass matrix and surrogate mass matrix be denoted M and $\widetilde{\mathsf{M}}$, respectively. Employing definition (5.1b), we take for granted that the accompanying surrogate bilinear form $\widetilde{m}(\cdot, \cdot)$ satisfies

$$|m(v_h, w_h) - \widetilde{m}(v_h, w_h)| \leq C_7 H^{q+1} \|v_h\|_0 \|w_h\|_0\,, \tag{8.2}$$

for some C_7 depending only on $\boldsymbol{\varphi}$, p, q, and $\|\Pi_H\|$.

In the special case that the geometry mapping $\boldsymbol{\varphi}(\widehat{\boldsymbol{x}})$ is a polynomial, we have a surrogate reproduction property of the mass form $m(u, v)$. This property is formalized in the following corollary to Proposition 5.1.

Corollary 8.1. *Assume that the domain mapping $\boldsymbol{\varphi}\colon \widehat{\Omega} \to \Omega$ is defined through a polynomial of order p, i.e., $\boldsymbol{\varphi} \in \left[\mathcal{Q}_p(\widehat{\Omega})\right]^n$. Let M be the coefficient matrix arising from the discretization of $m(\cdot, \cdot)$ and $\widetilde{\mathsf{M}}$ the corresponding surrogate matrix. If $q \geq n \cdot p - 1$, it holds that $\mathsf{M} = \widetilde{\mathsf{M}}$.*

Proof. Let $\boldsymbol{J}(\widehat{\boldsymbol{x}})$ be the Jacobian (tensor) of $\boldsymbol{\varphi}(\widehat{\boldsymbol{x}})$. The transformation of the integral from the physical to the reference domain is given by

$$m(u, v) = \int_\Omega u v \,\mathrm{d}x = \int_{\widehat{\Omega}} \widehat{u}\widehat{v} \,\det(\boldsymbol{J})\,\mathrm{d}\widehat{\boldsymbol{x}}.$$

It remains to show that $G(\widehat{\boldsymbol{x}}, \widehat{u}(\widehat{\boldsymbol{y}}), \widehat{v}(\widehat{\boldsymbol{y}})) = \widehat{u}(\widehat{\boldsymbol{y}})\widehat{v}(\widehat{\boldsymbol{y}})\,\det(\boldsymbol{J}(\widehat{\boldsymbol{x}}))$ is a polynomial of degree $n \cdot p - 1$ in the $\widehat{\boldsymbol{x}}$-variable. Since $\boldsymbol{\varphi} \in \left[\mathcal{Q}_p(\widehat{\Omega})\right]^n$, it holds that $\det(\boldsymbol{J}) \in \mathcal{Q}_{n\cdot p-1}(\widehat{\Omega})$. Applying Proposition 5.1 yields the desired reproduction property. □

8.2. A priori analysis of the eigenvalue error

Adopt the obvious notation, $(u_h^{(k)}, \lambda_h^{(k)})$ and $(\widetilde{u}_h^{(k)}, \widetilde{\lambda}_h^{(k)})$, for the standard IGA and surrogate IGA solutions corresponding to (8.1), respectively. Due to space limitations, we only analyze the convergence of the eigenvalues $\widetilde{\lambda}_h^{(k)} \to \lambda^{(k)}$.

Theorem 8.2. *Let $\theta > 1$. If $u^{(k)} \in H^{p+1}(\Omega)$, then there exists a constant C_8, depending only on p and $\boldsymbol{\varphi}$, such that*

$$\frac{|\lambda^{(k)} - \widetilde{\lambda}_h^{(k)}|}{\lambda^{(k)}} \leq C_8 h^{2p} + \theta \cdot (C_4 + C_7) H^{q+1}\,, \tag{8.3}$$

for every sufficiently small $H > 0$.

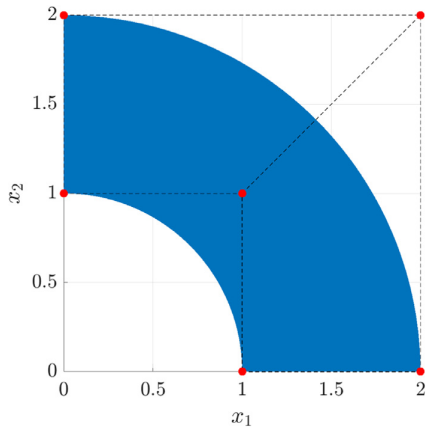


Fig. 8.1. The physical domain Ω for problems (8.1) and (9.1).

Proof. Clearly, $|\lambda^{(k)} - \widetilde{\lambda}_h^{(k)}| \leq |\lambda^{(k)} - \lambda_h^{(k)}| + |\lambda_h^{(k)} - \widetilde{\lambda}_h^{(k)}|$. It is known that if $u^{(k)} \in H^{p+1}(\Omega)$, then $|\lambda^{(k)} - \lambda_h^{(k)}| \leq C_8 \lambda^{(k)} h^{2p}$. We now focus on the term $|\lambda_h^{(k)} - \widetilde{\lambda}_h^{(k)}|$.

Let $V_h^{(k)}$ be the set of all k-dimensional subsets of V_h and fix $\theta > 1$. Two important members of this set are $E_h^{(k)} = \operatorname{span}\{u_h^{(l)}\}_{l=1}^k$ and $\widetilde{E}_h^{(k)} = \operatorname{span}\{\widetilde{u}_h^{(l)}\}_{l=1}^k$. Observe that

$$\begin{aligned} \widetilde{\lambda}_h^{(k)} &= \min_{E_h \in V_h^{(k)}} \max_{v \in E_h} \frac{\widetilde{a}(v,v)}{\widetilde{m}(v,v)} \leq \max_{v \in E_h^{(k)}} \frac{\widetilde{a}(v,v)}{\widetilde{m}(v,v)} = \max_{v \in E_h^{(k)}} \frac{a(v,v)}{m(v,v)} \frac{\widetilde{a}(v,v)}{a(v,v)} \frac{m(v,v)}{\widetilde{m}(v,v)} \\ &\leq \lambda_h^{(k)} \max_{v \in E_h^{(k)}} \frac{\widetilde{a}(v,v)}{a(v,v)} \max_{w \in E_h^{(k)}} \frac{m(w,w)}{\widetilde{m}(w,w)} . \end{aligned} \tag{8.4}$$

Note that $\frac{\widetilde{a}(v,v)}{a(v,v)} = 1 + \frac{\widetilde{a}(v,v) - a(v,v)}{a(v,v)} \leq 1 + C_4 H^{q+1}$, by Theorem 7.2. Similarly, by (8.2), $\frac{m(w,w)}{\widetilde{m}(w,w)} = 1 + \frac{\widetilde{m}(w,w) - m(w,w)}{\widetilde{m}(w,w)} \leq 1 + \theta \cdot C_7 H^{q+1}$, in the limit $H \to 0$. These observations, together with (8.4), imply

$$\widetilde{\lambda}_h^{(k)} \leq \lambda_h^{(k)} + \lambda_h^{(k)} (C_4 + \theta \cdot C_7) H^{q+1} , \tag{8.5a}$$

for every sufficiently small $H > 0$. Following a similar argument,

$$\lambda_h^{(k)} \leq \widetilde{\lambda}_h^{(k)} \frac{\widetilde{a}(v,v)}{\widetilde{a}(v,v)} \max_{w \in E_h^{(k)}} \frac{\widetilde{m}(w,w)}{m(w,w)} \leq \widetilde{\lambda}_h^{(k)} + \widetilde{\lambda}_h^{(k)} (\theta \cdot C_4 + C_7) H^{q+1} , \tag{8.5b}$$

for every sufficiently small $H > 0$. Note that (8.5a) implies that $\widetilde{\lambda}_h^{(k)} \leq \theta \lambda_h^{(k)}$, as $H \to 0$. After introducing this inequality into (8.5b), we arrive at the upper bound $|\lambda_h^{(k)} - \widetilde{\lambda}_h^{(k)}| \leq \lambda_h^{(k)} \theta \cdot (C_4 + C_7) H^{q+1}$, in the limit $H \to 0$. Because $\lambda_h^{(k)} \to \lambda^{(k)}$, we also have $\lambda_h^{(k)} \leq \theta \lambda^{(k)}$, as $H \to 0$. Only elementary algebra remains in order to arrive at (8.3). □

Remark 8.1. One upshot of Theorem 8.2 is that, if $H = \mathcal{O}(h)$, then one may wish to choose $q + 1 > 2p$, in order to recover an optimal spectral convergence rate. Of course, for irregular geometries, it is difficult to maintain the assumption $\lambda^{(j)} \in H^{p+1}(\Omega)$, and a lower q may still provide a useful approximation.

8.3. Numerical experiments

Our numerical experiments for problem (8.1) involve only the two-dimensional quarter annulus domain Ω, depicted in Fig. 8.1. This domain was chosen instead of Fig. 7.1(a) so that $u^{(k)} \in H^3(\Omega)$, for each k. Thus, when $p = 2$, Theorem 8.2 concludes that $|\lambda_h^{(k)} - \widetilde{\lambda}_h^{(k)}| \leq \mathcal{O}(h^4 + H^{q+1})$. Fig. 8.2, which shows the convergence of the first nine eigenvalues, for $p = 2$, verifies this result. Recall Remark 7.2. One again witnesses the parity

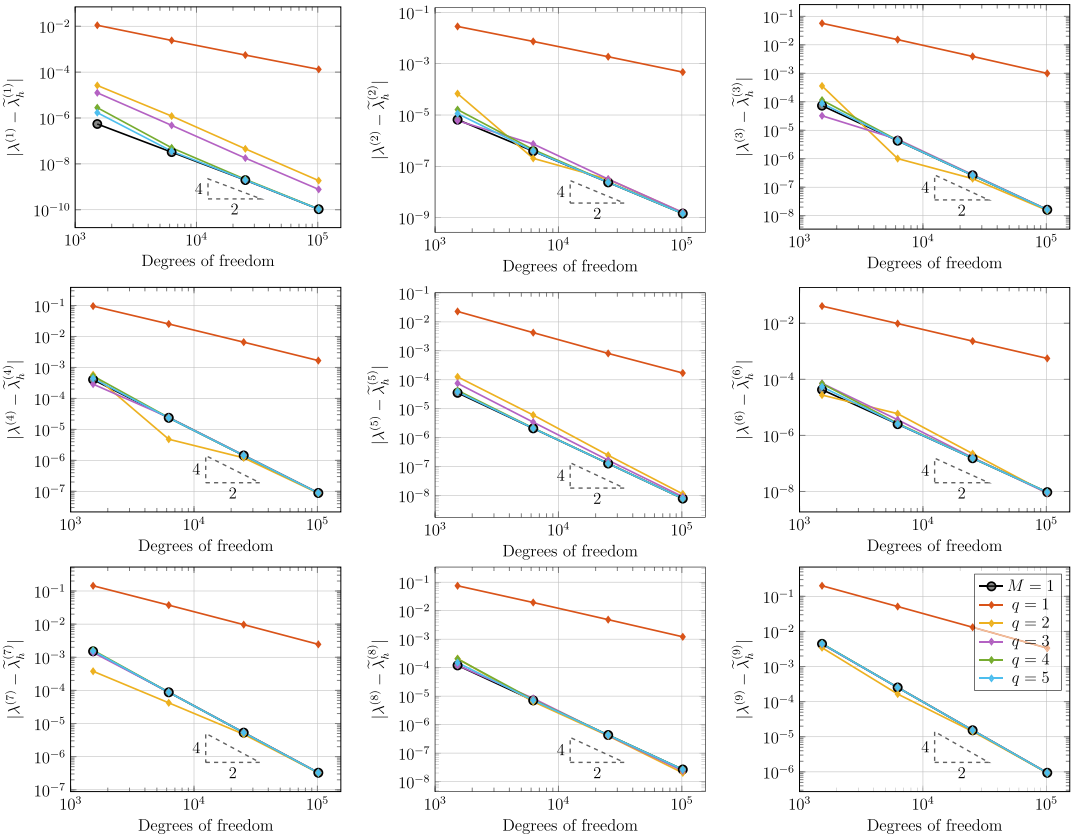


Fig. 8.2. Eigenvalue convergence plots. Quarter annulus geometry with $p = 2$ and the constant sampling length $M = 5$. Note that the "exact" solutions, $\lambda^{(k)}$, were taken from precomputed values from a high-order discretization on a much finer mesh.

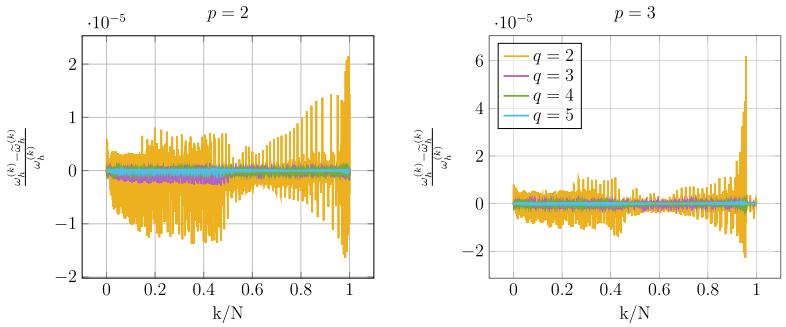


Fig. 8.3. Relative differences in the computed natural frequencies from the IGA and the surrogate IGA method for 50×50 control points and $M = 5$.

present in the L^2 error of the Poisson problems above. That is, we actually observe the stronger conclusion $|\lambda_h^{(k)} - \widetilde{\lambda}_h^{(k)}| = \mathcal{O}(h^4 + H^{q+2})$ when q is even.

A close inspection of Fig. 8.2 appears to indicate that the accuracy of the surrogate solution improves as the eigenvalues grow. This observation is in line with the previous numerical results for Poisson's equation, which showed nearly indistinguishable solutions for the "high-frequency" manufactured solution $u(\boldsymbol{x}) = \sin(20\,\pi\,x_1)\,\sin(20\,\pi\,x_2)$

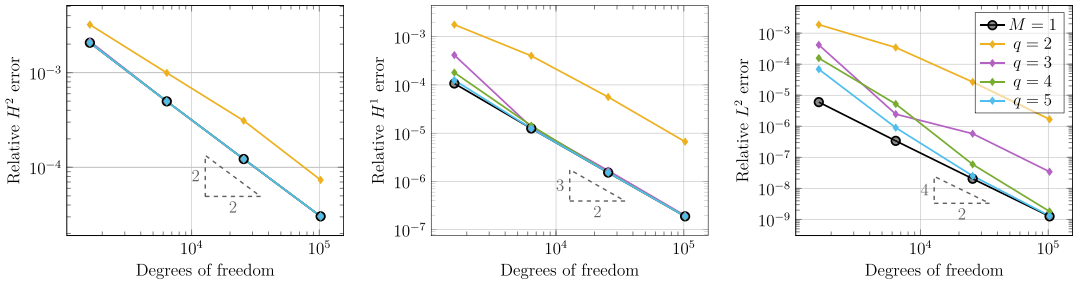


Fig. 9.1. Plate bending problem (9.1). Relative errors for $p = 3$, $M = 5$, and the manufactured solution $u(\boldsymbol{x}) = \sin(\pi\, x_1)\, \sinh(\pi\, x_2)$.

(cf. Figs. 7.2–7.4). Naturally, we should compare the accuracy of all eigenvalues computed with the standard IGA method to those coming from the surrogate IGA method. This is done, in part, in Fig. 8.3 for both $p = 2, 3$. Here, it is more meaningful to use the natural frequencies $(\omega^{(j)})^2 = \lambda^{(j)}$. Notice that the differences are extremely small across the entire range of computed frequencies.

9. Plate bending under a transverse load

Another clear advantage of IGA is the simplicity of discretizing high-order PDEs. In this section, we briefly demonstrate that the same features hold true for surrogate IGA methods. As a proof-of-concept, consider the simple Poisson–Kirchoff isotropic plate bending model. Given a function $f \in L^2(\Omega)$, the corresponding weak form is the following:

$$\text{Find } u \in H^2(\Omega) \cap H_0^1(\Omega) \text{ satisfying} \quad a(u, v) = F(v) \quad \text{for all } v \in H^2(\Omega) \cap H_0^1(\Omega), \tag{9.1}$$

where $a(u, v) = \int_\Omega \Delta u\, \Delta v \,\mathrm{d}x$ and $F(v) = \int_\Omega f\, v \,\mathrm{d}x$.

9.1. A higher-dimensional kernel

With the same principles as used for Poisson's equation, one may easily design a surrogate IGA method for (9.1). In our approach, the corresponding surrogate stiffness matrix $\widetilde{\mathsf{A}}$ was also defined using (5.1c). Notice that this definition does not preserve the entire kernel found in the true IGA stiffness matrix A. For instance, one may easily verify that all linear functions lie in the kernel of $a(\cdot, \cdot)$. Therefore, $\mathsf{A}\mathsf{c}^{(1)} = \mathsf{A}\mathsf{c}^{(2)} = 0$, where $\mathsf{c}^{(1)}, \mathsf{c}^{(2)}$ are the x_1- and x_2-coefficients of the control points, respectively. In our experiments, this property was only recovered in the limits $h \to 0$ or $H \to h$.

9.2. Numerical experiments

Let Ω be the quarter annulus domain depicted in Fig. 8.1. Fig. 9.1 shows the convergence of the errors, in the H^2, H^1, and L^2 norms, corresponding to this geometry Ω and the manufactured solution $u(\boldsymbol{x}) = \sin(\pi\, x_1)\, \sinh(\pi\, x_2)$. Even though the kernel is not preserved, the numerical results we witnessed are similar to those documented for the $p = 3$ experiments in the Poisson setting (see top row of Fig. 7.3). For instance, notice that the surrogate error with $q = 2$ is parallel to the reference IGA error ($M = 1$), in both the $H^1(\Omega)$ and $L^2(\Omega)$ norms.

Remark 9.1. Although we will not provide any rigorous analysis, if we recall Remark 7.1, the similarity between our Poisson results and those above may still appear somewhat surprising. Indeed, since only the zero row sum property is inherited in the surrogate $\widetilde{\mathsf{A}}$ when using (5.1c), we cannot improve on the upper bound in Lemma 7.1. Had the entire kernel of A been preserved in $\widetilde{\mathsf{A}}$, we conjecture that an optimal form of this bound would involve an h^{4-n} scaling factor. Such a factor should permit a surrogate solution $\widetilde{u}_h$ of two h-orders higher accuracy.

10. Stokes' flow

In this section, we consider a surrogate IGA discretization of Stokes' flow in a domain $\Omega \subseteq \mathbb{R}^n$. Given a viscosity $\mu \in \mathbb{R}_{>0}$, a function $\mathbf{f} \in \left[L^2(\Omega)\right]^n$, and a velocity field on the boundary $\mathbf{g} \in \left[H^{1/2}(\partial\Omega)\right]^n$, $\int_{\partial\Omega} \mathbf{g} \cdot \boldsymbol{n}\, \mathrm{d}s = 0$, the corresponding weak form is the following:

$$\begin{cases} \text{Find } \mathbf{u} \in [H^1(\Omega)]^n \text{ and } p \in L^2(\Omega)/\mathbb{R} \text{ satisfying } \operatorname{tr}\mathbf{u} = \mathbf{g} \text{ and} \\ a(\mathbf{u}, \mathbf{v}) + b(p, \mathbf{v}) + b(q, \mathbf{u}) = F(\mathbf{v}) \quad \text{for all } \mathbf{v} \in [H_0^1(\Omega)]^n \text{ and } q \in L^2(\Omega), \end{cases} \tag{10.1}$$

where $a(\mathbf{u}, \mathbf{v}) = \int_\Omega \mu \nabla\mathbf{u} \cdot \nabla\mathbf{v}\, \mathrm{d}x$, $b(p, \mathbf{v}) = \int_\Omega p\, \nabla \cdot \mathbf{v}\, \mathrm{d}x$, and $F(\mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v}\, \mathrm{d}x$. In this scenario, the pressure is not unique up to a constant, therefore we enforce the pressure to have zero mean value, i.e., $\int_\Omega p\, \mathrm{d}x = 0$.

10.1. Surrogate divergence matrices

Since no symmetry can be exploited, the surrogate divergence matrices B are constructed by employing definition (5.1a). Similarly, as in the mass term arising in Section 8, we have a surrogate reproduction property for the divergence form $b(q, \mathbf{u})$, when the geometry map is described by a polynomial. This property is formalized in the following corollary of Proposition 5.1.

Corollary 10.1. *Assume that the domain mapping $\boldsymbol{\varphi}\colon \widehat{\Omega} \to \Omega$ is defined through a polynomial of order p, i.e., $\boldsymbol{\varphi} \in \left[\mathcal{Q}_p(\widehat{\Omega})\right]^n$. Let B be the coefficient matrix arising from the discretization of $b(\cdot,\cdot)$ and $\widetilde{\mathsf{B}}$ the corresponding surrogate matrix. If $q \geq (n-1)p$, it holds that $\mathsf{B} = \widetilde{\mathsf{B}}$.*

Proof. In the following, we only consider the cases $n = 2$ and $n = 3$. Let $\boldsymbol{J}(\widehat{\boldsymbol{x}})$ be the Jacobian of $\boldsymbol{\varphi}(\widehat{\boldsymbol{x}})$. Assuming a gradient preserving transformation to the reference domain, the divergence $\nabla\cdot\mathbf{u}$ in the physical domain is transformed to $\operatorname{tr}(\boldsymbol{J}^{-1}\widehat{\nabla}\widehat{\mathbf{u}})$, where tr is the trace. Using the property $\boldsymbol{J}^{-1} = \det(\boldsymbol{J})^{-1}\operatorname{adj}(\boldsymbol{J})$, where $\operatorname{adj}(\boldsymbol{J})$ is the adjugate of $\boldsymbol{J}$, the bilinear form $b(\cdot,\cdot)$ may be written as

$$b(q, \mathbf{u}) = \int_\Omega p\, \nabla\cdot\mathbf{v}\, \mathrm{d}\boldsymbol{x} = \int_{\widehat{\Omega}} \widehat{p}\, \operatorname{tr}\left(\boldsymbol{J}^{-1}\widehat{\nabla}\widehat{\mathbf{u}}\right) \det(\boldsymbol{J})\, \mathrm{d}\widehat{\boldsymbol{x}} = \int_{\widehat{\Omega}} \widehat{p}\, \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{J})\widehat{\nabla}\widehat{\mathbf{u}}\right)\, \mathrm{d}\widehat{\boldsymbol{x}}.$$

It remains to show that $G(\widehat{\boldsymbol{x}}, \widehat{p}(\widehat{\boldsymbol{y}}), \widehat{\mathbf{u}}(\widehat{\boldsymbol{y}})) = \widehat{p}(\widehat{\boldsymbol{y}})\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{J}(\widehat{\boldsymbol{x}}))\widehat{\nabla}\widehat{\mathbf{u}}(\widehat{\boldsymbol{y}})\right)$ is a polynomial of degree $(n-1)p$ in the $\widehat{\boldsymbol{x}}$-variable. Applying Proposition 5.1 yields the desired reproduction property. The trace operator tr is linear, thus it suffices to analyze the entries of $\operatorname{adj}(\boldsymbol{J})$. In 2D, the components of $\operatorname{adj}(\boldsymbol{J})$ and $\boldsymbol{J}$ only differ by their position and sign. Since each component of $\boldsymbol{J}$ is an element of $\mathcal{Q}_p(\widehat{\Omega})$, we conclude that each component of $\operatorname{adj}(\boldsymbol{J})$ is also in $\mathcal{Q}_p(\widehat{\Omega})$. In 3D, the components of $\operatorname{adj}(\boldsymbol{J})$ are made up of determinants of 2×2 sub-matrices of $\boldsymbol{J}$. Taking the trace yields

$$\operatorname{tr}(\operatorname{adj}(\boldsymbol{J})) = \det\begin{pmatrix} \boldsymbol{J}_{22} & \boldsymbol{J}_{23} \\ \boldsymbol{J}_{32} & \boldsymbol{J}_{33} \end{pmatrix} + \det\begin{pmatrix} \boldsymbol{J}_{11} & \boldsymbol{J}_{13} \\ \boldsymbol{J}_{31} & \boldsymbol{J}_{33} \end{pmatrix} + \det\begin{pmatrix} \boldsymbol{J}_{11} & \boldsymbol{J}_{12} \\ \boldsymbol{J}_{21} & \boldsymbol{J}_{22} \end{pmatrix}.$$

For the first summand, we have $\boldsymbol{J}_{22} \in \mathcal{P}_p \otimes \mathcal{P}_{p-1} \otimes \mathcal{P}_p$, $\mathsf{J}_{23} \in \mathcal{P}_p \otimes \mathcal{P}_p \otimes \mathcal{P}_{p-1}$, $\boldsymbol{J}_{32} \in \mathcal{P}_p \otimes \mathcal{P}_{p-1} \otimes \mathcal{P}_p$, and $\boldsymbol{J}_{33} \in \mathcal{P}_p \otimes \mathcal{P}_p \otimes \mathcal{P}_{p-1}$. From this it follows that $\boldsymbol{J}_{22}\cdot\boldsymbol{J}_{33} \in \mathcal{P}_{2p} \otimes \mathcal{P}_{2p-1} \otimes \mathcal{P}_{2p-1}$ and $\boldsymbol{J}_{23}\cdot\boldsymbol{J}_{32} \in \mathcal{P}_{2p} \otimes \mathcal{P}_{2p-1} \otimes \mathcal{P}_{2p-1}$. This means that $\boldsymbol{J}_{22}\cdot\boldsymbol{J}_{33} - \boldsymbol{J}_{23}\cdot\boldsymbol{J}_{32} \in \mathcal{Q}_{2p}(\widehat{\Omega})$. Applying the same arguments to the other summands finally yields that $\operatorname{tr}(\operatorname{adj}(\boldsymbol{J})) \in \mathcal{Q}_{2p}(\widehat{\Omega})$. □

In order to discretize (10.1), an inf–sup stable space pair is required. For this purpose, we choose the isogeometric *subgrid* element as described in [37]. In this discretization, the velocity field is defined on a subgrid of the pressure where each pressure element is subdivided into 2^n elements. This allows for using a velocity space of order $p+1$ with $C^p(\widehat{\Omega})$ regularity and a pressure space of order p with $C^{p-1}(\widehat{\Omega})$ regularity.

We do not provide *a priori* error estimates for the Stokes problem, but in the case that the divergence matrix is reproduced one would follow similar arguments as presented for the Poisson problem. In the scenario where the divergence matrix is not reproduced, further work is necessary. However, the results in the next subsection suggest that the reproduction is not required in order to obtain an optimal order of convergence.

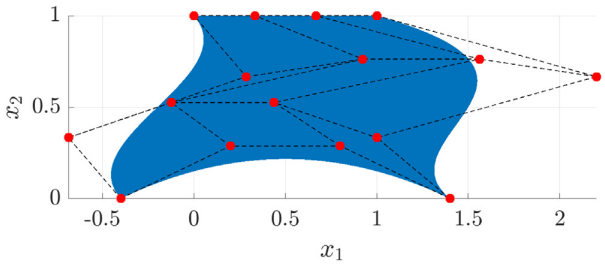


Fig. 10.1. The computational domain Ω used in the Stokes flow problem (10.1).

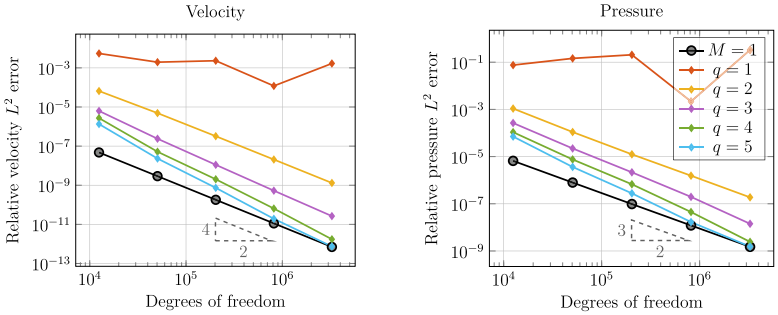


Fig. 10.2. Stokes' flow problem (10.1) with manufactured solution (10.2). The velocity is discretized with $p = 3$ and the pressure with $p = 2$. Relative L^2 velocity and pressure errors for $M = 5$.

10.2. Numerical experiments

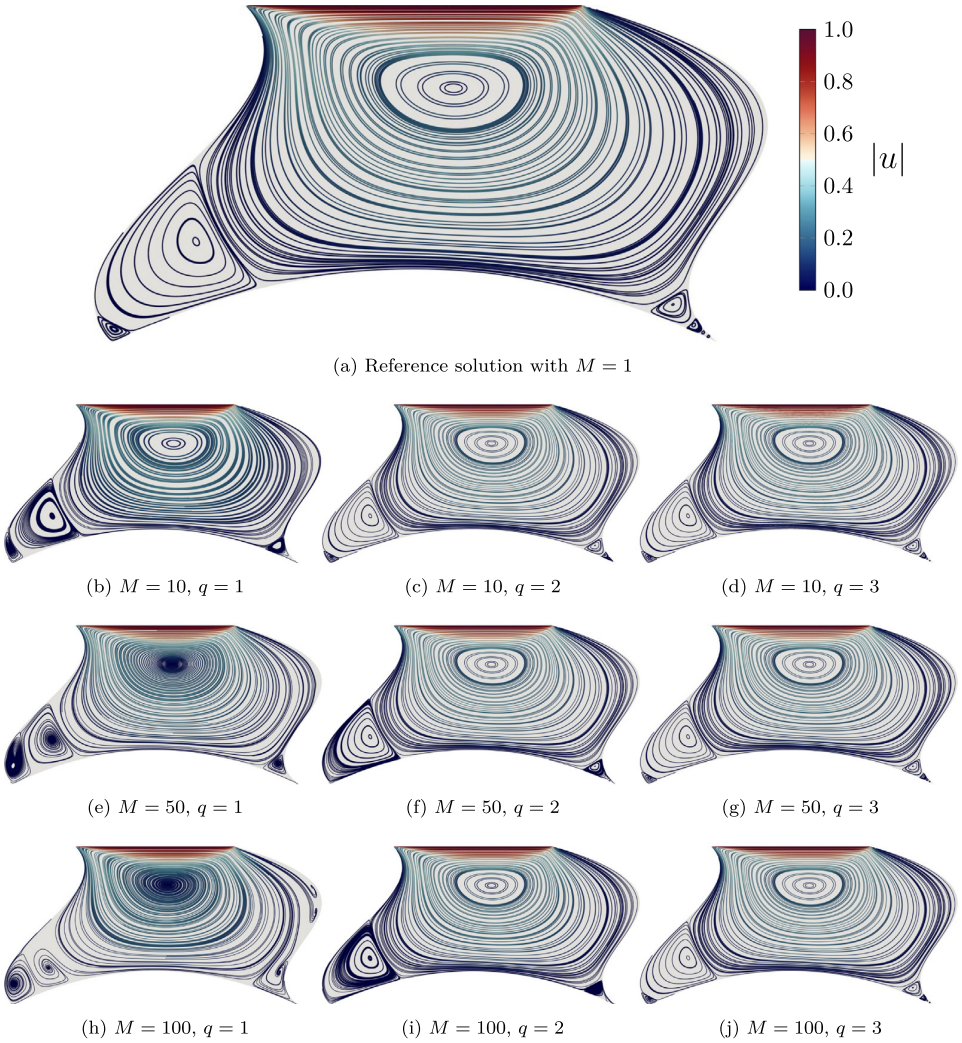
Our computational study of Stokes' flow is comprised of two separate experiments. In the first experiment, we provide a smooth manufactured solution in order to investigate convergence rates. In the second example, we consider a lid-driven cavity benchmark problem. Due to the discontinuous boundary conditions, the solution of this problem has singularities at two corners of the domain. Both examples are computed on the domain shown in Fig. 10.1 which was constructed by a Coons patch with a cubic boundary parameterization. We discretize the problem using the aforementioned subgrid element with a third order velocity and second order pressure. The viscosity is set to $\mu = 1$ for all scenarios.

In the first example, the manufactured solution is chosen to be

$$\mathbf{u}(x, y) = \begin{bmatrix} \frac{\sin(x)\cos(y)}{x+1} \\ \frac{((x+1)\cos(x) - \sin(x))\sin(y)}{(x+1)^2} \end{bmatrix}, \quad p(x, y) = y\sinh(x) + C_p. \tag{10.2}$$

Note that this solution satisfies $\nabla \cdot \mathbf{u} = 0$ and the constant $C_p \in \mathbb{R}$ is chosen such that the pressure mean is zero. The Dirichlet boundary condition $\mathbf{g}$ and the right hand side $\mathbf{f}$ are set accordingly to match the manufactured solution. In Fig. 10.2, we present convergence plots for the velocity and pressure separately for different surrogate orders q and fixed $M = 5$. For reference, we also include the standard discretization with $M = 1$ in these plots. We observe the expected convergence orders for all $q \geq 2$. In agreement with Corollary 10.1, the divergence matrices were perfectly reproduced for every $q \geq 3$.

In the second example, we consider a lid-driven cavity benchmark on the domain Fig. 10.1 where the fluid is driven on the top edge by constant velocity $\mathbf{g} = (1, 0)^\top$ and we assume no-slip boundary conditions $\mathbf{g} = \mathbf{0}$ on the remaining parts of the boundary. The degrees of freedom corresponding to the nodal basis functions in the top left and top right corner are set to zero. Furthermore, the volume forces are neglected, i.e., $\mathbf{f} = \mathbf{0}$. In Fig. 10.3(a), we show the velocity streamlines which were computed using a standard IGA approach on a mesh with 320×320 control points. The effect of different surrogate approaches on the velocity streamlines may be observed in Figs. 10.3(b)–10.3(j). In the case $q = 3$, where the divergence matrices are in fact reproduced, the streamlines show the same behavior as in the standard approach even for $M = 100$. For other values of q and M, the streamline behavior is different, but the streamlines are getting closer to the reference solution the larger q and



(a) Reference solution with $M = 1$

(b) $M = 10$, $q = 1$ (c) $M = 10$, $q = 2$ (d) $M = 10$, $q = 3$

(e) $M = 50$, $q = 1$ (f) $M = 50$, $q = 2$ (g) $M = 50$, $q = 3$

(h) $M = 100$, $q = 1$ (i) $M = 100$, $q = 2$ (j) $M = 100$, $q = 3$

Fig. 10.3. Lid-driven cavity benchmark velocity streamlines. The velocity is discretized with $p = 3$ and the pressure with $p = 2$ using surrogate approaches with varying M and q. The mesh is discretized by 320×320 control points. Note that actually using an interpolation order of $q = 1, 2, 3$ is still probably not recommended for standard practice since more accurate results should be expected with $q = 5$ while taking roughly the same time.

the smaller M becomes. We note that actually using an interpolation order of $q = 1, 2, 3$ in computation is still probably not recommended for standard practice. For instance, assembly using $q = 5$ took roughly the same time as either of these lower-order choices and, in this case, the surrogate solution $(\widetilde{\mathbf{u}}_h, \widetilde{p}_h)$ should be expected to be even more accurate.

11. Conclusion

This is the second in a series of articles on the surrogate matrix methodology [24,28], but only the first to consider its applications in isogeometric analysis. The companion article [28] can be used as a reference for the implementation of the surrogate matrix methodology in existing IGA softwares which employ element-based quadrature. In the present article, we considered a number of linear and static problems; namely, Poisson's equation, membrane vibration, plate bending, and Stokes' flow. In [24], (static) anisotropic variable–coefficient diffusion,

linearized elasticity, and (dynamic) p-Laplacian diffusion problems were studied in the lowest order finite element setting.

Acknowledgments

The authors wish to thank each of the referees for the detailed feedback they gave during the review process. Their insights significantly improved the quality of the final manuscript. In particular, we wish to acknowledge the careful reading of the second referee who pointed out a mistake in our original proof of This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 800898. This work was also partly supported by the German Research Foundation, Germany through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) and by grant WO671/11-1.

Appendix. Marsden's identity

The purpose of this appendix is to substantiate some of the claims made in Section 3.3, as well as provide a complete proof of Theorem 4.2. In turn, we adopt all of the notation and assumptions introduced in Section 2.2. We begin by stating Theorem A.1, the proof of which can be found in [38]; see also [39–41].

Theorem A.1 (*Marsden*). *Let* $\psi_k(\widehat{y}) = (\xi_{k+1} - \widehat{y}) \cdots (\xi_{k+p} - \widehat{y})$. *For any* $\widehat{x}, \widehat{y} \in [0, 1]$,

$$(\widehat{x} - \widehat{y})^p = \sum_{k=1}^{m} b_k(\widehat{x}) \psi_k(\widehat{y}) \,.$$

This theorem allows us to conclude that the expression (3.9) is valid and, moreover, $w \in \mathcal{Q}_p(\widetilde{\Omega}_{\mathbf{0}})$. This is shown in two steps.

Corollary A.2. *Let* $\Psi(\widetilde{\mathbf{y}}) = \prod_{i=1}^{n} \prod_{j=0}^{p-1} \left(\frac{j}{m-p} - \frac{p-1}{2(m-p)} - \widetilde{y}_i \right)$ *and* $\boldsymbol{p} = (p, \ldots, p) \in \mathbb{N}^n$. *Then, for any* $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}} \in \widetilde{\Omega}_{\mathbf{0}}$,

$$(\widetilde{\mathbf{x}} - \widetilde{\mathbf{y}})^{\boldsymbol{p}} = \sum_{\widetilde{\mathbf{x}}_i \in \widetilde{\mathbb{X}}} \widehat{B}(\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_i) \, \Psi(\widetilde{\mathbf{y}} - \widetilde{\mathbf{x}}_i) \,. \tag{A.1}$$

Proof. Observe that, for each $k = p+1, \ldots, m-p$, it holds that $\widetilde{x}^{(k)} + h = \widetilde{x}^{(k+1)} = \xi_{k+1} + (p+1) \cdot h/2$. Recall that $k = p+1, \ldots, m-p$ are exactly the indices of the cardinal B-splines $b_k(\widehat{x}) = b(\widehat{x} - \widetilde{x}^{(k)})$. Therefore, by Theorem A.1, we see that

$$(\widetilde{x}_i - \widetilde{y}_i)^p = \sum_{k=p+1}^{m-p} b(\widetilde{x}_i - \widetilde{x}^{(k)}) \prod_{j=0}^{p-1} \left(\widetilde{x}^{(k)} + \left(j - \frac{p-1}{2} \right) \cdot h - \widetilde{y}_i \right),$$

for each $i = 1, \ldots, n$. The result now follows from the definitions of $\widehat{B}$ and Ψ. □

Corollary A.2 can be used to write out an elegant expression for any polynomial in $\mathcal{Q}_p(\widetilde{\Omega}_{\mathbf{0}})$, in terms of cardinal B-splines. Indeed, let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ be a multi-index, let f be an arbitrary polynomial in $\mathcal{Q}_p(\widetilde{\Omega}_{\mathbf{0}})$, and let $D^{\boldsymbol{\alpha}}$ be the $\boldsymbol{\alpha}$-derivative operator, in the variable $\widetilde{\mathbf{y}}$. By Taylor's Theorem, it holds that

$$f(\widetilde{\mathbf{x}}) = \sum_{i=1}^{n} \sum_{\alpha_i \leq p} \frac{D^{\boldsymbol{\alpha}} f(\widetilde{\mathbf{y}})}{\boldsymbol{\alpha}!} (\widetilde{\mathbf{x}} - \widetilde{\mathbf{y}})^{\boldsymbol{\alpha}} \,, \tag{A.2}$$

for every $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}} \in \widetilde{\Omega}_{\mathbf{0}}$. Next, applying $D^{\boldsymbol{p}-\boldsymbol{\alpha}}$ to both sides of (A.1), we find that

$$\frac{(-1)^{|\boldsymbol{p}-\boldsymbol{\alpha}|} \boldsymbol{p}!}{\boldsymbol{\alpha}!} (\widetilde{\mathbf{x}} - \widetilde{\mathbf{y}})^{\boldsymbol{\alpha}} = \sum_{\widetilde{\mathbf{x}}_i \in \widetilde{\mathbb{X}}} \widehat{B}(\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_i) D^{\boldsymbol{p}-\boldsymbol{\alpha}} \Psi(\widetilde{\mathbf{y}} - \widetilde{\mathbf{x}}_i) \,. \tag{A.3}$$

Together, (A.2) and (A.3) imply $f(\widetilde{\mathbf{x}}) = \sum_{\widetilde{\mathbf{x}}_i \in \widetilde{\mathbb{X}}} \widehat{B}(\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_i)(Lf)(\widetilde{\mathbf{x}}_i)$, where

$$(Lf)(\widetilde{\mathbf{x}}) = \sum_{i=1}^{n} \sum_{\alpha_i \leq p} \frac{(-D)^{\boldsymbol{p}-\boldsymbol{\alpha}} \Psi(\widetilde{\mathbf{y}} - \widetilde{\mathbf{x}})}{\boldsymbol{p}!} D^{\boldsymbol{\alpha}} f(\widetilde{\mathbf{y}}) \,.$$

We may now state our second corollary of Theorem A.1.

Corollary A.3. *Let $W(\widehat{\boldsymbol{x}}) = \sum_{j=1}^{N} w_j \widehat{B}_j(\widehat{\boldsymbol{x}})$. If $W \in \mathcal{Q}_p(\widehat{\Omega})$, then there exists a polynomial $w \in \mathcal{Q}_p(\widetilde{\Omega}_{\mathbf{0}})$ such that $w_i = w(\widetilde{\boldsymbol{x}}_i)$, for each $\widetilde{\boldsymbol{x}}_i \in \widetilde{\mathbb{X}}$. Moreover, there exists a constant C, depending only on p, such that*

$$\|w\|_{W^{r,\infty}(\widetilde{\Omega}_{\mathbf{0}})} \leq C \|W\|_{W^{r,\infty}(\widehat{\Omega})} \qquad \text{for all } r \geq p\,. \tag{A.4}$$

Proof. Clearly, $w = LW \in \mathcal{Q}_p(\widetilde{\Omega}_{\mathbf{0}})$. Therefore, one readily determines that $\|w\|_{L^\infty(\widetilde{\Omega}_{\mathbf{0}})} \leq C \|W\|_{W^{p,\infty}(\widetilde{\Omega}_{\mathbf{0}})}$, for some constant C, depending only on p. Due to the equivalence of norms on finite dimensional vector spaces (note that the dimension of $\mathcal{Q}_p(\widetilde{\Omega}_{\mathbf{0}})$ depends only on p) and the fact $\widetilde{\Omega}_{\mathbf{0}} \subseteq \overline{\widehat{\Omega}}$, we immediately arrive at (A.4). □

We may now complete the proof of Theorem 4.2.

Proof of Theorem 4.2. Let $\widetilde{\boldsymbol{x}} \in \widetilde{\Omega}$ be arbitrary. Then, for any $\boldsymbol{\delta} \in \mathcal{D}$, we have that $\widetilde{\boldsymbol{x}} + \boldsymbol{\delta} \in \widetilde{\Omega}_{\mathbf{0}}$. Moreover, for every multi-index $|\boldsymbol{\alpha}| = r$, the product rule can be used to show that

$$\begin{aligned} D^{\boldsymbol{\alpha}} \Phi_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}) &= D^{\boldsymbol{\alpha}} \left[w(\widetilde{\boldsymbol{x}}) w(\widetilde{\boldsymbol{x}} + \boldsymbol{\delta}) \int_{\widehat{\omega}_{\boldsymbol{\delta}}} \widehat{\nabla} \left(\frac{\widehat{B}(\widehat{\boldsymbol{y}})}{W(\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}})} \right)^{\top} K(\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}}) \, \widehat{\nabla} \left(\frac{\widehat{B}_{\boldsymbol{\delta}}(\widehat{\boldsymbol{y}})}{W(\widetilde{\boldsymbol{x}} + \widehat{\boldsymbol{y}})} \right) \mathrm{d}\widehat{\boldsymbol{y}} \right] \\ &\leq C \cdot \|w\|^2_{W^{r,\infty}(\widetilde{\Omega}_{\mathbf{0}})} \|K\|_{W^{r,\infty}(\widehat{\Omega})} \|W\|^2_{W^{r,\infty}(\widehat{\Omega})} \|\widehat{\nabla} W\|^2_{W^{r,\infty}(\widehat{\Omega})} \\ &\quad \cdot \left(\|\widehat{\nabla} \widehat{B} \cdot \widehat{\nabla} \widehat{B}_{\boldsymbol{\delta}}\|_{L^1(\widehat{\omega}_{\boldsymbol{\delta}})} + \|\widehat{\nabla} \widehat{B} \cdot \widehat{B}_{\boldsymbol{\delta}}\|_{L^1(\widehat{\omega}_{\boldsymbol{\delta}})} + \|\widehat{\nabla} \widehat{B}_{\boldsymbol{\delta}} \cdot \widehat{B}\|_{L^1(\widehat{\omega}_{\boldsymbol{\delta}})} + \|\widehat{B} \cdot \widehat{B}_{\boldsymbol{\delta}}\|_{L^1(\widehat{\omega}_{\boldsymbol{\delta}})} \right), \end{aligned}$$

for some C depending only on $\boldsymbol{\alpha}$. Since both functions K, and W are determined by the choice of Ω, A.3 and a scaling argument show that $\|D^{\boldsymbol{\alpha}} \Phi_{\boldsymbol{\delta}}\|_{L^\infty(\widetilde{\Omega})} \leq C h^{n-2}$, where C now depends on p, $\boldsymbol{\alpha}$, and $\boldsymbol{\varphi}$. Lemma 4.1 now completes the proof. □

References

[1] T. Butler, P. Constantine, T. Wildey, A posteriori error analysis of parameterized linear systems using spectral methods, SIAM J. Matrix Anal. Appl. 33 (1) (2012) 195–209.

[2] S.A. Mattis, B. Wohlmuth, Goal-oriented adaptive surrogate construction for stochastic inversion, Comput. Methods Appl. Mech. Engrg. 339 (2018) 36–60.

[3] T.J. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39–41) (2005) 4135–4195.

[4] J.A. Cottrell, T.J. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, John Wiley & Sons, 2009.

[5] K. Hollig, Finite Element Methods with B-splines, vol. 26, Siam, 2003.

[6] T.J. Hughes, G. Sangalli, Mathematics of isogeometric analysis: a conspectus, in: Encyclopedia of Computational Mechanics, second ed., Wiley Online Library, 2018, pp. 1–40.

[7] R.R. Hiemstra, F. Calabro, D. Schillinger, T.J. Hughes, Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 316 (2017) 966–1004.

[8] A. Bressan, S. Takacs, Sum factorization techniques in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 352 (2019) 437–460.

[9] G. Sangalli, M. Tani, Matrix-free weighted quadrature for a computationally efficient isogeometric k-method, Comput. Methods Appl. Mech. Engrg. 338 (2018) 117–133.

[10] A. Mantzaflaris, B. Jüttler, B.N. Khoromskij, U. Langer, Low rank tensor methods in Galerkin-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 316 (2017) 1062–1085.

[11] C. Hofreither, A black-box low-rank approximation algorithm for fast matrix assembly in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 333 (2018) 311–330.

[12] P. Antolin, A. Buffa, F. Calabró, M. Martinelli, G. Sangalli, Efficient matrix computation for tensor-product isogeometric analysis: the use of sum factorization, Comput. Methods Appl. Mech. Engrg. 285 (2015) 817–828.

[13] A. Mantzaflaris, B. Jüttler, Integration by interpolation and look-up for Galerkin-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 284 (2015) 373–400.

[14] A. Mantzaflaris, B. Jüttler, Exploring matrix generation strategies in isogeometric analysis, in: M. Floater, T. Lyche, M.-L. Mazure, K. Mørken, L.L. Schumaker (Eds.), Mathematical Methods for Curves and Surfaces, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 364–382.

[15] F. Calabró, G. Sangalli, M. Tani, Fast formation of isogeometric Galerkin matrices by weighted quadrature, in: Special Issue on Isogeometric Analysis: Progress and Challenges, Comput. Methods Appl. Mech. Engrg. 316 (2017) 606–622.

[16] F. Fahrendorf, L.D. Lorenzis, H. Gomez, Reduced integration at superconvergent points in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 328 (2018) 390–410.

[17] T.J. Hughes, A. Reali, G. Sangalli, Efficient quadrature for NURBS-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 199 (5–8) (2010) 301–313.
[18] F. Auricchio, F. Calabro, T.J. Hughes, A. Reali, G. Sangalli, A simple algorithm for obtaining nearly optimal quadrature rules for nurbs-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 249 (2012) 15–27.
[19] R.R. Hiemstra, G. Sangalli, M. Tani, F. Calabro, T.J.R. Hughes, Fast formation and assembly of finite element matrices with application to isogeometric linear elasticity, ICES Report 19-03, The University of Texas at Austin, 2019.
[20] L.B. Da Veiga, A. Buffa, G. Sangalli, R. Vázquez, Mathematical analysis of variational isogeometric methods, Acta Numer. 23 (2014) 157–287.
[21] S. Bauer, M. Mohr, U. Rüde, J. Weismüller, M. Wittmann, B. Wohlmuth, A two-scale approach for efficient on-the-fly operator assembly in massively parallel high performance multigrid codes, Appl. Numer. Math. 122 (2017) 14–38.
[22] S. Bauer, M. Huber, S. Ghelichkhan, M. Mohr, U. Rüde, B. Wohlmuth, Large-scale simulation of mantle convection based on a new matrix-free approach, J. Comput. Sci. 31 (2019) 60–76.
[23] S. Bauer, M. Huber, M. Mohr, U. Rüde, B. Wohlmuth, A new matrix-free approach for large-scale geodynamic simulations and its performance, in: International Conference on Computational Science, Springer, 2018, pp. 17–30.
[24] D. Drzisga, B. Keith, B. Wohlmuth, The surrogate matrix methodology: a priori error estimation, SIAM J. Sci. Comput. 41 (6) (2019) A3806–A3838.
[25] A. Karatarakis, P. Karakitsios, M. Papadrakakis, GPU Accelerated computation of the isogeometric analysis stiffness matrix, Comput. Methods Appl. Mech. Engrg. 269 (2014) 334–355.
[26] C. de Falco, A. Reali, R. Vázquez, GeoPDEs: a research tool for isogeometric analysis of PDEs, Adv. Eng. Softw. 42 (12) (2011) 1020–1034.
[27] R. Vázquez, A new design for the implementation of isogeometric analysis in Octave and Matlab: GeoPDEs 3.0, Comput. Math. Appl. 72 (3) (2016) 523–554.
[28] D. Drzisga, B. Keith, B. Wohlmuth, The surrogate matrix methodology: a reference implementation for low-cost assembly in isogeometric analysis, 2019, arXiv preprint arXiv:1909.04029.
[29] L. Piegl, W. Tiller, The NURBS Book, Springer Science & Business Media, 2012.
[30] I.J. Schoenberg, Cardinal Spline Interpolation, vol. 12, Siam, 1973.
[31] I.J. Schoenberg, Contributions to the problem of approximation of equidistant data by analytic functions. part A. on the problem of smoothing or graduation. a first class of analytic approximation formulae, Quart. Appl. Math. 4 (1946) 45–99.
[32] I.J. Schoenberg, Contributions to the problem of approximation of equidistant data by analytic functions. part B. on the problem of osculatory interpolation. a second class of analytic approximation formulae, Quart. Appl. Math. 4 (1946) 112–141.
[33] H.-J. Bungartz, M. Griebel, Sparse grids, Acta Numer. 13 (2004) 147–269.
[34] W. Dahmen, R. De Vore, K. Scherer, Multidimensional spline approximation, SIAM J. Numer. Anal. 17 (3) (1980) 380–402.
[35] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python, 2001, http://www.scipy.org. Online (Accessed 11 February 2019).
[36] P.J. Davis, P. Rabinowitz, Methods of Numerical Integration, Courier Corporation, 2007.
[37] A. Bressan, G. Sangalli, Isogeometric discretizations of the Stokes problem: stability analysis by the macroelement technique, IMA J. Numer. Anal. 33 (2) (2013) 629–651.
[38] C. De Boor, B(asic)-spline basics, Technical report, UW–Madison Mathematics Research Center, 1986.
[39] C. De Boor, K. Höllig, S. Riemenschneider, Box Splines, vol. 98, Springer Science & Business Media, 1993.
[40] W. Dahmen, N. Dyn, D. Levin, On the convergence rates of subdivision algorithms for box spline surfaces, Constr. Approx. 1 (1) (1985) 305–322.
[41] W. Dahmen, C.A. Micchelli, Translates of multivariate splines, Linear Algebra Appl. 52 (1983) 217–234.

A.3. The surrogate matrix methodology: Accelerating isogeometric analysis of waves

The surrogate matrix methodology: Accelerating isogeometric analysis of waves

Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth

Following the successful application of the surrogate matrix methodology to accelerate matrix assembly for IGA in core article II (Appendix A.2), we extend the method to other applications targeting the analysis of waves. In particular, this work augments the previous article by considering multi-patch discretizations of time-harmonic, transient, and nonlinear PDEs as particular applications of the methodology. In this paper, the methodology is applied to several model problems in wave mechanics treated in the Bubnov–Galerkin isogeometric setting.

In order to analyze the method theoretically, we consider the Helmholtz equation equation as a model problem. We present an a priori error analysis for this scenario and demonstrate that the additional consistency error introduced by the utilization of surrogate matrices is independent of the wave number. Additionally, we conduct a floating point complexity analysis which establishes that the computational complexity of the methodology compares favorably to other present fast assembly techniques for isogeometric methods. In the numerical examples, we verify the theoretical error estimate for the Helmholtz equation. Furthermore, we consider a time-harmonic elastodynamic wave problem with PML absorbing boundary conditions. In a final example, we consider a transient nonlinear hyperelastic wave propagation example involving multiple patches. Here, the material is modeled by a compressible neo-Hookean material. The nonlinearity is resolved using Newton–Raphson's method which is relatable to a quasi-Newton method in the presence of surrogate matrices. Moreover, the time is discretized using a generalized-α time stepping scheme. We present evidence of an improved performance based on feasibility studies with the MATLAB software library GeoPDEs [51, 102].

In Section 2, we introduce the various equations of interest and present certain properties of their discretization which are required for the subsequent sections. In Section 3, we recall the essential features of the surrogate matrix methodology in the context of the Helmholtz equation and its extension to elastic waves. In Section 4, we present an a priori error analysis for the Helmholtz equation for different cases of boundary conditions. In addition, we point out particular aspects of the surrogate method in the presence of PML and in transient and nonlinear problems. In Section 5, we briefly remark on our implementation and its difference to the implementation presented in [41, 43]. Furthermore, we establish its FLOP computational complexity. In Section 6, we provide computational evidence for the performance benefits of the methodology with respect to accuracy and run-time. Notable speed-ups can be observed for large wave numbers without any loss in accuracy. Finally, in Section 7, we provide some concluding remarks.

I was significantly involved in finding the ideas and primarily responsible for setting up the mathematical framework and carrying out the scientific work presented in this article. Furthermore, I was in charge of writing the article while the co-authors contributed by making corrective changes.

Permission to include:

Daniel Drzisga, Brendan Keith, and Barbara Wohlmuth
The surrogate matrix methodology: Accelerating isogeometric analysis of waves
Computer Methods in Applied Mechanics and Engineering 372 (2020): 113322
(see also article [42] in the bibliography)

The following pages on copyright are excerpts from copies of the website

https://www.elsevier.com/about/policies/copyright#Author-rights

retrieved on 22 March 2020.

Copyright

Describes the rights related to the publication and distribution of research. It governs how authors (as well as their employers or funders), publishers and the wider general public can use, publish and distribute articles or books.

Journal author rights Government employees Elsevier's rights Protecting author rights Open access

Journal author rights

In order for Elsevier to publish and disseminate research articles, we need publishing rights. This is determined by a publishing agreement between the author and Elsevier. This agreement deals with the transfer or license of the copyright to Elsevier and authors retain significant rights to use and share their own published articles. Elsevier supports the need for authors to share, disseminate and maximize the impact of their research and these rights, in Elsevier proprietary journals* are defined below:

For subscription articles	For open access articles
Authors transfer copyright to the publisher as part of a journal publishing agreement, but have the right to: • Share their article for Personal Use, Internal Institutional Use and Scholarly Sharing purposes, with a DOI link to the version of record on ScienceDirect (and with the Creative Commons CC-BY-NC- ND license for author manuscript versions) • Retain patent, trademark and other intellectual property rights (including research data). • Proper attribution and credit for the published work.	Authors sign an exclusive license agreement, where authors have copyright but license exclusive rights in their article to the publisher**. In this case authors have the right to: • Share their article in the same ways permitted to third parties under the relevant user license (together with Personal Use rights) so long as it contains a CrossMark logo, the end user license, and a DOI link to the version of record on ScienceDirect. • Retain patent, trademark and other intellectual property rights (including research data). • Proper attribution and credit for the published work.

***Please note that society or third party owned journals may have different publishing agreements. Please see the journal's guide for authors for journal specific copyright information.**

****This includes the right for the publisher to make and authorize commercial use, please see "Rights granted to Elsevier" for more details.**

Help and Support

- Download a sample publishing agreement for subscription articles in ↓ English and ↓ French.
- Download a sample publishing agreement for open access articles for authors choosing a ↓ commercial user license and ↓ non-commercial user license.
- For authors who wish to self-archive see our sharing guidelines
- See our author pages for further details about how to promote your article.
- For use of Elsevier material not defined below please see our permissions page or visit the Permissions Support Center ↗ .

For Elsevier proprietary journals the following steps apply:

1. Authors sign a publishing agreement where they will have copyright but grant broad publishing and distribution rights to the publisher, including the right to publish the article on Elsevier's online platforms.
2. The author chooses an end user license under which readers can use and share the article.
3. The publisher makes the article available online with the author's choice of end user license.

Quick definitions

Personal use

Authors can use their articles, in full or in part, for a wide range of scholarly, non-commercial purposes as outlined below:

- Use by an author in the author's classroom teaching (including distribution of copies, paper or electronic)
- Distribution of copies (including through e-mail) to known research colleagues for their personal use (but not for Commercial Use)
- Inclusion in a thesis or dissertation (provided that this is not to be published commercially)
- Use in a subsequent compilation of the author's works
- Extending the Article to book-length form
- Preparation of other derivative works (but not for Commercial Use)
- Otherwise using or re-using portions or excerpts in other works

These rights apply for all Elsevier authors who publish their article as either a subscription article or an open access article. In all cases we require that all Elsevier authors always include a full acknowledgement and, if appropriate, a link to the final published version hosted on Science Direct.

Commercial use

This is defined as the use or posting of articles:

- **For commercial gain without a formal agreement with the publisher.**
  - For example by associating advertising with the full-text of the article, by providing hosting services to other repositories or to other organizations (including where an otherwise non-commercial site or repository provides a service to other organizations or agencies), or charging fees for document delivery or access
- **To substitute for the services provided directly by the journal.**
  - For example article aggregation, systematic distribution of articles via e-mail lists or share buttons, posting, indexing, or linking for promotional/marketing activities, by commercial companies for use by customers and intended target audiences of such companies (e.g. pharmaceutical companies and healthcare professionals/physician-prescribers).

If you would like information on how to obtain permission for such uses click here or if you would like to make commercial use of the article please visit the Permissions Support Center ↗ .

Internal institutional use

- Use by the author's institution for classroom teaching at the institution and for internal training purposes (including distribution of copies, paper or electronic, and use in coursepacks and courseware programs, but not in Massive Open Online Courses)
- Inclusion of the Article in applications for grant funding
- For authors employed by companies, the use by that company for internal training purposes

Notice of publication and copyright

First Published in “The surrogate matrix methodology: Accelerating isogeometric analysis of waves” in Computer Methods in Applied Mechanics and Engineering 372 (2020), published by Elsevier B.V.

DOI: https://doi.org/10.1016/j.cma.2020.113322

Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 372 (2020) 113322
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

The surrogate matrix methodology: Accelerating isogeometric analysis of waves

Daniel Drzisga*, Brendan Keith, Barbara Wohlmuth

Lehrstuhl für Numerische Mathematik, Fakultät für Mathematik (M2), Technische Universität München, Garching bei München, Germany

Received 10 April 2020; received in revised form 1 July 2020; accepted 28 July 2020
Available online 18 August 2020

Abstract

The surrogate matrix methodology delivers low-cost approximations of matrices (i.e., surrogate matrices) which are normally computed in Galerkin methods via element-scale quadrature formulas. In this paper, the methodology is applied to a number of model problems in wave mechanics treated in the Galerkin isogeometric setting. Herein, the resulting surrogate methods are shown to significantly reduce the assembly time in high frequency wave propagation problems. In particular, the assembly time is reduced with negligible loss in solution accuracy. This paper also extends the scope of previous articles in its series by considering multi-patch discretizations of time-harmonic, transient, and nonlinear PDEs as particular use cases of the methodology. Our a priori error analysis for the Helmholtz equation demonstrates that the additional consistency error introduced by the presence of surrogate matrices is *independent of the wave number*. In addition, our floating point analysis establishes that the computational complexity of the methodology compares favorably to other contemporary fast assembly techniques for isogeometric methods. Our numerical experiments demonstrate clear performance gains for time-harmonic problems, both with and without the presence of perfectly matched layers. Notable speed-ups are also presented for a transient problem with a compressible neo-Hookean material.

Keywords: Matrix assembly; Helmholtz equation; Linear elasticity; Hyperelasticity; Surrogate numerical methods; Isogeometric analysis

1. Introduction

Many techniques to accelerate the formation and assembly of coefficient matrices in Galerkin isogeometric analysis (Galerkin IGA) display their power only as the approximation order p grows. For example, in n-space dimensions, sum factorization reduces the computational complexity of element-wise matrix formation from $\mathcal{O}(p^{3n})$, realized with standard nested quadrature loops, to $\mathcal{O}(p^{2n+1})$ [1,2]. Alternatively, a weighted quadrature rule [3,4], which specifies a different quadrature rule for each individual test function, can reduce the number of quadrature points per element from $\mathcal{O}(p^n)$ to simply $\mathcal{O}(1)$. In turn, using such a rule reduces the cost of matrix formation in nested quadrature loops from $\mathcal{O}(p^{3n})$ to $\mathcal{O}(p^{2n})$. Combining both acceleration techniques can provide an even greater improvement to performance if element-wise assembly is superseded by a row/column loop. Indeed,

* Corresponding author.
E-mail addresses: drzisga@ma.tum.de (D. Drzisga), keith@ma.tum.de (B. Keith), wohlmuth@ma.tum.de (B. Wohlmuth).

https://doi.org/10.1016/j.cma.2020.113322

sum factorization and weighted quadrature, when used together with a row/column loop, has a floating point computational complexity of only $\mathcal{O}(p^{n+1})$ [5].

The surrogate matrix methodology is another way to reduce the assembly time in Galerkin methods. However, unlike the strategies mentioned above, its power comes in the small mesh size limit $h \to 0$. In fact, the methodology was first born out of applications in the classical lowest-order ($p = 1$) finite element setting [6–9]; that is, where each of the preceding approaches mentioned above have roughly the same cost. The surrogate matrix methodology is compatible with row/column loop assembly. It can also be combined with sum factorization and weighted quadrature, however, that is not a focus of this work.

The fundamental observation behind the surrogate matrix methodology is that if the basis functions used in the trial and test spaces have a specific translational symmetry, then a functional relationship can be drawn between non-zero coefficients in the matrix and points in the reference domain. This relationship is explicitly established via a finite number (specifically, $\mathcal{O}(p^n)$) of so-called *stencil functions*. If these stencil functions are smooth, they need only to be sampled at a sparse collection of points (dependent upon h) in the reference domain in order to be accurately approximated. In order to collect these sample values and, thus, define each approximate (i.e., *surrogate*) stencil function, only specific rows/columns in the final matrix need to be computed via quadrature. Thereafter, once enough samples have been collected, the remaining entries can be filled in by simply evaluating the surrogate stencil functions; an operation with a cost of $\mathcal{O}(p^n q)$, where q is the (B-spline) degree of the surrogate stencil functions.

Even though q is typically chosen larger than p, the floating point complexity remains comparable to that of other fast assembly strategies for Galerkin isogeometric methods [1–5,10–18]. More importantly, stencil functions provide a flexible platform for efficient processor-memory access which can be used to avoid cache thrashing and significantly reduce the time-to-solution in large scale, matrix-free, massively parallel computations. This has been carefully demonstrated in previous work [6–9] and is also not a focus of the present contribution.

Many mathematical aspects of the surrogate matrix methodology were worked out in the isogeometric setting in [19]. In that paper, we showed that the use of surrogate matrices introduces an additional consistency error in the discrete solution which must be controlled by the discretization error of the original method. The a priori error analysis, based on variational crimes [20], is not much different than that of reduced quadrature rules [16,21] or of the integration by interpolation and look-up strategy investigated in [13,14,18].

This paper is part of a series which can be read in any order [9,19,22]. In this contribution, we advance the mathematical development of the methodology and focus on a representative set of time-harmonic, transient, and nonlinear wave propagation problems. In particular, we present an a priori error analysis for the Helmholtz equation which shows that the additional consistency error introduced by the surrogate methodology is *independent of the wave number*. Although, we focus only on acoustic and hyperelastic waves, we expect that our conclusions will carry over to other material models as well as to other fields of application such as electro- and magnetodynamics and multi-physics wave propagation. Complementary studies with vibration and plate bending are documented in [19].

As we did in [19], we present evidence of improved performance based only on small-scale feasibility studies with the MATLAB software library GeoPDEs [23,24]. In particular, we do not consider a parallel implementation or row/column loop assembly. Although our floating point complexity analysis holds even without row/column loop assembly, both of these aspects are expected to only deliver added benefits to isogeometric surrogate matrix methods.

This paper deals with the Helmholtz equation, linearized elastic waves, and hyperelastic waves modeled with neo-Hookean materials. In Section 2, we set the stage by introducing the various equations of interest and mention certain properties of their discretization which are required for the sections which follow. In Section 3, we describe the essential features of the surrogate matrix methodology in the context of scalar solution variables and its extension to vector-valued solution variables. In Section 4, we present an a priori error analysis for the Helmholtz equation. In addition, we specify certain aspects of surrogate methods in the presence of perfectly matched layers (PMLs) and in transient and nonlinear problems. In Section 5, we briefly remark on our implementation and establish its (floating point operation) computational complexity. In Section 6, we provide computational evidence for the performance benefits of the methodology. Finally, in Section 7, we give some concluding remarks.

2. Preliminaries

In this section, we introduce the equations of interest and put forward the main notation of the paper.

2.1. General equations

Let Ω be a fixed Lipschitz domain in $\mathbb{R}^n$, where $n = 2, 3$. In addition, assume that the boundary of Ω is partitioned into two relatively open sets $\overline{\Gamma_{\mathrm{D}} \cup \Gamma_{\mathrm{N}}} = \partial\Omega$, $\Gamma_{\mathrm{D}} \cap \Gamma_{\mathrm{N}} = \emptyset$, and denote its outward unit normal by $\boldsymbol{n}$. Let W be a differentiable energy density functional, $\rho_0 \colon \Omega \to \mathbb{R}_{>0}$ be a mass density function, and $\alpha, \beta \in \mathbb{C}$, $\alpha \neq 0$, two constants. Consider the following abstract wave propagation problem on Ω, taken over the time interval $t \in [0, T]$:

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{u}_0 && \text{at } t = 0, \\ \dot{\boldsymbol{u}} &= \boldsymbol{v}_0 && \text{at } t = 0, \\ \operatorname{Div} \partial_{\boldsymbol{u}} W(\boldsymbol{u}) + \boldsymbol{f} &= \rho_0 \ddot{\boldsymbol{u}} && \text{in } \Omega \times (0, T], \\ \alpha \boldsymbol{u} + \beta \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} &= \boldsymbol{g} && \text{on } \Gamma_{\mathrm{D}} \times (0, T], \\ \partial_{\boldsymbol{u}} W(\boldsymbol{u}) \boldsymbol{n} &= \boldsymbol{t} && \text{on } \Gamma_{\mathrm{N}} \times (0, T]. \end{aligned} \tag{2.1}$$

As usual, the partial derivative in time t is denoted by $\dot{}$ and Div denotes the (row-wise) divergence operator.

Note that when W is quadratic in $\boldsymbol{u}$, we may also define the time harmonic form of (2.1) as follows:

$$\begin{aligned} -\operatorname{Div} \partial_{\boldsymbol{u}} W(\boldsymbol{u}) - k^2 \boldsymbol{u} &= \boldsymbol{f} && \text{in } \Omega, \\ \alpha \boldsymbol{u} + \beta \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} &= \boldsymbol{g} && \text{on } \Gamma_{\mathrm{D}}, \\ \partial_{\boldsymbol{u}} W(\boldsymbol{u}) \boldsymbol{n} &= \boldsymbol{t} && \text{on } \Gamma_{\mathrm{N}}. \end{aligned} \tag{2.2}$$

Here, $k \in \mathbb{R}_{\geq 0}$ is the wave number.

2.2. Examples

Our focus lies on a number of equations that can be cast in this abstract form of (2.1) and (2.2). In the case of scalar-valued solution variables, we consider the energy density functional

$$W(u) = \frac{c^2}{2} \nabla u^{\mathsf{T}} \nabla u,$$

where c is the propagation speed. Invoking this energy functional in (2.1), one retrieves the acoustic wave equation. On the other hand, the linearized elastodynamic equations for compressible homogeneous and isotropic materials are obtained by employing the energy density functional

$$W(\boldsymbol{u}) = \frac{\lambda}{2} \operatorname{tr}(\boldsymbol{\varepsilon}(\boldsymbol{u}))^2 + \mu \boldsymbol{\varepsilon}(\boldsymbol{u}) \colon \boldsymbol{\varepsilon}(\boldsymbol{u}), \tag{2.3}$$

where λ and μ are the Lamé parameters and $\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\mathsf{T}})$. Lastly, we consider the nonlinear response of a compressible neo-Hookean material by invoking the energy density functional

$$W(\boldsymbol{u}) = \frac{\lambda}{2} \ln(\det(\mathbf{F}(\boldsymbol{u})))^2 - \mu \ln(\det(\mathbf{F}(\boldsymbol{u}))) + \frac{\mu}{2} \big(\operatorname{tr}(\mathbf{F}(\boldsymbol{u})^{\mathsf{T}} \mathbf{F}(\boldsymbol{u})) - \operatorname{tr}(\mathbf{I}) \big), \tag{2.4}$$

where $\mathbf{F}(\boldsymbol{u}) = \mathbf{I} + \nabla \mathbf{u}$.

Note that the time-harmonic form of the acoustic wave equation is equivalent to the Helmholtz equation. In the next subsection, we give a short summary of several mathematical aspects of the Helmholtz equation which are used in the sequel.

2.3. Helmholtz equation

Let $\alpha = -\mathrm{i}k$, $\beta = 1$, and $\Gamma_{\mathrm{D}} = \partial\Omega$. In this setting, (2.2) results in the Helmholtz equation with impedance boundary conditions:

$$\begin{aligned} -\Delta u - k^2 u &= f && \text{in } \Omega, \\ \frac{\partial u}{\partial \boldsymbol{n}} - \mathrm{i}k u &= g && \text{on } \partial\Omega. \end{aligned} \tag{2.5}$$

For the sake of completeness, we now give a brief summary of results from [25,26].

Begin with a fixed wave number $1 \leq k_0 \leq k \leq k_1$ and define the k-dependent norm $\|u\|_{\mathcal{H}}^2 = \|\nabla u\|_{L^2(\Omega)}^2 + k^2\|u\|_{L^2(\Omega)}^2$. Next, assume that Ω is convex and that the domain mapping $\boldsymbol{\varphi}\colon \widehat{\Omega} \to \Omega$ from the reference domain $\widehat{\Omega}$ to the physical domain Ω is smooth. Let $\boldsymbol{J}(\widehat{\boldsymbol{x}})$ be the Jacobian of $\boldsymbol{\varphi}(\widehat{\boldsymbol{x}})$, $\det(\boldsymbol{J}) > 0$. We define the sesquilinear forms

$$\begin{aligned} a(u,v) &= \int_{\Omega} \nabla u \cdot \nabla \overline{v}\,\mathrm{d}\boldsymbol{x} = \int_{\widehat{\Omega}} \boldsymbol{J}^{-\mathsf{T}}\widehat{\nabla}\widehat{u} \cdot \boldsymbol{J}^{-\mathsf{T}}\widehat{\nabla}\overline{\widehat{v}}\det(\boldsymbol{J})\,\mathrm{d}\widehat{\boldsymbol{x}}, \\ m(u,v) &= \int_{\Omega} u\overline{v}\,\mathrm{d}\boldsymbol{x} = \int_{\widehat{\Omega}} \widehat{u}\overline{\widehat{v}}\det(\boldsymbol{J})\,\mathrm{d}\widehat{\boldsymbol{x}}, \\ b(u,v) &= \int_{\partial\Omega} u\overline{v}\,\mathrm{d}\boldsymbol{x} = \int_{\partial\widehat{\Omega}} \widehat{u}\overline{\widehat{v}}\,\det(\boldsymbol{J})\|\boldsymbol{J}^{-\mathsf{T}}\boldsymbol{n}\|\,\mathrm{d}\widehat{\boldsymbol{x}}, \end{aligned} \tag{2.6}$$

where $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^n$. The functions $\widehat{u}$ on the reference domain $\widehat{\Omega}$ are defined by the identity $\widehat{u} = u \circ \boldsymbol{\varphi}$.

Let $V_h \subseteq H^1(\Omega)$ be a finite-dimensional subspace with basis functions of order $p \in \mathbb{N}$ corresponding to a grid of length h. In particular, assume that $p \geq c_p \log k_1$, for a suitable constant $c_p \in \mathbb{R}_{>0}$, and $h \leq c_h \frac{\log k_1}{k_1}$, for a properly selected constant $c_h \in \mathbb{R}_{>0}$. The interested reader is referred to [26, Assumpt. 4.1] for more details on these assumptions. According to [26, Prop. 2.1], the discrete variational Helmholtz formulation,

$$\begin{cases} \text{Find } u_h \in V_h \text{ satisfying} \\ a(u_h,v) - k^2 m(u_h,v) - \mathrm{i}k b(u_h,v) = \displaystyle\int_{\Omega} f\overline{v}\,\mathrm{d}x + \int_{\partial\Omega} g\overline{v}\,\mathrm{d}x \quad \text{for all } v \in V_h\,, \end{cases} \tag{2.7}$$

has a unique solution. Let $u \in H^1(\Omega)$ be the solution of (2.7) over the space $H^1(\Omega)$. By [27, Prop. 8.1.3], $\|u\|_{\mathcal{H}} \leq C(k,\Omega)\big(\|f\|_{H^1(\Omega)'} + \|g\|_{H^{-1/2}(\partial\Omega)}\big)$, where $C(k,\Omega) > 0$ is a wave number and domain-dependent constant.

Let the symbol $\lesssim$ denote inequality by a generic positive constant, independent of k and h. According to [27, Prop. 8.1.4] and [28],

$$\|u\|_{\mathcal{H}} \lesssim \|f\|_{L^2(\Omega)} + \|g\|_{L^2(\partial\Omega)}, \tag{2.8a}$$

when Ω is convex. If, in addition, $g = 0$ and $f \in H^1(\Omega)$, then it also holds that

$$\|u\|_{\mathcal{H}} \lesssim k^{-1}\|f\|_{H^1(\Omega)}, \tag{2.8b}$$

by [26, Lemma 3.4]. According to [26, Cor. 4.6], the bounds

$$\|u - u_h\|_{\mathcal{H}} \lesssim (hk)^p\big(\|f\|_{H^{p-1}(\Omega)} + \|g\|_{H^{p-1/2}(\partial\Omega)}\big), \tag{2.9a}$$

hold for convex domains with regularity $p-1$ and $f \in H^{p-1}(\Omega)$, $g \in H^{p-1/2}(\partial\Omega)$. Furthermore, if one additionally assumes that $g = 0$, then one has the improved estimate

$$\|u - u_h\|_{\mathcal{H}} \lesssim (hk)^p k^{-1}\|f\|_{H^{p-1}(\Omega)}. \tag{2.9b}$$

Again, this follows from [26, Cor. 4.6]. Evidently, by the bounds above, uniform stability, i.e.,

$$\|u_h\|_{\mathcal{H}} \lesssim \|u\|_{\mathcal{H}}\,, \tag{2.10}$$

is obtained for all wave numbers $k > 0$. For more details, as well as numerous generalizations of the bounds above, the interested reader is referred to [26] and the references therein.

Taking $u_h = \sum_{i=1}^N \mathsf{u}_i\phi_i$, where $\{\phi_i\}_{i=1}^N$ is a basis for V_h, problem (2.7) induces the following matrix equation for the coefficient vector $\mathsf{u} = [\mathsf{u}_1, \mathsf{u}_2, \ldots, \mathsf{u}_N]^{\mathsf{T}}$:

$$\mathsf{K}\mathsf{u} - k^2\mathsf{M}\mathsf{u} - k\mathrm{i}\mathsf{B}\mathsf{u} = \mathsf{f}, \tag{2.11}$$

where $\mathsf{K}_{ij} = a(\phi_j,\phi_i)$, $\mathsf{M}_{ij} = m(\phi_j,\phi_i)$, $\mathsf{B}_{ij} = b(\phi_j,\phi_i)$, and $\mathsf{f}_i = \int_{\Omega} f\overline{\phi_i}\,\mathrm{d}x + \int_{\partial\Omega} g\overline{\phi_i}\,\mathrm{d}x$. In the next section, we replace (2.11) by a closely related approximation (i.e., surrogate).

3. Surrogate matrices: Exploiting basis structure

In this section, we illustrate the main ingredients of the surrogate matrix methodology in Galerkin IGA, using the Helmholtz equation as an example. The goal is to show how to replace (2.11) by some closely related equation

$$\widetilde{\mathsf{K}}\widetilde{\mathsf{u}} - k^2\widetilde{\mathsf{M}}\widetilde{\mathsf{u}} - k\mathrm{i}\mathsf{B}\widetilde{\mathsf{u}} = \mathsf{f}, \tag{3.1}$$

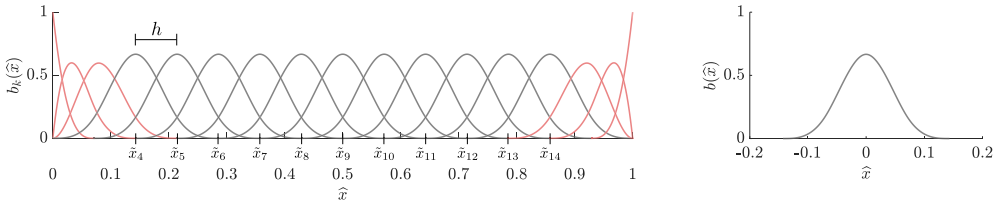


Fig. 3.1. 1D B-spline basis functions $\{b_k\}$ with cardinal B-splines in gray. These B-splines come from a third-order $p = 3$ uniform knot vector with $m = 17$. Note the points $\widetilde{x}_k$ for each $k = p+1, \dots, m-p$ and the mesh size h (left). Each gray basis function is equivalent, up to translation, to the function $b(\widehat{x})$ (right).

where $\widetilde{\mathsf{M}} \approx \mathsf{M}$ and $\widetilde{\mathsf{K}} \approx \mathsf{K}$ are faster to assemble, and the two solutions $\mathsf{u} \approx \widetilde{\mathsf{u}}$ are close to identical. Note that we choose not to replace the matrix B. Its assembly cost is of reduced complexity, since only basis functions at the boundaries need to be considered, and this B is ultimately much sparser than either K or M. The first ingredient of the surrogate matrix construction, is the concept of cardinal B-splines.

3.1. Cardinal B-splines

Every univariate B-spline basis $\{b_k\}_{k=1}^m$ is defined by an ordered multiset, or *knot vector*, $\Xi = \{\xi_1, \dots, \xi_{m+p+1}\}$ [29]. From now on, we assume that every such Ξ is an *open uniform knot vector* on the unit interval $[0, 1]$. That is, $\xi_1, \dots, \xi_{p+1} = 0$, $\xi_{m+1}, \dots, \xi_{m+p+1} = 1$, and $\xi_{k+1} - \xi_k = \frac{1}{m-p}$, otherwise. For large enough m, such knot vectors deliver a vast majority of translation invariant basis functions, such as those depicted in gray in Fig. 3.1. These functions are called *cardinal B-splines* [30–32]. We hereby refer to $h = \max_{1 \le k \le m-1} |\xi_{k+1} - \xi_k| = \frac{1}{m-p}$ as the *mesh size* parameter and define $\widetilde{x}_k = (k - \frac{p+1}{2}) \cdot h$, for each $k = p+1, \dots, m-p$. See Fig. 3.1 for an illustration of the points $\widetilde{x}_k$ and the mesh size h.

The open uniform knot vectors described above generate $m - 2p$ univariate cardinal B-spline basis functions which can each be expressed as $b_k(\widehat{x}) = b(\widehat{x} - \widetilde{x}_k)$, where $b(\widehat{x})$ is a function centered at the origin, as depicted on the right of Fig. 3.1. Just as in [19], we do not consider NURBS spaces with different polynomial orders $p_1, \dots, p_n$ in each Cartesian direction. Therefore, the tensor product definition of the multivariate B-spline basis, $\{\widehat{B}_{\boldsymbol{i}}(\widehat{\boldsymbol{x}})\}$, immediately delivers $(m-2p)^n$ *multivariate* cardinal B-splines, $\widehat{B}_{\boldsymbol{i}}(\widehat{\boldsymbol{x}}) = \widehat{B}(\widehat{\boldsymbol{x}} - \widetilde{\boldsymbol{x}}_{\boldsymbol{i}})$, where $\widetilde{\boldsymbol{x}}_{\boldsymbol{i}} = \left(\widetilde{x}_{i_1}, \dots, \widetilde{x}_{i_n}\right)$ and $\widehat{B}(\boldsymbol{x}) = b(x_1) \cdots b(x_n)$.

Here and from now on, we identify every global index $i \in \mathcal{I} = \{1, \dots, N = m^n\}$ with a multi-index $\boldsymbol{i} = (i_1, \dots, i_n)$, $1 \le i_k \le m$, through the colexicographical relationship $i = i_1 + (i_2 - 1)m + \cdots + (i_n - 1)m^{n-1}$. For future reference, we denote the set of all such $\widetilde{\boldsymbol{x}}_{\boldsymbol{i}} = \widetilde{\boldsymbol{x}}_i$ by $\widetilde{\mathbb{X}}$. Notice that the ratio of cardinal B-spline basis functions to total B-spline basis functions, $\left(\frac{m-2p}{m}\right)^n$, quickly tends to unity as m increases.

3.2. Stencil functions

In Galerkin methods, stencil functions provide an explicit functional relationship between entries in the global coefficient matrices, so long as the underlying basis has a particular structure. Here, we recall a simple definition of stencil functions which comes about by exploiting the structure of cardinal B-splines. For a generalization to NURBS bases made out of cardinal B-splines, see [19]. Meanwhile, for a description of stencil functions derived from simplicial basis functions, see [9].

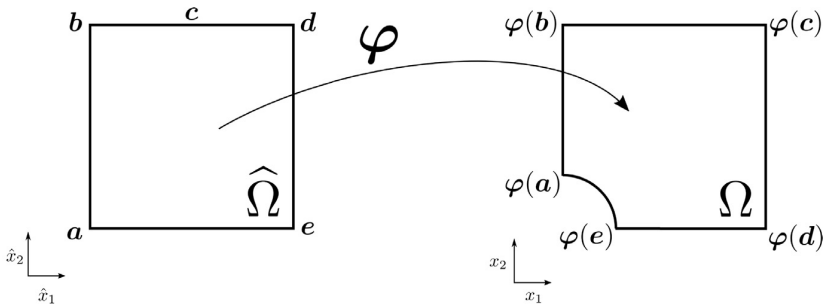
Begin by recalling the sesquilinear forms $m(\cdot, \cdot)$ and $a(\cdot, \cdot)$ defined in (2.6) and the notation from Section 3.1. For $\widehat{B} \colon \mathbb{R}^n \to \mathbb{R}$, let us consider the following scalar-valued functions:

$$\mathcal{M}(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}}) = m(\widehat{B}(\cdot - \widetilde{\boldsymbol{y}}), \widehat{B}(\cdot - \widetilde{\boldsymbol{x}})), \qquad \mathcal{K}(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}}) = a(\widehat{B}(\cdot - \widetilde{\boldsymbol{y}}), \widehat{B}(\cdot - \widetilde{\boldsymbol{x}})).$$

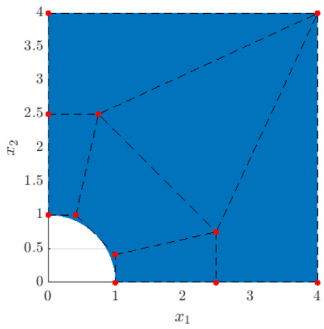
It may be readily observed that

$$\mathcal{M}(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j) = [\mathsf{M}]_{ij} \qquad \text{and} \qquad \mathcal{K}(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j) = [\mathsf{K}]_{ij}$$

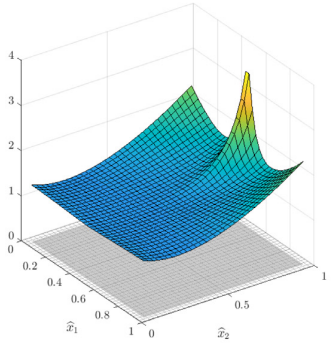
for every $i, j \in \mathcal{I}$.



(a) Single geometry map $\boldsymbol{\varphi}$ that takes the reference domain $\widehat{\Omega}$ to the physical domain Ω.



(b) Single patch control net of the physical domain Ω.



(c) Stencil function of the stiffness matrix for $\delta = (0,0)^{\mathsf{T}}$ plotted over the reference domain.

Fig. 3.2. Geometry map, control net, and stencil function in the case of a single patch geometry.

Notice that the mass matrix M and the stiffness matrix K are always sparse simply because $m(\widehat{B}_j, \widehat{B}_i)$ and $a(\widehat{B}_j, \widehat{B}_i)$ both vanish whenever the supports of $\widehat{B}_j$ and $\widehat{B}_i$ do not overlap. For the same reason, both $\mathcal{M}(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})$ and $\mathcal{K}(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})$ return zero whenever $\|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{x}}\| \geq 0$ is large enough.

In order to demarcate from these trivial outcomes, we rewrite $\mathcal{M}(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})$ and $\mathcal{K}(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})$ in terms of $\widetilde{\boldsymbol{x}}$ and a translation $\boldsymbol{\delta} = \widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{x}}$ by defining

$$\mathcal{M}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}) = \mathcal{M}(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{x}} + \boldsymbol{\delta}) \qquad \text{and} \qquad \mathcal{K}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}) = \mathcal{K}(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{x}} + \boldsymbol{\delta}).$$

Taking $\boldsymbol{\delta} = \widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{x}}_i$, we clearly have

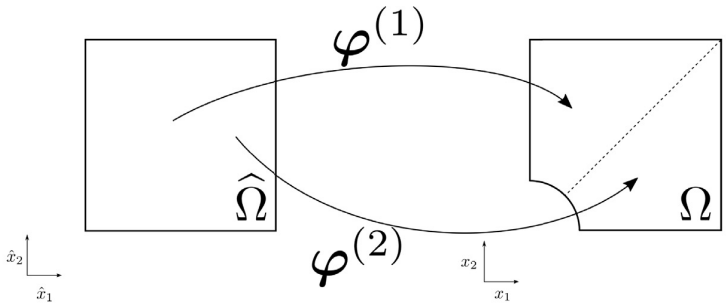
$$[\mathsf{M}]_{ij} = \mathcal{M}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i) \qquad \text{and} \qquad [\mathsf{K}]_{ij} = \mathcal{K}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i). \tag{3.2}$$

For this reason, we only need to pay attention to $\boldsymbol{\delta} \in \mathcal{D}$, where

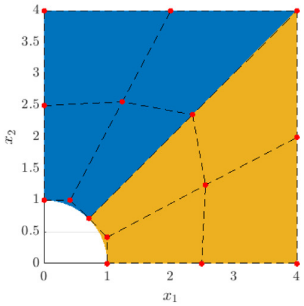
$$\mathcal{D} = \{\widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{x}}_i : \operatorname{supp}(\widehat{B}(\cdot - \widetilde{\boldsymbol{x}}_i)) \cap \operatorname{supp}(\widehat{B}(\cdot - \widetilde{\boldsymbol{x}}_j)) \neq \emptyset,\ i, j \in \mathcal{I}\}.$$

Using the fact that each point in $\mathcal{I}$ is uniformly spaced through the reference domain, one can easily show that $\#\mathcal{D} = (2p+1)^n$. Thus, $\mathcal{D}$ can be seen as a finite index set. We call each function $\mathcal{M}_{\boldsymbol{\delta}}$ and $\mathcal{K}_{\boldsymbol{\delta}}$, enumerated by $\boldsymbol{\delta} \in \mathcal{D}$, a *stencil function*. The interested reader is referred to [9,19] as well as Figs. 3.2 and 3.3 for various pictures of stencil functions.

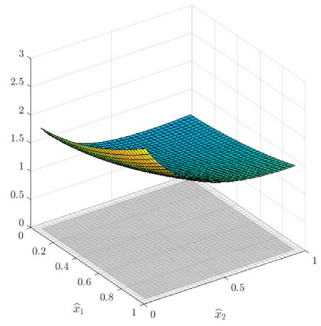
Remark 3.1. In a multi-patch setting, the physical domain Ω is partitioned into a finite number of disjoint subdomains $\overline{\Omega} = \bigcup_{\ell=1}^{L} \overline{\Omega^{(\ell)}}$. Each *patch* $\Omega^{(\ell)}$ is identified with the same parametric domain $\widehat{\Omega}$ via a unique isogeometric transformation $\boldsymbol{\varphi}^{(\ell)}(\widehat{\Omega}) = \Omega^{(\ell)}$. For this reason, extending the definition of the stencil functions to account for multi-patch geometries is straightforward. Indeed, one only needs to define a separate set of stencil functions $\mathcal{M}_{\boldsymbol{\delta}}^{(\ell)}$ and $\mathcal{K}_{\boldsymbol{\delta}}^{(\ell)}$, for each patch index ℓ.



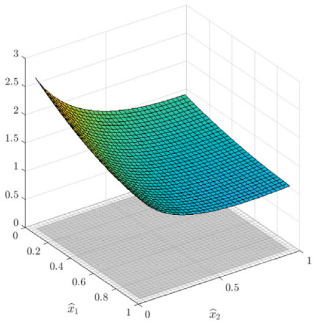
(a) Two geometry maps $\varphi^{(1)}$ and $\varphi^{(2)}$ that map the reference domain $\widehat{\Omega}$ to the physical domain Ω.



(b) Two patch control net of the physical domain Ω.



(c) Stencil function of the stiffness matrix for $\delta = (0,0)^{\mathsf{T}}$ and the first patch.



(d) Stencil function of the stiffness matrix for $\delta = (0,0)^{\mathsf{T}}$ and the second patch.

Fig. 3.3. Geometry map, control net, and stencil function in the case of a two patch geometry.

3.3. Surrogate stencil functions

The equations in (3.2) are simply functional relationships between the entries of each submatrix M and K and the arguments of $\mathcal{M}_{\boldsymbol{\delta}}$ and $\mathcal{K}_{\boldsymbol{\delta}}$, respectively. In other words, evaluating $\mathcal{M}_{\boldsymbol{\delta}}$ at any point $\widetilde{\boldsymbol{x}}_i \in \widetilde{\mathbb{X}}$ is operationally equivalent to computing the matrix entry $[\mathsf{M}]_{ij}$. Therefore, evaluating $\mathcal{M}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i)$, for each $\boldsymbol{\delta} \in \mathcal{D}$, requires computing precisely all the non-zero coefficients in the ith row of M. The same observation clearly also holds when evaluating $\mathcal{K}_{\boldsymbol{\delta}}$.

If stencil functions are smooth, then they may be accurately approximated by their values at a relatively small number of points $\widetilde{\boldsymbol{x}}_{i^{\mathrm{s}}} \in \widehat{\Omega}$. For our purposes, it is enough to let $\widetilde{\mathbb{X}}^{\mathrm{s}} \subseteq \widetilde{\mathbb{X}}$ be the set of all such *sample points* $\widetilde{\boldsymbol{x}}_{i^{\mathrm{s}}}$ and let $\mathcal{I}^{\mathrm{s}} \subseteq \mathcal{I}$ be the corresponding set of indices. This procedure first requires collecting all pairs $(\widetilde{\boldsymbol{x}}_{i^{\mathrm{s}}}, [\mathsf{M}]_{i^{\mathrm{s}}j})$, for every $\widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{x}}_{i^{\mathrm{s}}} \in \mathcal{D}$ and $i^{\mathrm{s}} \in \mathcal{I}^{\mathrm{s}}$, but may be done simply by modifying existing assembly algorithms to compute only the required rows. Entries in surrogate matrices may then be generated by just evaluating the approximated stencil functions at the remaining points in $\widetilde{\mathbb{X}} \setminus \widetilde{\mathbb{X}}^{\mathrm{s}}$ and filling in the corresponding rows $\mathcal{I} \setminus \mathcal{I}^{\mathrm{s}}$.

Define

$$[\widetilde{\mathsf{M}}]_{ij} = \widetilde{\mathcal{M}}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i) \qquad \text{and} \qquad [\widetilde{\mathsf{K}}]_{ij} = \widetilde{\mathcal{K}}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i)\,, \tag{3.3}$$

where $\widetilde{\mathcal{M}}_{\boldsymbol{\delta}}$ and $\widetilde{\mathcal{K}}_{\boldsymbol{\delta}}$ are such approximations of $\mathcal{M}_{\boldsymbol{\delta}}$ and $\mathcal{K}_{\boldsymbol{\delta}}$, respectively. If these so-called *surrogate stencil functions*, $\widetilde{\mathcal{M}}_{\boldsymbol{\delta}}$ and $\widetilde{\mathcal{K}}_{\boldsymbol{\delta}}$, are expressed in an easily evaluated basis, then $\widetilde{\mathsf{M}}$ (resp. $\widetilde{\mathsf{K}}$) can be formed much faster than M (resp. K), simply because of the numerical integration that is avoided. Moreover, for large enough problems, the coefficients in the surrogate stencil functions should require significantly less storage than the coefficients in the original matrix they are used to approximate. This makes simply storing the stencil function coefficients and reading out evaluations of $\widetilde{\mathcal{M}}_{\boldsymbol{\delta}}$ or $\widetilde{\mathcal{K}}_{\boldsymbol{\delta}}$ very desirable during each matrix–vector multiply in matrix-free methods, especially when the matrices themselves cannot fit in main memory; see, e.g., [9].

In this paper, we construct surrogate stencil functions by interpolating $\mathcal{M}_\delta$ and $\mathcal{K}_\delta$ with a uniform B-spline basis of order $q \geq 0$ with a quasi-uniform knot vector $\widetilde{\Xi} = \widetilde{\Xi}_1 \times \cdots \times \widetilde{\Xi}_n$, where each knot $\widetilde{\boldsymbol{\xi}}_{\boldsymbol{i}} \in \widetilde{\Xi}$ is taken from $\widetilde{\mathbb{X}}^{\mathrm{s}}$. Just as the accuracy of the discrete solution u_h is affected by the mesh size parameter h, the accuracy of surrogate stencil functions is affected by a sampling length

$$H = \max_{|\boldsymbol{j}|=1,\boldsymbol{i}} \left\{ \|\widetilde{\boldsymbol{\xi}}_{\boldsymbol{i}+\boldsymbol{j}} - \widetilde{\boldsymbol{\xi}}_{\boldsymbol{i}}\|_\infty \,:\, \widetilde{\boldsymbol{\xi}}_{\boldsymbol{i}}, \widetilde{\boldsymbol{\xi}}_{\boldsymbol{i}+\boldsymbol{j}} \in \widetilde{\Xi} \right\}. \tag{3.4}$$

As a simplifying accommodation, we assume that all stencil functions $\mathcal{M}_\delta$ and $\mathcal{K}_\delta$ are defined at every sampling point $\widetilde{\boldsymbol{x}}_{i^{\mathrm{s}}} \in \widetilde{\mathbb{X}}^{\mathrm{s}}$. As argued in [19, Section 4], this implies that $\widetilde{\mathbb{X}}^{\mathrm{s}} \subseteq \widetilde{\Omega} \subsetneq \widehat{\Omega}$, where

$$\widetilde{\Omega} = \left[\frac{3p+1}{2(m-p)}, 1 - \frac{3p+1}{2(m-p)} \right]^n.$$

For a more complete description of the interpolation strategy used in the coming experiments, as well as the resulting analysis, see Section 5.1 and [22]. Note that explicit interpolation is not at all required to generate an accurate surrogate. Indeed, a different least-squares regression approach, with a high-order polynomial basis, was successfully applied in [9]. Many approximation alternatives remain to be investigated.

3.4. Structure-preserving surrogates

Constructing the complete surrogate matrices $\widetilde{\mathsf{M}}$ and $\widetilde{\mathsf{K}}$ out of the corresponding stencil functions requires the consideration of interactions with non-cardinal basis functions. The simplest way to account for the entries of $\widetilde{\mathsf{M}}$ and $\widetilde{\mathsf{K}}$ which are not defined via (3.3) is to compute them directly with numerical quadrature, as in traditional IGA assembly algorithms. This is the choice we make here, however, alternative choices are available by using additional stencil functions which exploit symmetries on lower-dimensional planes, as described in [19, Section 3.6].

Exploiting the symmetry of the mass matrix, we define

$$[\widetilde{\mathsf{M}}]_{ij} = \begin{cases} \widetilde{\mathcal{M}}_\delta(\widetilde{\boldsymbol{x}}_i) & \text{if } \widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j \in \widetilde{\Omega},\ i \leq j, \\ [\widetilde{\mathsf{M}}]_{ji} & \text{if } \widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j \in \widetilde{\Omega},\ i > j, \\ [\mathsf{M}]_{ij} & \text{otherwise.} \end{cases} \tag{3.5}$$

Note that this definition requires interpolating $((2p+1)^n+1)/2$ stencil functions. Constructing the surrogate stiffness matrix $\widetilde{\mathsf{K}}$ could follow in the same manner as (3.5), however, a surrogate matrix with better approximation properties can be found if we attempt to preserve part of the kernel of the original matrix K.

Note that the kernel of K contains all repeated coefficient vectors, $\mathrm{span}\{[1, 1, \ldots, 1]^{\mathsf{T}}\}$. Indeed, because $a(1, w) = a(w, 1) = 0$ for all $w \in H^1(\Omega)$ and because B-splines and NURBS have the partition of unity property $\sum_j B_j(\boldsymbol{x}) = 1$, it holds that

$$0 = a(1, \phi_i) = \sum_j a(B_j, \phi_i) = \sum_j [\mathsf{K}]_{ij}\,, \quad \text{for each } i = 1, \ldots, N.$$

Note that this identity may be rewritten

$$[\mathsf{K}]_{ii} = -\sum_{j \neq i} [\mathsf{K}]_{ij}\,.$$

For this reason, we pose the following symmetric kernel-preserving definition for the surrogate stiffness matrix:

$$[\widetilde{\mathsf{K}}]_{ij} = \begin{cases} \widetilde{\mathcal{K}}_\delta(\widetilde{\boldsymbol{x}}_i) & \text{if } \widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j \in \widetilde{\Omega},\ i < j, \\ [\widetilde{\mathsf{K}}]_{ji} & \text{if } \widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j \in \widetilde{\Omega},\ i > j, \\ [\mathsf{K}]_{ij} & \text{in all other cases where } i \neq j, \\ -\sum_{k \neq i} [\widetilde{\mathsf{K}}]_{ik} & \text{if } i = j. \end{cases} \tag{3.6}$$

Note that definition (3.6) requires interpolating $((2p+1)^n - 1)/2$ stencil functions. We define the corresponding surrogate sesquilinear forms $\widetilde{a}(u, v) = \bar{\mathsf{v}}^{\mathsf{T}} \widetilde{\mathsf{K}} \mathsf{u}$ and $\widetilde{m}(u, v) = \bar{\mathsf{v}}^{\mathsf{T}} \widetilde{\mathsf{M}} \mathsf{u}$ where u and v are the coefficient vectors of u and v in the $\{\phi_i\}_{i=1}^N$ basis, respectively.

Remark 3.2. Definitions (3.5) and (3.6) also apply in the obvious way to the multi-patch setting. Indeed, they can be simply used to define every patch-wise coefficient matrix $\widetilde{\mathsf{M}}^{(\ell)}$ and $\widetilde{\mathsf{K}}^{(\ell)}$ using the corresponding patch-wise stencil functions $\mathcal{M}_{\delta}^{(\ell)}$ and $\mathcal{K}_{\delta}^{(\ell)}$, respectively.

Remark 3.3. The definition of the surrogate mass matrix $\widetilde{\mathsf{M}}$ in (3.5) does not preserve the exact volume of the domain in the sense that $\sum_i \sum_j [\widetilde{\mathsf{M}}]_{ij} \neq \int_\Omega 1 \, \mathrm{d}\boldsymbol{x}$; cf. [19, Remark 5.1]. However, the volume may still be preserved by changing its construction in the following way. Let D be a diagonal matrix with $[\mathsf{D}]_{ii} = \int_\Omega B_i(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$ for each i. The true stiffness matrix can be split into $\mathsf{M} = \mathsf{D} + \mathsf{M}_0$ where $[\mathsf{M}_0]_{ij} = [\mathsf{M}]_{ij}$ for all $j \neq i$ and $[\mathsf{M}_0]_{ii} = -\sum_{j \neq i} [\mathsf{M}]_{ij}$ for all i. Since M_0 has the same structure and zero row-sum property as K, the surrogate matrix $\widetilde{\mathsf{M}}_0$ may be defined as in (3.6). Therefore, defining the mass matrix surrogate as $\widetilde{\mathsf{M}} = \mathsf{D} + \widetilde{\mathsf{M}}_0$ yields the desired property $\sum_i \sum_j [\widetilde{\mathsf{M}}]_{ij} = \sum_i \sum_j [\mathsf{M}]_{ij} = \int_\Omega 1 \, \mathrm{d}\boldsymbol{x}$. This definition only requires the additional assembly of the diagonal matrix D which can be stored in a vector. The required quadrature formula may also be of lower accuracy, because functions of order p and not $2p$ need to be integrated. This observation is not further investigated here.

Remark 3.4. The majority of the definitions above generalize immediately to variational problems with vector-valued solutions. Nevertheless, in the case of linear elasticity, preserving all of the infinitesimal rigid body motions in the definition of a surrogate elasticity stiffness matrix, is more complicated and expensive than preserving the one-dimensional kernel of K, as done in (3.6). Our numerical experiments do not show any significant need to incorporate such a feature.

3.5. Smooth geometry transformations

In many studies (see, e.g., [5,29]) the geometry transformation $\boldsymbol{\varphi} : \widehat{\Omega} \to \Omega$ is not globally smooth. For illustration, consider the singular transformation depicted in Fig. 3.2(a) which has a singularity coming from the lack of smoothness at the top right corner of the physical domain. This singularity in the geometry transformation implies a singularity in the determinant of the Jacobian $\boldsymbol{J}$ present in (2.6). In turn, a singularity also appears in the corresponding stencil functions; see, e.g., Fig. 3.2(c).

Singular geometry transformations will usually introduce singular features in the stencil functions. As singular functions are more difficult to approximate accurately, using unnecessary singular geometry maps should be avoided with surrogate matrix methods. For instance, in the example above, the singularity may be removed simply by using two patches instead of just one. In Fig. 3.3(b), an obvious two-patch geometry parameterization is used and it is obvious to infer that the resulting stencil functions will be globally smooth; Fig. 3.3(b) shows one such representative.

4. Surrogate matrices: Theory and applications

In this section, we present an a priori error estimate for the Helmholtz case which shows that the consistency error introduced by the surrogate methodology is *wave number independent*. Next, we explain how the surrogate methodology is advantageous in wave propagation problems with absorbing boundary conditions. Finally, we give a short survey of other insights and interpretations which apply for time-dependent and nonlinear problems.

4.1. A priori error estimates for the Helmholtz equation

The following theorem certifies optimal order convergence of the discretization (3.1), under certain assumptions on $\widetilde{a}(u, v) = \bar{\mathsf{v}}^{\mathsf{T}} \widetilde{\mathsf{K}} \mathsf{u}$ and $\widetilde{m}(u, v) = \bar{\mathsf{v}}^{\mathsf{T}} \widetilde{\mathsf{M}} \mathsf{u}$. Justification for the stability assumptions made in (4.1a) and (4.1b) comes from previous work (e.g., [19]); for further details, see Remark 4.4.

Theorem 4.1. *Invoke all the hypotheses of Section 2.3 and define H via (3.4). Moreover, let $q_1, q_2 \in \mathbb{N}_0$ and assume that*

$$|a(u, v) - \widetilde{a}(u, v)| \lesssim H^{q_1+1} \|\nabla u\|_{L^2(\Omega)} \|\nabla v\|_{L^2(\Omega)}, \tag{4.1a}$$

$$|m(u, v) - \widetilde{m}(u, v)| \lesssim H^{q_2+1} \|u\|_{L^2(\Omega)} \|v\|_{L^2(\Omega)}, \tag{4.1b}$$

for all $u, v \in V_h$. Then, for all sufficiently small H, we have the existence of a unique solution $\widetilde{\mathfrak{u}}$ of (3.1) and the following a priori error estimate for $\widetilde{u}_h = \sum_i \widetilde{\mathfrak{u}}_i \phi_i$:

$$\|u - \widetilde{u}_h\|_{\mathcal{H}} \lesssim (hk)^p \big(\|f\|_{H^{p-1}(\Omega)} + \|g\|_{H^{p-1/2}(\partial\Omega)} \big) + H^{q+1} \big(\|f\|_{L^2(\Omega)} + \|g\|_{L^2(\partial\Omega)} \big), \tag{4.2a}$$

where $q = \min\{q_1, q_2\}$. If, in addition, $g = 0$, then we have the alternative estimate

$$\|u - \widetilde{u}_h\|_{\mathcal{H}} \lesssim k^{-1} \big((hk)^p \|f\|_{H^{p-1}(\Omega)} + H^{q+1} \|f\|_{H^1(\Omega)} \big). \tag{4.2b}$$

Proof. Fix $k > 0$. Define the sesquilinear form

$$\mathcal{A}(u, v) = a(u, v) - k^2 m(u, v) - kib(u, v) \quad \text{for all } u, v \in H^1(\Omega),$$

and denote its discrete stability constant by γ_h. Likewise, define the surrogate sesquilinear form

$$\widetilde{\mathcal{A}}(u, v) = \widetilde{a}(u, v) - k^2 \widetilde{m}(u, v) - kib(u, v) \quad \text{for all } u, v \in V_h.$$

Observe that

$$\widetilde{\mathcal{A}}(\widetilde{u}_h, v) = \int_\Omega f \overline{v} \, \mathrm{d}x + \int_{\partial\Omega} g \overline{v} \, \mathrm{d}x = \mathcal{A}(u_h, v) \quad \text{for all } v \in V_h. \tag{4.3}$$

The assumptions on h and p from Section 2.3 imply that

$$\gamma_h = \inf_{u \in V_h \setminus \{0\}} \sup_{v \in V_h \setminus \{0\}} \frac{|\mathcal{A}(u, v)|}{\|u\|_{\mathcal{H}} \|v\|_{\mathcal{H}}} > 0.$$

This guarantees uniform stability of the original isogeometric discretization for all sufficiently small h. Our first aim is to demonstrate that a similar property holds for the surrogate discretization given by (3.1). Indeed, observe that for any arbitrary $u \in V_h$,

$$\begin{aligned} \sup_{v \in V_h \setminus \{0\}} \frac{|\widetilde{\mathcal{A}}(u, v)|}{\|v\|_{\mathcal{H}}} &\geq \sup_{v \in V_h \setminus \{0\}} \frac{|\mathcal{A}(u, v)|}{\|v\|_{\mathcal{H}}} - \sup_{v \in V_h \setminus \{0\}} \frac{|\widetilde{\mathcal{A}}(u, v) - \mathcal{A}(u, v)|}{\|v\|_{\mathcal{H}}} \\ &\geq \big(\gamma_h - \max\{C_1 H^{q_1+1}, C_2 H^{q_2+1}\} \big) \|u\|_{\mathcal{H}}. \end{aligned}$$

Therefore,

$$\widetilde{\gamma}_h = \inf_{u \in V_h \setminus \{0\}} \sup_{v \in V_h \setminus \{0\}} \frac{|\mathcal{A}(u, v)|}{\|u\|_{\mathcal{H}} \|v\|_{\mathcal{H}}} > \gamma_h - \max\{C_1 H^{q_1+1}, C_2 H^{q_2+1}\} > 0,$$

for all sufficiently small H.

Assuming sufficiently small h and H, it now follows that u_h and $\widetilde{u}_h$ both exist and are unique. By the triangle inequality, $\|u - \widetilde{u}_h\|_{\mathcal{H}} \leq \|u - u_h\|_{\mathcal{H}} + \|u_h - \widetilde{u}_h\|_{\mathcal{H}}$. Invoking (4.3) and then (4.1), the consistency error term, $\|u_h - \widetilde{u}_h\|_{\mathcal{H}}$, may be bounded from above as follows:

$$\widetilde{\gamma}_h \|u_h - \widetilde{u}_h\|_{\mathcal{H}} \leq \sup_{v \in V_h \setminus \{0\}} \frac{|\widetilde{\mathcal{A}}(u_h - \widetilde{u}_h, v)|}{\|v\|_{\mathcal{H}}} = \sup_{v \in V_h \setminus \{0\}} \frac{|\widetilde{\mathcal{A}}(u_h, v) - \mathcal{A}(u_h, v)|}{\|v\|_{\mathcal{H}}} \lesssim H^{q+1} \|u_h\|_{\mathcal{H}}.$$

Inequality (4.2a) now follows from (2.10), (2.8a) and (2.9a). Likewise, if $g = 0$, inequality (4.2b) follows from (2.10), (2.8b) and (2.9b). □

Remark 4.1. In (4.2a), it is important to note that the consistency error $\|u_h - \widetilde{u}_h\|_{\mathcal{H}}$, stemming from the surrogate matrices, is independent of the wave number. Meanwhile, in the same setting, the upper bound on the discretization error $\|u - u_h\|_{\mathcal{H}}$ scales like k^p. This makes the surrogate methodology very attractive for large wave number problems, since the total error $\|u - \widetilde{u}_h\|_{\mathcal{H}}$ will tend to be dominated by the discretization error $\|u - u_h\|_{\mathcal{H}}$.

Remark 4.2. In the special case $g = 0$, considered by (4.2b), it is well known that the discretization error improves by a factor of k^{-1}. What is perhaps surprising in the analysis above is that the consistency error of the surrogate method will also improve by the same factor, at least provided that $f \in H^1(\Omega)$. This conclusion follows immediately from the improved stability estimate (2.8b). Thus, in both the $g \neq 0$ and $g = 0$ settings, the ratio between the discretization error and consistency error remains $\mathcal{O}(k^p)$.

Remark 4.3. According to results from [25,26], the error bounds may carry an additional factor of $k^{\frac{5}{2}}$ if Ω is not convex.

Remark 4.4. Theorem 7.2 in [19] shows that (4.1a) holds for the surrogate stiffness matrix $\widetilde{\mathsf{K}}$ defined by (3.6). Likewise, assumption (4.1b) can be shown to hold for the surrogate mass matrix $\widetilde{\mathsf{M}}$ defined in (3.5); cf. [19, Section 8.1].

4.2. Perfectly matched layer boundary conditions

Open wave problems posed on unbounded domains are commonly solved on truncated computational domains. In order to solve such problems accurately, spurious reflections of the outgoing waves, caused by the truncated domain, need to be absorbed. One approach to simulate this behavior is the perfectly matched layer (PML) absorbing boundary condition introduced in [33]. With this approach, the domain of interest is extended by an artificial absorbing layer made from a special medium. Many alternative strategies for general curvilinear domains have been proposed since then, but the underlying idea stays the same.

One possibility, which we choose to follow, is stretching the real domain into a complex domain. This stretching is achieved by replacing the physical domain map $\boldsymbol{\varphi}(\widehat{\mathbf{x}})$ by an artificial map

$$\widetilde{\boldsymbol{\varphi}}(\widehat{\mathbf{x}}) = \boldsymbol{\varphi}(\widehat{\mathbf{x}}) + \mathrm{i}C\mathbf{f}(\widehat{\mathbf{x}}),$$

where $\mathbf{f}$ is zero on the domain of interest and is smoothly increasing to unity on the layer's boundary. The constant $C > 0$ is a problem dependent penalty term controlling the strength of the absorption of the layer. Details on the integral transformations introduced by this complex stretching may be found in [34].

The surrogate matrix methodology is very suitable for simulations with PMLs because the discretization error $\|u - u_h\|_{\mathcal{H}}$ is usually bounded from below by a positive constant depending on the size and shape of the absorbing layer. On the other hand, the consistency error $\|u_h - \widetilde{u}_h\|_{\mathcal{H}}$ only measures the distance between the two approximate solutions and, therefore, still tends to zero as the mesh is refined. Our experience has indicated that the difference between the standard IGA solution and the surrogate IGA solution is rarely distinguishable, even at low wave numbers. Moreover, as we demonstrate in Section 6, the consistency error, although generally small, tends to be largest in the absorbing layer. Because only the non-absorbing part of the domain is of interest, these errors in the absorbing layer are of no interest. We consider PML boundary conditions for a linear elastodynamics problem with periodic pressure loading in Section 6.3.

4.3. Discretization in time

Explicit and implicit time discretization schemes require matrices to propagate solutions forward in time. Implicit schemes additionally require solving one or more linear systems at each time step. The surrogate matrix methodology can also be used in such cases for assembling these propagation matrices. However, if the problem is linear and the iteration matrices do not change over time, and unless a matrix-free approach is considered, each matrix only needs to be assembled once. In this case, the achievable speed-up depends on the number of time steps. Indeed, the total relative performance improvement will diminish as the number of time steps grows.

The upshot changes for nonlinear problems where the performance of the surrogate methodology is independent of the number of time steps. Indeed, the propagation matrices need to be reassembled throughout the simulation because they depend both on the solution at previous time steps and on the iterates of the current time step. We showcase a time-dependent nonlinear hyperelastic wave problem in Section 6.4 and use it to compare performance.

4.4. Nonlinear problems using Newton's method

It has already been demonstrated in [9] that the surrogate matrix methodology is suitable for nonlinear problems. However, in that work, we only considered Picard fixed point iterations. Although our results were promising, we found that many iterations were required to arrive at the desired solver tolerance. In this work, we chose to focus on solving nonlinear problems with Newton's method where the Jacobian matrix needs to be reassembled in each iteration. Now, because the surrogate matrix methodology only yields approximations of matrices, the surrogate

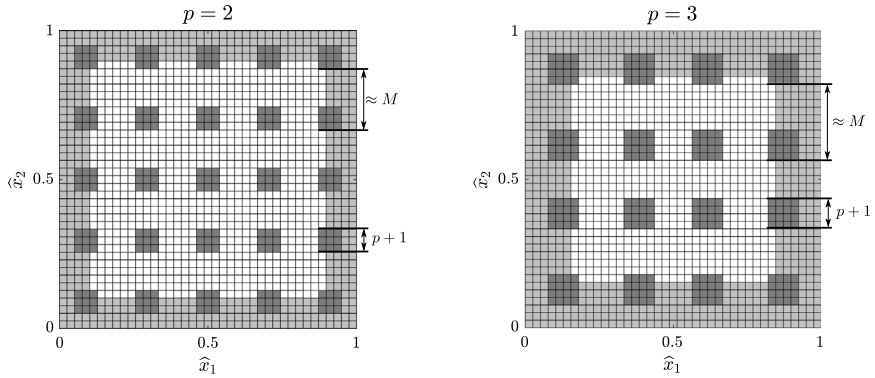


Fig. 5.1. The active elements (shown in gray) involved in the surrogate assembly for $M = 10$ with forty knots in each Cartesian direction. The light gray elements correspond to the active boundary elements and the dark gray elements correspond to the inner active elements required for the sampling of the stencil functions.

Jacobian matrix is simply just an approximation of the true Jacobian matrix. This means that a Newton method combined with a surrogate method may be more easily interpreted as just a sophisticated quasi-Newton method for the original problem. One particular consequence is that the consistency error $\|u_h - \widetilde{u}_h\|_{\mathcal{H}}$ will vanish with the number of Newton iterations. Note that in many nonlinear problems, optimizations such as exploiting the symmetry in (3.5) or the row-sum property in (3.6) cannot be used.

5. Surrogate matrices: Algorithmic considerations

In this section, we give a short comment on the differences of the implementation used in this paper when compared to implementations of our previous work in [19,22]. We conclude this section with a computational complexity estimate for the asymptotic number of floating point operations (FLOPs) required for the surrogate matrix methodology.

5.1. Implementation

As in [19], all of the experiments documented in this paper were implemented using the GeoPDEs package for Isogeometric Analysis in MATLAB and Octave [23,24]. Our implementation reused most of the original functionality in GeoPDEs. A detailed explanation of the modifications and extensions is given in [22], albeit only for the Poisson equation. Apart from the software implementation aspects, which are more or less unchanged from our previous work, in this paper we utilized a slightly different strategy for selecting the sample points $\widetilde{\boldsymbol{x}}_A^{\mathfrak{s}}$ and we used a different B-spline interpolation function.

Let $M > 0$ be a fixed integer. Roughly speaking, when constructing the multivariate B-spline functions, $\widetilde{\mathcal{M}}_\delta$ and $\widetilde{\mathcal{K}}_\delta$, our goal is to interpolate only about $1/M$ of the points in $\widetilde{\mathbb{X}}$, in each Cartesian direction. In [19], this was done by simply taking every Mth point in $\widetilde{\mathbb{X}}$, in each direction, and adding in every Mth boundary point, if it was skipped over. In this work, in order to better distribute the sample points, we first find the total number of points L in one Cartesian direction in $\widetilde{\mathbb{X}}$, and then sample every $(L-1)/\operatorname{ceil}\{\frac{L-1}{M}\}$ point, after rounding to the nearest integer. By starting at a given corner, this strategy makes sure boundary points are sampled and that all points are roughly evenly spaced; cf. Fig. 5.1. Of course, other sampling point distributions, as for example Chebyshev nodes, may also be used with this approach. Moreover, in this paper, we used the function `spapi`, provided by the MATLAB curve fitting toolbox, instead of the standard MATLAB functions `interp2` and `interp3` or the SciPy Python function `RectBivariateSpline`. `spapi` allows for more general higher-order B-spline interpolations although it is slightly slower than the other functions.

Note that our method for evaluating $\widetilde{\mathcal{M}}_\delta(\widetilde{\boldsymbol{x}}_i)$ and $\widetilde{\mathcal{K}}_\delta(\widetilde{\boldsymbol{x}}_i)$ is by no means optimal; cf. [19,22]. Ideally, we would employ a row- or column-wise loop assembly procedure and only loop over the required rows or columns as it is done in [6–9]; see Fig. 5.2. Instead, we decided to construct our tests using an established software which employs

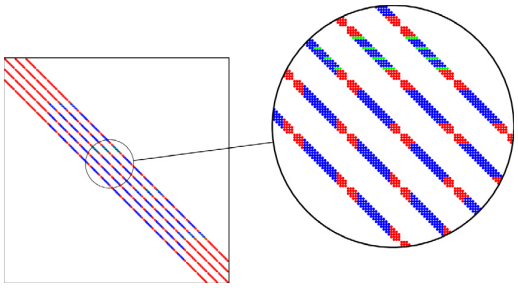


Fig. 5.2. Sparsity pattern of the surrogate mass matrix $\widetilde{\mathsf{M}}$ where $H = 5h$. The red and green points indicate the entries of the matrix which are evaluated with Gaussian quadrature. The blue points indicate the entries which are obtained by evaluating the surrogate stencil functions $\mathcal{M}_\delta$. The red points correspond to the basis functions near the boundaries and the green entries are used as supporting points for the interpolation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

element-wise loops and standard Gaussian quadrature. Because the vast majority of IGA software employs element-wise loops, our tests can provide references for many readers to predict how surrogate matrices could accelerate their own codes. One drawback of our decision is that in order to evaluate $\widetilde{\mathcal{M}}_\delta$ and $\widetilde{\mathcal{K}}_\delta$ at any single point $\widetilde{\boldsymbol{x}}_i$, we had to perform quadrature on every "active element" located in the support of the basis function centered at $\widetilde{\boldsymbol{x}}_i$; see Fig. 5.1. It is notable that we could easily overcome this wasteful expense to provide significant speed-ups; cf. Section 6. An explanation for this, using an estimate for the asymptotic number of required floating point operations, is given in Section 5.3.

5.2. Mesh-dependent sampling lengths

In this subsection, we recall the concept of mesh-dependent sampling lengths. Instead of using a fixed M for any mesh, we may allow M to depend on the mesh size h. Let H be the maximum distance in any Cartesian direction, between any two points in $\widetilde{\mathbb{X}}^s$, cf. (3.4). Recall that $q \geq 0$ is the order of the B-spline interpolation space used in constructing the surrogate stencil functions. Generally, the error in a surrogate matrix method has the following form:

$$\|u - \widetilde{u}_h\| \leq C_a h^{p+a} + C_b H^{q+b}\,, \tag{5.1}$$

where $\|\cdot\|$ is a generic norm and each $a, b \geq 0$, $C_a, C_b > 0$ are real-valued constants, independent of h. The first term on the right hand side of (5.1) controls the discretization error one finds in the standard IGA method; namely $\|u - u_h\| \leq C_a h^{p+a}$. The second term accounts for the loss of consistency in the surrogate method, $\|u_h - \widetilde{u}_h\| \leq C_b H^{q+b}$. See (4.2a) and (4.2b) for particular examples of such estimates in the case of the Helmholtz equation.

A necessary property is that the discretization error dominates the consistency error in the small mesh size limit $h \to 0$. That is, we must design $H^{q+b} = o(h^{p+a})$ because it will cause the surrogate method to have the same asymptotic accuracy as the method it is replacing. If H is related to h via a constant factor $M > 0$, i.e., $H = M \cdot h$, this property is guaranteed, so long as $q + b > p + a$. However, it is by no means necessary for H and h to be proportional to each other or even for $H = \mathcal{O}(h)$.

The best performance is achieved when $H = o(h)$; that is, when $H = M \cdot h$ and $M = H/h \to \infty$ as $h \to 0$. Provided that $q + b > p + a$, a natural choice which enforces this property is the definition

$$M(h) = \max\{1, \lfloor C \cdot h^{\epsilon - 1 + \frac{p+a}{q+b}} \rfloor\},$$

where $\epsilon, C > 0$ are a tunable parameters. Notice that this definition implies that $M(h)$ will grow more rapidly, in the $h \to 0$ limit, as the interpolation order q is increased. For further information about mesh-dependent sampling lengths, see [19, Section 7.3.3].

5.3. Floating point computational complexity

The time-to-solution in a simulation depends on the culmination of many factors, not solely the number of FLOPs. Indeed, good performance usually relies on a good problem-, scale-, and architecture-dependent balance between FLOPs and memory traffic. In this subsection we present a simple back-of-the-envelope complexity argument, based only on FLOPs, for the surrogate mass matrix $\widetilde{\mathsf{M}}$ defined in (3.5). A complexity argument for the surrogate stiffness matrix $\widetilde{\mathsf{K}}$ would be almost identical. The order estimates presented here should only be understood as crude predictions of the overall performance to be expected in practice.

Begin with an open uniform knot vector $\Xi = \{\xi_1, \dots, \xi_{m+p+1}\}$ and let it define the multivariate B-spline basis $\{\widehat{B}_i(\widehat{x})\}$, $i = 1, \dots, N$, described in Section 3.1. We assume that this B-spline basis forms the approximation space V_h used in the discretization of both M and $\widetilde{\mathsf{M}}$. As mentioned in the introduction, the best complexity of formation and assembly with Galerkin IGA may be as little as $\mathcal{O}(r^n p)$ with $r = p$ [5]. Employing standard element-wise Gaussian quadrature, the complexity increases to $\mathcal{O}(r^n p)$ for $r = p^2$. Of course, such an estimate has an implicit dependence on the mesh size $h = \frac{1}{m-p}$. Accounting for both h- and p-dependence, the IGA assembly has at least a complexity of $\mathcal{O}(N r^n p)$, where $N = \mathcal{O}(h^{-n})$ is the number of degrees of freedom.

We now argue that if $H = o(h)$, assembling the surrogate mass matrix (3.5), with a B-spline interpolation of order q, costs $\mathcal{O}(h^{-n} p^n q)$ FLOPs, with a small leading constant, regardless of the quadrature rule used. As usual in such analysis, we assume that the univariate B-spline or NURBS basis functions and their gradients are pre-evaluated at the quadrature points and stored in memory. This assumption, is not a great drawback because the knot vector Ξ is uniform and so the memory footprint of the univariate basis functions evaluated at the quadrature points is small.

In order to estimate the complexity of forming $\widetilde{\mathsf{M}}$, we must separately account for the cost of computing each of the different nonzero entries in the matrix. However, we immediately disregard the cost of enforcing symmetry and assume that it only changes the constants found in the final FLOP estimate.

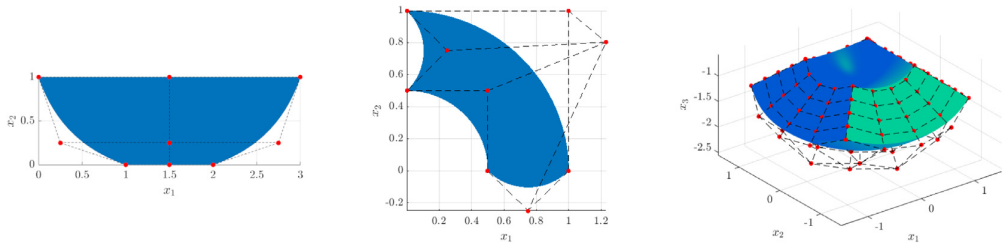
There are three different types of non-zero entries in $\widetilde{\mathsf{M}}$; see, e.g., Fig. 5.2. First, there are the entries computed by evaluating the surrogate stencil functions $\widetilde{\mathcal{M}}_{\boldsymbol{\delta}}$ at points in $\widetilde{\mathbb{X}}$; cf. the blue points in Fig. 5.2. There are $\mathcal{O}(p^n)$ surrogate stencil functions which need to be evaluated at $\#\widetilde{\mathbb{X}} = (m - 2p)^n$ rows. Employing sum-factorization, the final estimate of the surrogate stencil function evaluation is $\mathcal{O}(m^n p^n q) = \mathcal{O}(h^{-n} p^n q)$. Next, there are each of the non-zero coefficients coming from interaction with basis functions which do not have the cardinal structure; i.e., $\widehat{B}_i(\widehat{\boldsymbol{x}}) \neq \widehat{B}(\boldsymbol{x} - \widetilde{\boldsymbol{x}}_i)$, cf. the red points in Fig. 5.2. There are $\mathcal{O}(N - \#\widetilde{\mathbb{X}}) = \mathcal{O}(m^n - (m-2p)^n) = \mathcal{O}(m^{n-1}) = \mathcal{O}(h^{-n+1})$ basis functions without this structure. In turn, there are $\mathcal{O}(h^{-n+1})$ rows/column in $\widetilde{\mathsf{K}}$ which are filled in using standard IGA assembly procedures, thus providing an optimal complexity of $\mathcal{O}(h^{-n+1} r^n p)$. Asymptotically, as $h \to 0$, this contribution is negligible compared to the cost of evaluating the surrogate stencil functions. However, for large h, this term may significantly contribute to the total performance.

Lastly, there are the computations which must be performed in order to sample the stencil functions. Recall the identification $[\mathsf{M}]_{ij} = \mathcal{M}_{\boldsymbol{\delta}}(\widetilde{\boldsymbol{x}}_i)$. These coefficients are precisely those appearing at the green points in Fig. 5.2. Since each point in $\widetilde{\mathbb{X}}^s$ is at most a distance H apart, in each Cartesian direction, this leads to at most $\mathcal{O}(H^{-n})$ rows, each with a cost of $\mathcal{O}(r^n p)$. Written in terms of H, the cost of sampling has a total complexity of $\mathcal{O}(H^{-n} r^n p)$. Exploiting the tensor-product structure of the approximation space, the B-spline interpolation itself requires n LU decompositions of sparse univariate collocation matrices which are banded with bandwidth $\mathcal{O}(q)$. Computing the LU decomposition of one such banded matrix without pivoting requires $\mathcal{O}(H^{-1} q^2)$ operations [35]. Applying the forward and backward substitutions to the $\mathcal{O}(H^{-n+1})$ right-hand sides requires $\mathcal{O}(H^{-n} q)$ operations. Since the interpolation needs to be done for all $\mathcal{O}(p^n)$ stencil functions, the total cost of the interpolation step is $\mathcal{O}(H^{-1} p^n q^2 + H^{-n} p^n q)$ FLOPs.

Having accounted for the three different types of non-zero entries, it is now evident that the cost of forming $\widetilde{\mathsf{M}}$ can be separated into four contributions: evaluating the stencil functions ($\mathcal{O}(h^{-n} p^n q)$); numerical integration with the non-cardinal basis functions ($\mathcal{O}(h^{-n+1} r^n p)$); sampling the stencil functions ($\mathcal{O}(H^{-n} r^n p)$); and performing the interpolation of the stencil functions ($\mathcal{O}(H^{-1} p^n q^2 + H^{-n} p^n q)$). Since there are always $\mathcal{O}(h^{-n})$ rows in the final matrix, the average complexity per row is as follows:

$$\text{avg. cost} = \frac{\mathcal{O}(h^{-n} p^n q) + \mathcal{O}(h^{-n+1} r^n p) + \mathcal{O}(H^{-n} r^n p) + \mathcal{O}(H^{-1} p^n q^2 + H^{-n} p^n q)}{h^{-n}}.$$

In the small mesh size limit, employing a fixed sampling parameter $M > 0$ throughout the full sequence of meshes, we see that the complexity in p is still at least $\mathcal{O}(r^n p)$. However, in this setting, the constant factor in the



(a) Convex domain Ω_1 (b) Non-convex domain Ω_2 (c) Non-convex domain Ω_3

Fig. 6.1. Domains considered for the Helmholtz problem.

$\mathcal{O}(r^n p)$ term is proportional to $1/M > 0$, which may be very small. In general,

$$\text{avg. cost} = \mathcal{O}(p^n q) + \mathcal{O}(r^n p), \qquad \text{if } H = \mathcal{O}(h).$$

In the case of a mesh-dependent sampling length, $\lim_{h\to 0} \frac{h}{H} = 0$, so the complexity estimate is improved. Indeed,

$$\text{avg. cost} = \mathcal{O}(p^n q), \qquad \text{if } H = o(h).$$

Some remarks about these estimates are now in order. From the deductions above, it is clear that for any interpolation order $q > p$, forming $\widetilde{\mathsf{M}}$ will have a poorer floating point complexity than $\mathcal{O}(p^{n+1})$, which is achievable with some other methods [5]. Nevertheless, experience has lead the authors to conclude that it tends to actually be very desirable to select a large $q > p$ when forming any surrogate matrix. Although it is quite clear that a large q positively influences the convergence rate of the consistency error term in (5.1), we have seen very little change in performance with any q we have studied. One reason for this may be that the constant factor in the $\mathcal{O}(p^n q)$ estimate is extremely small; in particular, much smaller than the constant in front of the $\mathcal{O}(r^n p)$ terms attributed to performing quadrature. We posit that this may be the case because the term derives only from function evaluation which tend to be very cache-aware operations. Note that the experiments and measurements in Section 6 use an implementation with element-loop assembly and Gaussian quadrature; i.e., $r = p^2$.

6. Numerical examples

In order to show the applicability and efficiency of the presented methods, we performed a set of numerical experiments which are documented here. In Section 6.1, we consider the Helmholtz equation with various boundary conditions, and in Section 6.2 the same problem with a non-constant wave number. In Section 6.3, we consider a time harmonic problem involving linear elasticity. Finally, in Section 6.4, we consider a nonlinear, transient, hyperelastic wave propagation example.

All run-time measurements were obtained on a machine equipped with two Intel® Xeon® Gold 6136 processors with a nominal base frequency of 3.0 GHz. Each processor has 12 physical cores which results in a total of 24 physical cores. The total available memory of 251 GB is split into two NUMA domains; one for each socket.

6.1. Helmholtz equation

In this subsection, we investigate the surrogate matrix methodology in case of the Helmholtz example (2.5) and verify the theoretical results stated in Theorem 4.1. We investigate the problem on three representative domains. Namely, the convex domain Ω_1 depicted in Fig. 6.1(a) and two non-convex domains; the quarter annulus with bumps Ω_2 depicted in Fig. 6.1(b) and the part of a spherical shell Ω_3 shown in Fig. 6.1(c).

In the first set of experiments, we fix the trial space V_h and the surrogate matrix parameters and vary the wave number. This will indicate dependence of the various errors on the wave number k. For Ω_1 and Ω_2 we fix $m = 640$ and $M = 5$, and for Ω_3, $m = 100$ and $M = 17$. In each setting, we set $p = 2$ and $q = 5$. Let $\mathcal{H}_a^{(1)}$ be a Hankel function of the first kind and $r = \|\mathbf{x}\|$. As analytical solutions, we choose

$$u(r) = \frac{\mathrm{i}}{4}\mathcal{H}_0^{(1)}(kr) \text{ in 2D} \qquad \text{and} \qquad u(r) = \frac{\mathrm{i}}{4r}\mathrm{e}^{\mathrm{i}kr} \text{ in 3D.} \tag{6.1}$$

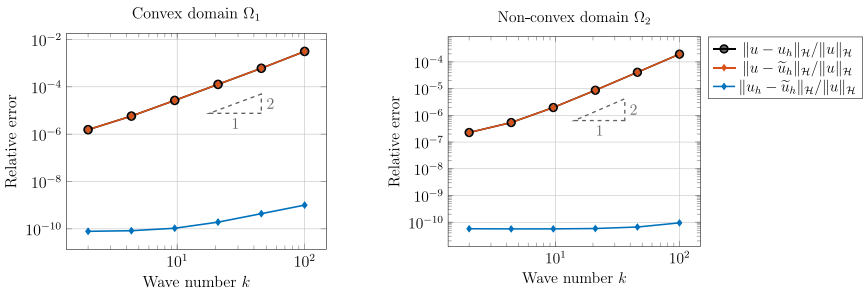


Fig. 6.2. Demonstration of k-dependency on the various errors for the Helmholtz problem on Ω_1 and Ω_2, respectively.

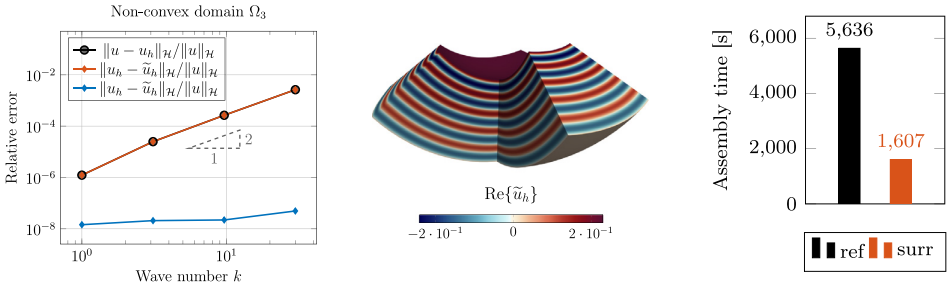


Fig. 6.3. Demonstration of k-dependency on the various errors for the Helmholtz problem on Ω_3 (left). Plot of the real part of the surrogate solution for $k = 30$ (center). Assembly time comparison between reference (ref) and surrogate (surr) method (right).

These choices yield $f = 0$ so long as the origin is not included in the domain. The Robin-type boundary term g is computed by using the analytical solution u.

The relative errors in the $\mathcal{H}$-norm are presented in Fig. 6.2 for the first two domains and on the left-hand side of Fig. 6.3 for Ω_3. Additionally, we plot the real part of the surrogate solution for $k = 30$ in the center of Fig. 6.3 and show the assembly time comparison between the standard and surrogate method on the right-hand side of Fig. 6.3. Already, with a fixed $M = 17$, a speed-up of about 251% can be observed. The discretization error in all cases grows like k^p, as predicted in Theorem 4.1. Moreover, the relative consistency error in the $\mathcal{H}$-norm is almost independent of k. This agrees well enough with our predictions since the assumptions made in Section 2.3 may not hold for the very highest wave numbers we considered.

For our second set of experiments, we consider the non-convex domain Ω_2 and the same 2D analytical solution defined in (6.1). Here, we vary h but fix $q = 5$, use the mesh-dependent sampling parameter $M = \max\left\{1, \lfloor 0.5 \cdot m^{1-\frac{p+1}{q+1}} \rfloor\right\}$, and consider each $k \in \{8, 16, 32, 64, 128\}$. For this scenario, we plot relative total errors, relative consistency errors, and speed-ups versus $\frac{m}{k} \propto \frac{1}{kh}$.

The relative errors in the $\mathcal{H}$-norm for the selected wave numbers $k \in \{8, 128\}$ can be observed in the plot on the left-hand side of Fig. 6.4. Here, both $\frac{\|u-u_h\|_{\mathcal{H}}}{\|u\|_{\mathcal{H}}}$ and $\frac{\|u-\widetilde{u}_h\|_{\mathcal{H}}}{\|u\|_{\mathcal{H}}}$ are presented. For a common wave number k, the two relative error curves lie almost perfectly on top of each other and clearly demonstrate the estimated optimal order of convergence, $\mathcal{O}((\frac{m}{k})^{-2})$. In the center plot of Fig. 6.4, we present the relative consistency errors for each $k \in \{8, 16, 32, 64, 128\}$. From this plot, it is both obvious that the consistency errors are much smaller than the corresponding discretization errors and that they do not have any notable dependence on the ratio m/k. On the right-hand side of Fig. 6.4, the speed-ups of the assembly time for those wave numbers are presented. The largest speed-up of 3178% may be observed for $k = 128$ on the finest mesh.

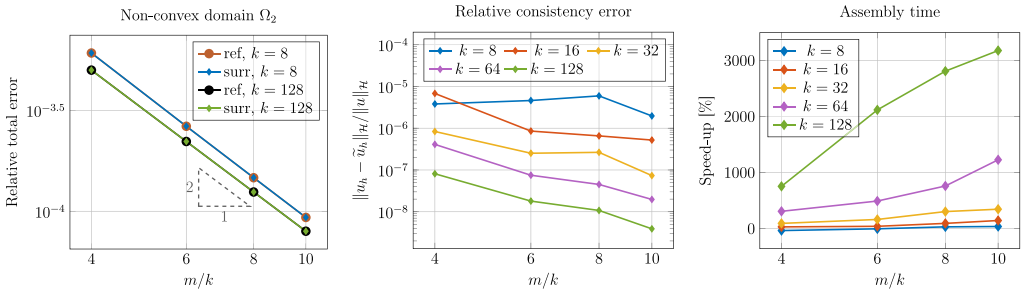


Fig. 6.4. Relative reference and surrogate errors in the $\mathcal{H}$-norm for different wave numbers k computed on the non-convex domain Ω_2 (left). Relative consistency errors (center) and speed-up of the assembly time of the same problem (right). Recall here that $M = \max\left\{1, \lfloor 0.5 \cdot m^{1-\frac{p+1}{q+1}} \rfloor\right\}$.

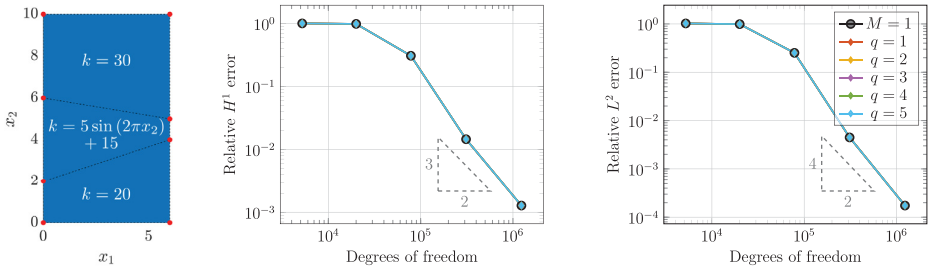


Fig. 6.5. Wedge domain (left). Relative $H^1(\Omega)$ and $L^2(\Omega)$ errors for $p = 3$, $M = 5$, and the manufactured solution $u(x_1, x_2) = \sin(20\pi x_1)\sin(20\pi x_2)$ for the Helmholtz problem with non-constant wave number $k(x_1, x_2)$ from (6.2) (center and right).

6.2. Helmholtz equation with non-constant wave number

In this subsection, we consider the Helmholtz equation (2.5) in which the wave number is non-constant over the physical domain. This type of problem occurs, for instance, in the modeling of acoustic waves with heterogeneous wavespeed; see, e.g., [36] and the references therein. Here, we use an example inspired from [37] on the wedge domain $(0, 6) \times (0, 10) \subseteq \mathbb{R}^2$ presented in the left of Fig. 6.5. The domain is discretized with three patches. In the top and bottom patches, we utilize a constant wave number and in the central patch, we use a spatially varying wave number. This choice introduces a jump in the coefficient along the patch interfaces. Since the matrix entries corresponding to the basis functions close to the patch boundaries are integrated by the standard approach, the surrogate method is not impeded by this discontinuity. In particular, we choose the spatially varying wave number

$$k(x_1, x_2) = \begin{cases} 20 & \text{for } 0 \le x_2 < \frac{x_1}{3} + 2, \\ 5\sin(2\pi x_2) + 15 & \text{for } \frac{x_1}{3} + 2 \le x_2 < 6 - \frac{x_1}{6}, \\ 30 & \text{for } 6 - \frac{x_1}{6} \le x_2 \le 10, \end{cases} \tag{6.2}$$

and we consider two settings.

In the first one, we consider a manufactured solution $u(x_1, x_2) = \sin(20\pi x_1)\sin(20\pi x_2)$ which we use to obtain the right-hand side f and g in (2.5). In the center and right of Fig. 6.5, we present the relative $H^1(\Omega)$ and $L^2(\Omega)$ errors for $p = 3$, $M = 5$, $q \in \{1, 2, 3, 4, 5\}$, and decreasing h. We observe that the surrogate method is able to reproduce the solution of the standard approach. Moreover, the surrogate solutions exhibit the same error convergence rates as the reference solution ($M = 1$) for all choices of q.

In the second setting, we provide only the source and boundary terms which do not stem from a manufactured solution. This allows us to compare the discrete solutions only. We choose $f(\mathbf{x}) = \frac{1}{\pi a}\exp(-\|\mathbf{x} - \mathbf{c}\|^2 a^{-2})$ with $a = 5 \cdot 10^{-3}$, $\mathbf{c} = (3, 9.5)^\mathsf{T}$, $g = 0$, and $k(x_1, x_2)$ from (6.2). For the discretization parameters, we choose $p = 3$, $q \in \{1, 2, 3\}$, and $M = 5$. In the left of Fig. 6.6, we present the real part of the solution obtained with the standard

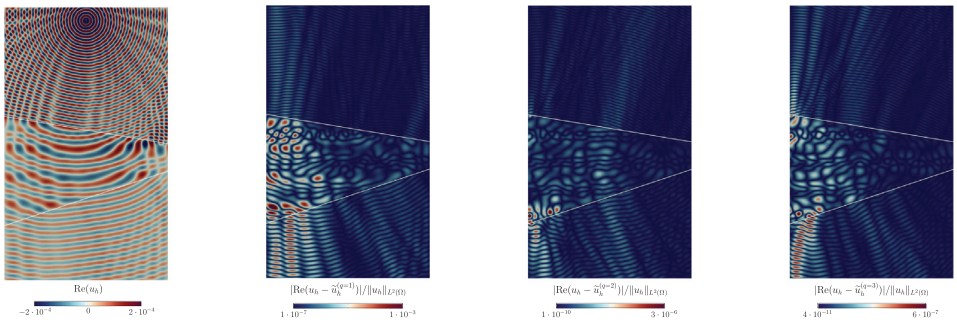


Fig. 6.6. Real part of the solution obtained with the standard approach and $p = 3$ for the Helmholtz problem with non-constant wave number $k(x_1, x_2)$ from (6.2) and non-manufactured solution (left). Relative difference of the real part of the standard solution and the real part of each of the surrogate solutions with increasing $q \in \{1, 2, 3\}$ (second from left to right).

approach ($M = 1$). The solutions obtained with the surrogate method do not differ visually. We emphasize this fact by showing the relative differences of the real part of the standard solution and the real part of each of the surrogate solutions in the right of Fig. 6.6. We observe that the differences decrease with increasing q and that the largest differences are located within the patch in which the wave number varies as well as in the left part of the bottom patch.

6.3. Linear elastodynamics with periodic pressure loading

In this set of experiments, we focus on the time-harmonic linear elastodynamics problem (2.2) with the energy density functional W in (2.3). We start with (2.1) and apply a pressure that fluctuates periodically, with angular frequency ω, in the interior of the circular hole. The setup is depicted in Fig. 6.7. Let $\boldsymbol{\sigma} = \partial_{\boldsymbol{u}} W(\boldsymbol{u})$. On the circular boundary, we apply a time-dependent pressure of the form $-\boldsymbol{\sigma}\boldsymbol{n}\cdot\boldsymbol{n} = p(t) = \frac{p_0}{\sqrt{2\pi}} \mathrm{e}^{-\mathrm{i}\omega t}$. The problem is transformed into the frequency domain resulting in the time harmonic equations (2.2). In this particular experiment, we choose $\omega = 50\pi$, $p_0 = 1$, $r_0 = 1$, $L = 4$, $\lambda = 2$, $\mu = 1$, and $\rho_0 = 1$. The analytical solution to this problem may be written in polar coordinates (r, θ) as follows [38]:

$$u_r(r) = -\frac{p_0 r_0 \zeta H_1^{(2)}\left(\frac{\omega r}{\gamma}\right)}{\mu\left(\gamma \omega r_0 H_0^{(2)}(\omega r_0) - 2\zeta H_1^{(2)}(\omega r_0)\right)}.$$

Here, $\mathcal{H}_a^{(2)}$ is the Hankel function of the second kind, $\gamma = \sqrt{\frac{\lambda+2\mu}{\rho_0}}$, and $\zeta = \sqrt{\frac{\mu}{\rho_0}}$.

We investigate PML absorbing boundary conditions. For this problem, the stiffness matrix K and mass matrix M in (2.2) need to be assembled. For the surrogate matrices, this is achieved by employing definitions similar to (3.5) and (3.6) but for the vector-valued setting mentioned in Remark 3.4. We prescribe symmetric boundary conditions on Γ_1 and Γ_4 and the pressure is applied to Γ_5. On the remaining boundaries, Γ_2 and Γ_3, homogeneous Dirichlet boundary conditions are prescribed. The region of interest and the PML region are separated through ℓ. Each physical coordinate x_k for $k = 1, 2$ is mapped according to the following specific stretching function from [39,40]:

$$\widetilde{x}_k = \begin{cases} x_k & \text{if } 0 < x_k \leq \ell, \\ x_k + \mathrm{i}\frac{C}{\omega}\left(\frac{x_k - \ell}{L - \ell}\right)^n & \text{if } \ell < x_k \leq L, \end{cases} \tag{6.3}$$

where $\ell = 3$, $n = 2$, and $C = 5$.

As with the second set of experiments with the Helmholtz equation, we also choose to adopt a mesh-dependent sampling parameter $M = M(h)$ which balances of the error and performance in our favor. Here, we consider $M(h) = \max\left\{2, \left\lfloor 2 \cdot h^{\frac{p-q+1/2}{q+1}} \right\rfloor\right\}$, where it is implicitly understood that $q > p$.

For $p = 2$ and $q = 5$, the real part of the surrogate solution and difference from the standard solution are presented in Fig. 6.8. Note that the largest difference is observed in the PML region. This can be attributed to the

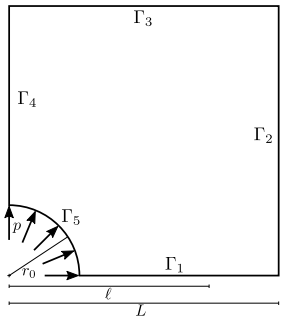


Fig. 6.7. Plate with circular hole setup in linear elastodynamics problem.

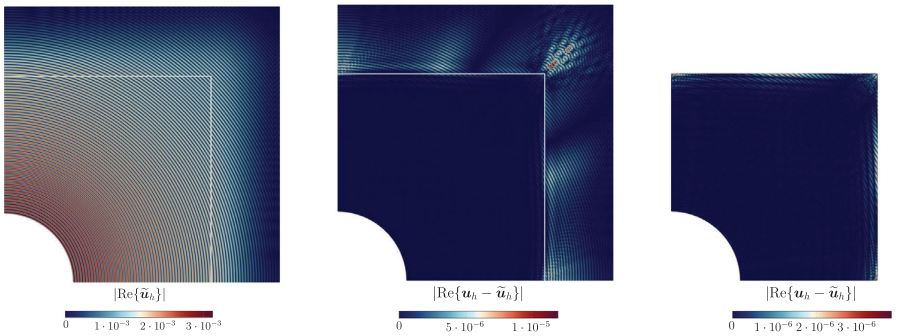


Fig. 6.8. Magnitude of the real part of the solution on the finest mesh with $\omega = 50\pi$ and PML boundary conditions. Surrogate IGA solution with $q = 5$ and mesh-dependent sampling parameter M (left). Difference between standard and surrogate IGA solutions (center and right).

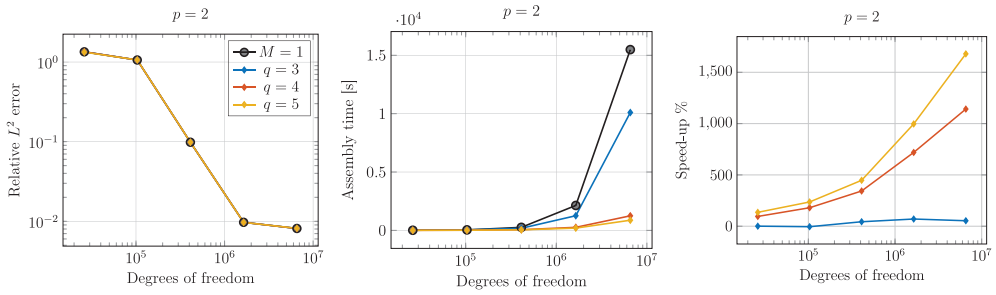


Fig. 6.9. Relative L^2 errors in $\Omega \cap (0, \ell)^2$ and assembly times for periodic pressure loading with $\omega = 50\pi$ and $p = 2$ computed with PML absorbing boundary conditions.

fact that the stencil functions are also affected by the stretching function (6.3). Inspecting the right-hand side of Fig. 6.8, it is clear that the difference in the two solutions in the domain of interest, $\Omega \cap (0, \ell)^2$, is an order of magnitude less than the difference in the two solutions in the PML.

Relative L^2 errors in $\Omega \cap (0, \ell)^2$ and the associated assembly times are shown in Fig. 6.9. For all h and q considered, $\|\boldsymbol{u} - \boldsymbol{u}_h\|_{L^2(\Omega)}$ is indistinguishable from $\|\boldsymbol{u} - \widetilde{\boldsymbol{u}}_h\|_{L^2(\Omega)}$. We do not observe asymptotic error convergence, even in the standard IGA case, because of the presence of the PML. For $q = 5$ on the finest mesh, we observe a speed-up of 1679%, without any degradation in the L^2 error.

We refrain from showing non-harmonic linear elastodynamic examples. They are only of little relevance because the stiffness matrix K needs to be computed only once in the first time step and can be reused for each subsequent

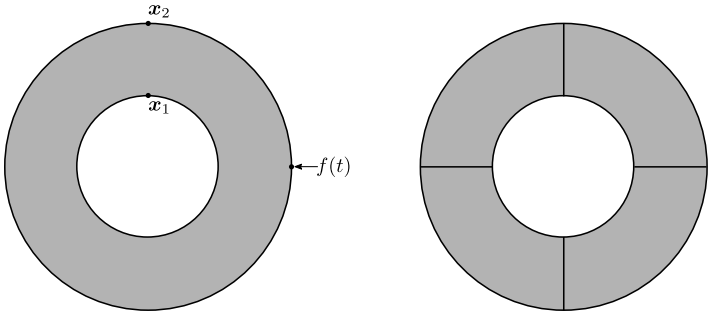


Fig. 6.10. Problem setup (left) and patches (right).

step. Nevertheless, the surrogate approach may be of interest when applied in a matrix-free setting since K would need to be recomputed for each matrix–vector product.

6.4. Nonlinear hyperelastic waves

In this final set of experiments, we consider transient, nonlinear, hyperelastic wave propagation obeying (2.1) with the energy density functional (2.4). The problem setup is illustrated in Fig. 6.10. As domain, we choose the annulus $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : 1 < \|\mathbf{x}\| < 2\}$ and the time interval $[0, T]$, with $T = 7.5$. We employ the material parameters $\rho_0 = 1$, $E = 1$, and $\nu = 0.35$. The corresponding Lamé parameters are obtained via the expressions $\mu = \frac{E}{2(1+\nu)}$ and $\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}$. We prescribe zero initial displacements and zero initial velocity; i.e., $\mathbf{u}_0 = \mathbf{v}_0 = \mathbf{0}$. At the boundary $\partial\Omega$, we apply a pointwise force pulse at the right side, i.e., $\partial_{\boldsymbol{u}} W(\boldsymbol{u})\boldsymbol{n} = \delta(x_1 - 2) \cdot f(t) \cdot (-1, 0)^{\mathsf{T}}$, with

$$f(t) = \begin{cases} \frac{1}{10} \sin\left(\frac{\pi t}{t_f}\right) & t \le t_f, \\ 0 & \text{otherwise}, \end{cases}$$

where $t_f = 1/5$. The computational domain is split into four patches, as depicted on the right hand side of Fig. 6.10, each of which is discretized with $m = 100$ and $p = 2$. The sub-matrices belonging to each patch are assembled in parallel.

For the time-discretization, we employ the nonlinear generalized-α method described in [41] with the damping parameters $\rho_\infty = \frac{1}{2}$, $\alpha_m = \frac{1}{2}\frac{3-\rho_\infty}{1+\rho_\infty}$, and $\alpha_f = \frac{1}{1+\rho_\infty}$. Moreover, we choose $\Delta t = 5 \cdot 10^{-3}$ as the time step size. At each time step, we perform two Newton iterations. In each iteration the nonlinear matrix is being reassembled using either the standard approach or the surrogate matrix approach, with $q = 5$ and $M = 18$. Since the mass matrix term does not change over time, we assemble it once using the standard approach and re-use it in the subsequent steps.

In order to compare the solutions of the standard and surrogate method, we record the displacement in y-direction over time at two positions, $\mathbf{x}_1 = (0, 1)^{\mathsf{T}}$ and $\mathbf{x}_2 = (0, 2)^{\mathsf{T}}$, in Fig. 6.11. No visual difference can be observed. On the left-hand side of Fig. 6.12, we present the kinetic, internal, and total energy divided by two versus the time $t \ge 0.3$ for both approaches. We observe, once again, no visual difference in the two solutions. The central plot in Fig. 6.12 shows the time required to complete each time step. For each time step, this required time includes the right-hand side evaluation as well as the reassembly and inversion of the tangent matrices for each Newton iteration. We use the MATLAB backslash operator to invert the emerging systems which takes on average 6.41 s per time step independent of which assembly method is used. In total, inverting the tangent matrices in the standard approach takes up approximately 10.7% of the time and approximately 26.6%, if the surrogate method is used. For the sake of completeness, we present the accumulated total time and the accumulated time required for the inversion of the systems in the right-hand side of Fig. 6.12. In this scenario, a speed-up of about 142% may be observed. Finally, in Fig. 6.13, we illustrate the von Mises stress at different times. The faster traveling body waves reach the point $\mathbf{x}_2$ first, followed by the surface waves, which result in the greatest displacements; cf. Fig. 6.11.

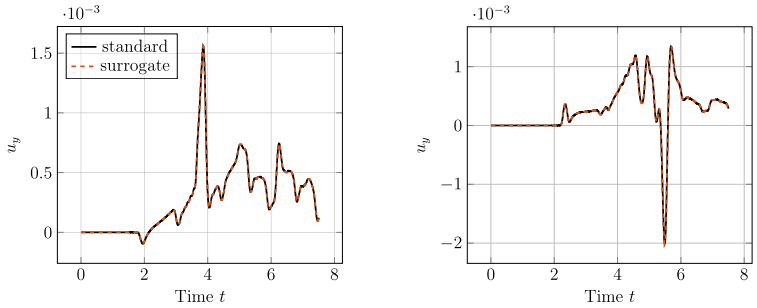


Fig. 6.11. Displacements in y-direction u_y recorded at $\mathbf{x}_1 = (0, 1)^\mathsf{T}$ (left) and $\mathbf{x}_2 = (0, 2)^\mathsf{T}$ (right) for the standard and surrogate method.

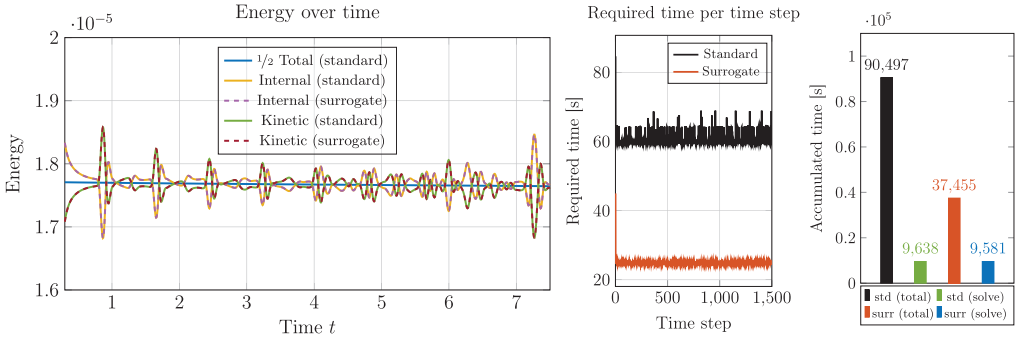


Fig. 6.12. Total divided by two, kinetic, and internal energy over the time interval $[0.3, T]$ (left). Required time per time step (center). Accumulated total time and accumulated time spent on the inversion of the tangent matrices (right).

7. Conclusion

In this work, we applied the surrogate matrix methodology to several problems emerging in the investigation of waves in the isogeometric setting. We performed an a priori error analysis for the Helmholtz equation which demonstrated that the additional consistency error introduced by the presence of surrogate matrices is independent of the wave number. Moreover, we presented a floating point analysis showing that the computational complexity of the methodology compares favorably to other state-of-the-art assembly techniques for isogeometric analysis.

We confirmed the theoretical error estimates for the Helmholtz equation by performing benchmark computations showing the correct convergence behavior. We furthermore showed that the methodology is beneficial when applied to wave problems with PML absorbing boundary conditions by considering a linear problem in elastodynamics. Finally, we applied the methodology to a transient, nonlinear, hyperelastic wave propagation problem with a material modeled by a compressible neo-Hookean material. This last example showed the efficacy of the methodology for implicit time stepping schemes in which a nonlinear problem is solved by Newton's method in each time step. Our numerical experiments demonstrate clear performance gains in all experiments and we observed speed-ups of up to 3178%, when compared to the reference assembly algorithm, without losing any significant accuracy.

Thus far, in order to address the feasibility of this methodology in isogeometric analysis, we have relied on the MATLAB software GeoPDEs [23]. This software greatly lends itself to rapid prototyping but is not actually suitable for high performance experiments and it only supports element-loop assembly. Element-loop assembly is not necessary for the surrogate matrix methodology. Row/column-loop assembly would only provide better performance. Nevertheless, because our implementations employ element-loop assembly, which is presently the dominant assembly strategy employed in IGA software, it allows our experiments to directly suggest how surrogate matrices would perform in many other codes.

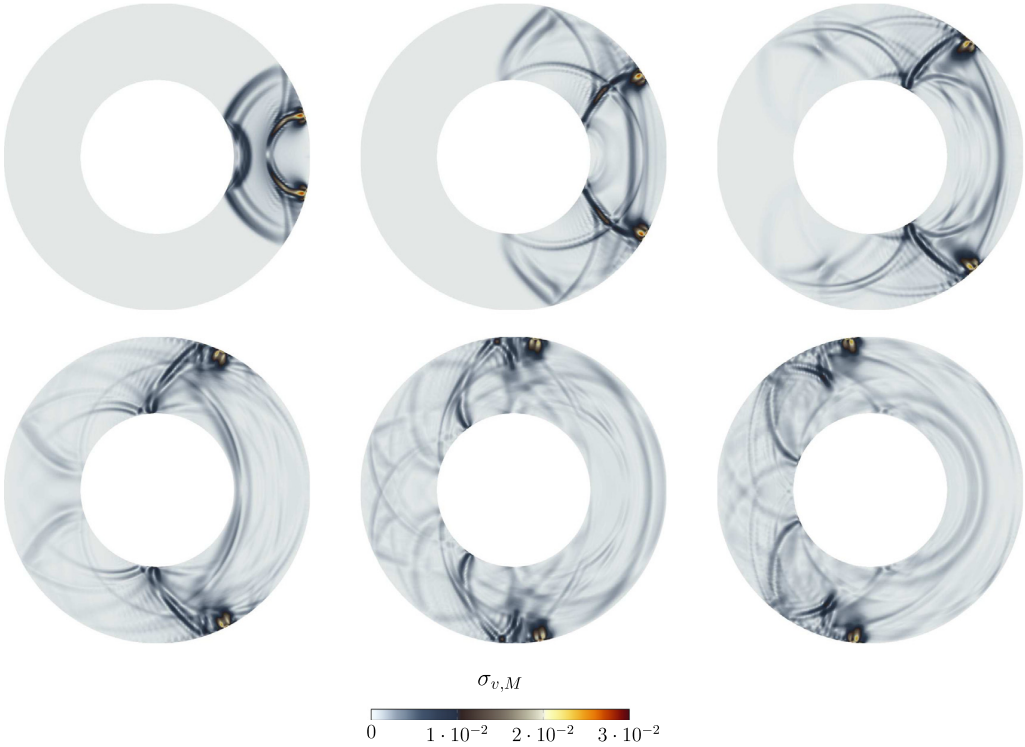


Fig. 6.13. Von Mises stress $\sigma_{v,M}$ in the nonlinear hyperelastic wave problem at times $t \in \{1, 2, 3, 4, 5, 6\}$.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank René Hiemstra for his insightful conversations about our work. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 800898. This work was also partly supported by the German Research Foundation through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) and by grant WO671/11-1.

References

[1] A. Bressan, S. Takacs, Sum factorization techniques in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 352 (2019) 437–460.

[2] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, G. Sangalli, Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization, Comput. Methods Appl. Mech. Engrg. 285 (2015) 817–828.

[3] F. Calabrò, G. Sangalli, M. Tani, Fast formation of isogeometric Galerkin matrices by weighted quadrature, Comput. Methods Appl. Mech. Engrg. 316 (2017) 606–622, Special Issue on Isogeometric Analysis: Progress and Challenges.

[4] G. Sangalli, M. Tani, Matrix-free weighted quadrature for a computationally efficient isogeometric k-method, Comput. Methods Appl. Mech. Engrg. 338 (2018) 117–133.

[5] R.R. Hiemstra, G. Sangalli, M. Tani, F. Calabrò, T.J. Hughes, Fast formation and assembly of finite element matrices with application to isogeometric linear elasticity, Comput. Methods Appl. Mech. Engrg. 355 (2019) 234–260.

[6] S. Bauer, M. Mohr, U. Rüde, J. Weismüller, M. Wittmann, B. Wohlmuth, A two-scale approach for efficient on-the-fly operator assembly in massively parallel high performance multigrid codes, Appl. Numer. Math. 122 (2017) 14–38.

[7] S. Bauer, M. Huber, M. Mohr, U. Rüde, B. Wohlmuth, A new matrix-free approach for large-scale geodynamic simulations and its performance, in: International Conference on Computational Science, Springer, 2018, pp. 17–30.
[8] S. Bauer, M. Huber, S. Ghelichkhan, M. Mohr, U. Rüde, B. Wohlmuth, Large-scale simulation of mantle convection based on a new matrix-free approach, J. Comput. Sci. 31 (2019) 60–76.
[9] D. Drzisga, B. Keith, B. Wohlmuth, The surrogate matrix methodology: A priori error estimation, SIAM J. Sci. Comput. 41 (6) (2019) A3806–A3838.
[10] R.R. Hiemstra, F. Calabrò, D. Schillinger, T.J. Hughes, Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 316 (2017) 966–1004.
[11] A. Mantzaflaris, B. Jüttler, B.N. Khoromskij, U. Langer, Low rank tensor methods in Galerkin-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 316 (2017) 1062–1085.
[12] C. Hofreither, A black-box low-rank approximation algorithm for fast matrix assembly in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 333 (2018) 311–330.
[13] A. Mantzaflaris, B. Jüttler, Integration by interpolation and look-up for Galerkin-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 284 (2015) 373–400.
[14] A. Mantzaflaris, B. Jüttler, Exploring matrix generation strategies in isogeometric analysis, in: M. Floater, T. Lyche, M.-L. Mazure, K. Mørken, L.L. Schumaker (Eds.), Mathematical Methods for Curves and Surfaces, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 364–382.
[15] F. Fahrendorf, L.D. Lorenzis, H. Gomez, Reduced integration at superconvergent points in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 328 (2018) 390–410.
[16] T.J. Hughes, A. Reali, G. Sangalli, Efficient quadrature for NURBS-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 199 (5–8) (2010) 301–313.
[17] F. Auricchio, F. Calabrò, T.J. Hughes, A. Reali, G. Sangalli, A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 249 (2012) 15–27.
[18] M. Pan, B. Jüttler, A. Giust, Fast formation of isogeometric galerkin matrices via integration by interpolation and look-up, Comput. Methods Appl. Mech. Engrg. 366 (2020) 113005.
[19] D. Drzisga, B. Keith, B. Wohlmuth, The surrogate matrix methodology: Low-cost assembly for isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 361 (2020) 112776.
[20] S. Brenner, R. Scott, The Mathematical Theory of Finite Element Methods, Vol. 15, Springer Science & Business Media, 2007.
[21] D. Schillinger, S.J. Hossain, T.J. Hughes, Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 277 (2014) 1–45.
[22] D. Drzisga, B. Keith, B. Wohlmuth, The surrogate matrix methodology: A reference implementation for low-cost assembly in isogeometric analysis, MethodsX (2020) 100813.
[23] C. de Falco, A. Reali, R. Vázquez, GeoPDEs: A research tool for isogeometric analysis of PDEs, Adv. Eng. Softw. 42 (12) (2011) 1020–1034.
[24] R. Vázquez, A new design for the implementation of isogeometric analysis in Octave and Matlab: GeoPDEs 3.0, Comput. Math. Appl. 72 (3) (2016) 523–554.
[25] J.M. Melenk, S. Sauter, Wavenumber explicit convergence analysis for Galerkin discretizations of the Helmholtz equation, SIAM J. Numer. Anal. 49 (3) (2011) 1210–1243.
[26] S. Esterhazy, J.M. Melenk, An analysis of discretizations of the Helmholtz equation in L2 and in negative norms, Comput. Math. Appl. 67 (4) (2014) 830–853.
[27] J.M. Melenk, On Generalized Finite Element Methods (Ph.D. thesis), research directed by Dept. of Mathematics. University of Maryland at College Park, 1995.
[28] P. Cummings, X. Feng, Sharp regularity coefficient estimates for complex-valued acoustic and elastic Helmholtz equations, Math. Models Methods Appl. Sci. 16 (01) (2006) 139–160.
[29] T.J. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39–41) (2005) 4135–4195.
[30] I.J. Schoenberg, Cardinal Spline Interpolation, Vol. 12, SIAM, 1973.
[31] I.J. Schoenberg, Contributions to the problem of approximation of equidistant data by analytic functions. Part A. on the problem of smoothing or graduation. A first class of analytic approximation formulae, Quart. Appl. Math. 4 (1946) 45–99.
[32] I.J. Schoenberg, Contributions to the problem of approximation of equidistant data by analytic functions. Part B. On the problem of osculatory interpolation. A second class of analytic approximation formulae, Quart. Appl. Math. 4 (1946) 112–141.
[33] J.-P. Berenger, A perfectly matched layer for the absorption of electromagnetic waves, J. Comput. Phys. 114 (2) (1994) 185–200.
[34] P.J. Matuszyk, L.F. Demkowicz, Parametric finite elements, exact sequences and perfectly matched layers, Comput. Mech. 51 (1) (2013) 35–45.
[35] G.H. Golub, C.F. Van Loan, Matrix Computations, fourth ed., Johns Hopkins, 2013.
[36] J. Chan, R.J. Hewett, T. Warburton, Weight-adjusted discontinuous Galerkin methods: wave propagation in heterogeneous media, SIAM J. Sci. Comput. 39 (6) (2017) A2935–A2961.
[37] Y.A. Erlangga, C.W. Oosterlee, C. Vuik, A novel multigrid based preconditioner for heterogeneous Helmholtz problems, SIAM J. Sci. Comput. 27 (4) (2006) 1471–1492.
[38] E. Kausel, Fundamental Solutions in Elastodynamics: A Compendium, Cambridge University Press, 2006.
[39] C. Michler, L. Demkowicz, J. Kurtz, D. Pardo, Improving the performance of perfectly matched layers by means of hp-adaptivity, Numer. Methods Partial Differential Equations 23 (4) (2007) 832–858.
[40] A.V. Astaneh, B. Keith, L. Demkowicz, On perfectly matched layers for discontinuous Petrov–Galerkin methods, Comput. Mech. 63 (6) (2018) 1131–1145.
[41] J.A. Cottrell, T.J. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, John Wiley & Sons, 2009.

B. Further articles

B.1. Stencil scaling for vector-valued PDEs on hybrid grids with applications to generalized Newtonian fluids

Stencil scaling for vector-valued PDEs on hybrid grids with applications to generalized Newtonian fluids

Daniel Drzisga, Ulrich Rüde, and Barbara Wohlmuth

The stencil scaling approach for low-order finite element discretizations presented in [14] marked itself successful for accelerating matrix-free methods on HHGs. However, only scalar elliptic PDEs were considered there. In this work, we investigate the applicability of this approach to vector-valued PDEs and observe that the idea of the scalar stencil scaling cannot be applied straightaway. We show that the simple scaling for vector-valued PDEs results in the discretization of a different PDE, if the PDE coefficient is not constant. In order to overcome this issue, we develop a new stencil scaling approach by adding a correction term to the discrete stencils. Although this vector-valued scaling is more expensive than the original scalar stencil scaling, it still has the ability to reproduce the standard finite element solution in shorter time compared to the standard on-the-fly stencil assembly.

In this work, we focus on variable coefficient vector-valued partial differential equations as they emerge in the modeling of many physical phenomena. The presented method is based on scaling reference stencils originating from a linear finite element discretization of a constant coefficient PDE. This method assumes the utilization of HHGs and it may be applied to vector-valued second-order elliptic partial differential equations directly or as part of more complicated problems. The major novelty of this work is the presentation of an improved method to assemble the stencils for these problems, particularly appropriate for matrix-free solvers.

In Section 2, we state the model problem and introduce the stencil scaling approach with the required correction term. Moreover, the efficient pre-computation of the correction terms in 2D and 3D is presented. In Section 3, we describe an efficient approach to compute a regularized strain rate which is then used to define the node-wise viscosities used in the numerical examples. In Section 4, we provide theoretical computational complexity estimates demonstrating the advantages of this new approach compared to the traditional on-the-fly integration and stored matrix approaches. In Section 5, we verify the theoretical complexity analysis from Section 4 by performing a roofline analysis for residual computations and the underlying MVPs. Furthermore, we demonstrate the error convergence rates and the run-time of this extended stencil scaling through a number of numerical experiments. In particular, we consider experiments based on the mathematical models of linear elastostatics and generalized incompressible Stokes flow. In the concluding example, a nonlinear shear-thinning non-Newtonian example is considered. The largest considered example involved solving an incompressible Stokes problem utilizing 12 288 compute cores on the state of the art supercomputer SuperMUC-NG. Finally, in Section 6, we give some concluding remarks.

I was significantly involved in finding the ideas and primarily responsible for setting up the mathematical framework and carrying out the scientific work presented in this article. Furthermore, I was in charge of writing the article while the co-authors contributed by making corrective changes.

Permission to include:

Daniel Drzisga, Ulrich Rüde, and Barbara Wohlmuth
Stencil scaling for vector-valued PDEs on hybrid grids with applications to generalized Newtonian fluids
SIAM Journal on Scientific Computing 42.6 (2020): B1429–B1461
(see also article [45] in the bibliography)

On the following page, a copy of the first page of the consent to publish agreement by SIAM may be found. This page includes the author's rights. A digital version of the consent to publish form may be found at

https://www.siam.org/publications/journals/about-siam-journals/information-for-authors

(Accessed on 22 March 2020)

Society for Industrial and Applied Mathematics (SIAM)

Consent to Publish

SIAM ("Publisher") requires Authors of articles in SIAM publications to provide a formal written Consent to Publish. The Author must sign the agreement except, in the case of "work-for-hire", when the Author's employer may sign as the party that has the right to grant rights to the Publisher. If there are multiple Authors of the material governed by this document, the term "Author" as used here refers to each and all of them, jointly and severally. [1]

Title of Contribution ("Work"): Stencil scaling for vector-valued PDEs with applications to generalized Newtonian fluids
Authors: Daniel Drzisga, Ulrich Rüde, and Barbara Wohlmuth
Name of Journal: SIAM Journal on Scientific Computing
Manuscript Number: M126789

1. Author's Warranty

By signing this Consent, the Author warrants all of the following: The Work has not been published before in any form except as a preprint, unless explicitly noted as a footnote to the title. The Work is not being concurrently submitted to and is not under consideration by another publisher. The names listed above as authors appear in the manuscript itself, no author entitled to credit has been omitted, and there are no unnamed authors. The Author has the right to make the grants made to the Publisher complete and unencumbered. The Author also warrants that the Work does not libel anyone, violate anyone's privacy or publicity rights, infringe anyone's copyright, trademark, or trade secrets, or otherwise violate anyone's statutory or common law rights.

2. Author's Rights

A1. The Author may reproduce and distribute the Work (including derivative works) in connection with the Author's teaching, technical collaborations, conference presentations, lectures, or other scholarly works and professional activities as well as to the extent the fair use provisions of the U.S. Copyright Act permit. If the copyright is granted to the Publisher, then the proper notice of the Publisher's copyright should be provided.
A2. The Author may post the final draft of the Work, as it exists immediately prior to editing and production by the Publisher, on noncommercial pre-print servers such as arXiv.org.
A3. The Author may post the final published version of the Work on the Author's personal web site and on the web server of the Author's institution, provided that proper notice of the Publisher's copyright is included and that no separate or additional fees are collected for access to or distribution of the work.

3. Publisher's Rights

Even if the Author does not transfer Copyright to the Publisher, the Author grants the Publisher the following rights in perpetuity.

P1. The Publisher has unlimited rights throughout the world to publish and distribute the final version of the Work in any form and in all media now known or hereafter discovered.
P2. The Publisher has unlimited rights throughout the world to translate the final version of the Work and exercise all rights in all media in the resulting

Notice of publication and copyright

First Published in “Stencil scaling for vector-valued PDEs on hybrid grids with applications to generalized Newtonian fluids” in SIAM Journal on Scientific Computing 42.6 (2020), published by the Society for Industrial and Applied Mathematics (SIAM).

DOI: https://doi.org/10.1137/19M1267891

SIAM J. SCI. COMPUT.
Vol. 42, No. 6, pp. B1429–B1461

STENCIL SCALING FOR VECTOR-VALUED PDES ON HYBRID GRIDS WITH APPLICATIONS TO GENERALIZED NEWTONIAN FLUIDS*

DANIEL DRZISGA†, ULRICH RÜDE‡, AND BARBARA WOHLMUTH†

Abstract. Matrix-free finite element implementations for large applications provide an attractive alternative to standard sparse matrix data formats due to the significantly reduced memory consumption. Here, we show that they are also competitive with respect to the run-time in the low-order case if combined with suitable stencil scaling techniques. We focus on variable coefficient vector-valued partial differential equations as they arise in many physical applications. The presented method is based on scaling constant reference stencils originating from a linear finite element discretization instead of evaluating the bilinear forms on the fly. This method assumes the usage of hierarchical hybrid grids, and it may be applied to vector-valued second-order elliptic partial differential equations directly or as a part of more complicated problems. We provide theoretical and experimental performance estimates showing the advantages of this new approach compared to the traditional on-the-fly integration and stored matrix approaches. In our numerical experiments, we consider two specific mathematical models, namely, linear elastostatics and incompressible Stokes flow. The final example considers a nonlinear shear-thinning generalized Newtonian fluid. For this type of nonlinearity, we present an efficient approach for computing a regularized strain rate which is then used to define the nodewise viscosity. Depending on the compute architecture, we could observe maximum speedups of 64% and 122% compared to the on-the-fly integration. The largest considered example involved solving a Stokes problem with 12288 compute cores on the state-of-the-art supercomputer SuperMUC-NG.

Key words. matrix-free, finite elements, variable coefficients, stencil scaling

AMS subject classifications. 65N30, 65N55, 65Y05, 65Y20

DOI. 10.1137/19M1267891

1. Introduction.

In this article, we study the efficiency of large-scale low-order finite element computations, and we examine which accuracy can be obtained at what cost. High performance computing is expensive, not only in terms of investments in supercomputer systems, but also in terms of operational cost. In particular, energy consumption is becoming a critical factor; see, e.g., the emerging rankings such as the GREEN500 list.[1] Therefore, it is crucial to rethink long-established computing practices and to study, quantify, and improve the efficiency of current numerical algorithms.

We primarily strive to reduce the absolute compute times. This is, of course, a viable goal in its own right, but the compute times are also directly related to the required energy for a computation. At this point, we note that while scalability is necessary for efficient large-scale parallel computing, scalability alone does not imply an efficient use of resources. In fact, inefficient codes are often found to scale better than efficient ones. Similarly, the asymptotic convergence rate of a discretization scheme is an important mathematical criterion affecting the accuracy, but ultimately

*Submitted to the journal's Computational Methods in Science and Engineering section June 13, 2019; accepted for publication (in revised form) August 27, 2020; published electronically December 1, 2020.

https://doi.org/10.1137/19M1267891

Funding: This work was partly supported by the German Research Foundation through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA), and by grant WO671/11-1.

†Institute for Numerical Mathematics (M2), Technische Universität München, 85748 Garching bei München, Germany (drzisga@ma.tum.de, wohlmuth@ma.tum.de).

‡Department of Computer Science 10, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, 91058, Germany (ulrich.ruede@fau.de).

[1] https://www.top500.org/green500/lists/2019/11/

only the error itself matters, including the constants involved. Such considerations gain additional relevance at a time when Moore's law slows down and technological progress is no longer producing computers that automatically run twice as fast with every new year. In this situation, innovation and improvements must rely increasingly on better implementations and on algorithms that are better suited for the available architectures.

Considering efficiency in this more rigorous sense, it is found that data transport is a critical factor in addition to the executed operations. Here, data transport includes not only message passing communication in a large parallel cluster but also the data transport within each node of such a cluster, i.e., from main memory to the CPU—and even within a CPU between the different layers of caches and the registers of the functional units [19]. The energy consumption for operations and data transport in a typical CPU architecture has been quantified in [1]. Additionally, it is, of course essential to exploit fine-grained concurrency in the form of multinode architectures and by the use of vectorization. In order to achieve optimal performance, we must be aware that the current speed of memory cannot keep up with the speed of processors and that most of the energy is spent on the data transfers. Therefore, an important characteristic relevant for the efficiency of numerical algorithms on modern computers is their *balance* or *floating-point intensity*, i.e., the ratio of floating-point operations (FLOPs) performed per byte of memory access [19].

Almost all traditional finite element libraries construct global stiffness matrices by looping over local elements and adding their contributions to the global matrix. Even when stored in compressed formats, these matrices require significantly more memory than storing the solution vectors. Not only the memory consumption does present a challenge, we also need to take into account the memory traffic and latency in loading the nonzero matrix indices and entries.

To improve on the memory consumption and memory access, matrix-free methods constitute a possible remedy where only the results of matrix vector products are computed without assembling and storing the whole global matrix. Different strategies exist to implement matrix-free methods, but the predominant candidate for low-order finite elements is the element-by-element approach [3, 9, 12, 15, 35], wherein local stiffness matrices are multiplied by local vectors and later added to the global solution vector. These local stiffness matrices may be either stored individually in memory—which actually requires more memory than storing the global matrix—or computed on the fly. When using high-order finite elements, the weak forms can be integrated on the fly using standard or reduced quadrature formulas [11, 24, 25, 26, 30]. This is a well-suited strategy for future architectures because of its high arithmetic intensity [27], but we present a method that can compete with matrix-based methods even in the low-order case. In [4], we presented an alternative matrix-free stencil scaling approach for accelerating low-order finite element implementations suited for scalar second-order elliptic partial differential equations (PDEs). There, it was shown that the method was able to reduce the computational cost significantly.

Here, we will expand on this idea and present a similar matrix-free approach for vector-valued second-order elliptic PDEs. The construction is based on the use of hierarchical hybrid grids (HHGs) which form the basis in the HHG [6, 7, 18] and Hybrid Tetrahedral Grids (HyTeG) [23] frameworks. These grids are constructed by starting with an initial, possibly unstructured, simplicial triangulation of a polygonal domain and refining each element multiple times uniformly in order to create a hierarchy of meshes. Ultimately, we associate to each of these meshes a piecewise linear finite element space. By exploiting the structure obtained by these uniform refinements, it is

possible to improve the performance of the finite element solver by using a stencil-based code. Vector-valued second-order elliptic PDEs arise in the modeling of elastostatics and fluid dynamics and play an important role in mathematical modeling. We show that the idea of the scalar stencil scaling cannot be applied to these equations, since for vector-valued PDEs the simple scaling results in the discretization of a different PDE if the coefficient is not constant. Thus, there is a need for a modified stencil scaling method that is also suited for matrix-free finite element implementations on HHGs. Although this vector-valued scaling is more complicated and more expensive than the scalar stencil scaling, it has the ability to reproduce the standard finite element solutions while requiring only a fraction of the time to obtain them.

The principal novelty of this paper is the presentation of an improved method for assembling stencils for vector-valued second-order elliptic PDEs suitable for matrix-free solvers on HHGs coupled with a linear finite element discretization. We provide theoretical and experimental performance comparisons which outline the advantages of the stencil scaling approach. Furthermore, we show the convergence and the run-times of this extended stencil scaling through numerical experiments. In these experiments, we consider two specific mathematical models, namely, linear elastostatics and generalized incompressible Stokes flow. In the final example, a nonlinear shear-thinning non-Newtonian example is considered, where the viscosity depends on the shear rate.

2. Model equations and discretization. The goal of this paper is to speed up matrix-free finite element implementations for solving vector-valued second-order elliptic PDEs in a domain $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, of the form

$$\begin{aligned} -\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} &= \mathbf{f} \quad \text{in } \Omega, \\ \mathbf{u} &= \mathbf{g} \quad \text{on } \Gamma_{\mathrm{D}}, \\ \boldsymbol{\sigma} \cdot \mathbf{n} &= \hat{\mathbf{t}} \quad \text{on } \Gamma_{\mathrm{N}}, \end{aligned} \tag{2.1}$$

where the stress $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\varepsilon})$ depends on the strain and additional material parameters. One particular example for $\boldsymbol{\sigma}$ that we investigate more thoroughly is the stress tensor for linear elasticity with isotropic continuous materials given by Hooke's law as $\boldsymbol{\sigma}(\boldsymbol{\varepsilon}) = 2\mu\boldsymbol{\varepsilon} + \lambda \operatorname{tr}(\boldsymbol{\varepsilon}) I$. Furthermore, generalized incompressible Stokes flow problems may also be cast in this form when adding additional constraints. In this case, the stress tensor is defined by $\boldsymbol{\sigma}(\boldsymbol{\varepsilon}) = 2\mu\boldsymbol{\varepsilon} - pI$, where an additional pressure variable p has been introduced, and the incompressibility constraint $\boldsymbol{\nabla} \cdot \mathbf{u} = 0$ in Ω is enforced. The domain boundary $\partial\Omega$ is split into two disjoint parts, the nontrivial Dirichlet boundary Γ_{D} and the Neumann boundary Γ_{N}. See Table 1 for a complete list of occurring variables and their definitions.

TABLE 1
Required symbols and their definitions.

Symbol	Definition
$\mathbf{u}$	displacement or velocity
p	pressure
$\boldsymbol{\varepsilon}$	strain: $\frac{1}{2}\left(\boldsymbol{\nabla}\mathbf{u} + \boldsymbol{\nabla}\mathbf{u}^{\top}\right)$
$\mathbf{f}$	body forces
$\mathbf{g}$	prescribed displacement or velocity
$\hat{\mathbf{t}}$	external forces
$\mathbf{n}$	outward-pointing unit-normal vector
λ	Lamé's first parameter
μ	shear modulus/dynamic viscosity

For the rest of this section, we restrict ourselves to the case of linear elastostatics, since the method may be applied to the momentum balance of the Stokes equations in the same way. This is demonstrated in the numerical results presented in subsection 5.2. For simplicity, we consider a homogeneous Dirichlet boundary Γ_{D} for the rest of this section, so $\mathbf{g} = \mathbf{0}$. The weak form of (2.1) in the case of linear elastostatics employing Hooke's law reads as follows: For a suitable space V incorporating the Dirichlet boundary conditions, find $\mathbf{u} \in V$ such that $a(\mathbf{u}, \mathbf{v}) = f(\mathbf{v}) \; \forall \, \mathbf{v} \in V$, where

$$
\begin{aligned}
a(\mathbf{u}, \mathbf{v}) &= \langle 2\mu \boldsymbol{\varepsilon}(\mathbf{u}), \boldsymbol{\varepsilon}(\mathbf{v}) \rangle_{\Omega} + \langle \lambda \boldsymbol{\nabla} \cdot \mathbf{u}, \boldsymbol{\nabla} \cdot \mathbf{v} \rangle_{\Omega}, \\
f(\mathbf{v}) &= \langle \mathbf{f}, \mathbf{v} \rangle_{\Omega} + \langle \hat{\mathbf{t}}, \mathbf{v} \rangle_{\Gamma_N}.
\end{aligned} \tag{2.2}
$$

By $\langle \cdot, \cdot \rangle_{\Omega}$ we denote the standard duality product in V.

In order to discretize the problem, we decompose the computational domain in the typical HHG manner [5, 6, 7]. Let $\mathcal{T}_H$ be a possibly unstructured simplicial triangulation of a bounded polygonal or polyhedral domain $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$. Based on this initial grid, we construct a hierarchy of $L+2$, $L \in \mathbb{N}$, grids $\mathcal{T} = \{\mathcal{T}_h, \; h = 2^0 H, \ldots, 2^{-L-1} H\}$ by successive global uniform refinement. As is standard, each of these refinements is achieved by subdividing all elements into 2^d subelements. For details of the refinement in 3D, we refer the reader to [8]. We also call $\mathcal{T}_H$ macro-triangulation and denote its elements by T, whereas the elements of $\mathcal{T}_h$ are denoted by t_h. For better readability, we drop the index h whenever its value is clear from the context. Since our data structures require at least one interior vertex per macro-element, we use the mesh $\mathcal{T}_h$ with $h = 2^{-2} H$ as the coarse grid in our multigrid solver hierarchy. Each of the $\mathcal{T}_h$ for $h \leq 2^{-2} H$ has some crucial properties we want to exploit. The element neighborhood at each vertex in the interior of a macro-element is always the same; cf. Figure 1. Provided that the coefficient in the PDE is constant, we can construct stencils as in a finite-difference scheme but with finite elements. Another important property is that the neighboring elements have some similarities. In 2D, there are always two elements attached to each stencil edge; see left-hand side of Figure 1. These two elements t and t^m are congruent and differ only by a reflection along the stencil edge. A similar structural property holds in 3D which we discuss later in subsection 2.4.

Associated with $\mathcal{T}_h$ is the space $V_h \subset V$ of piecewise linear finite elements. Let $\boldsymbol{e}_i \in \mathbb{R}^d$ be the canonical unit vector with $(\boldsymbol{e}_i)_j = \delta_{ij}$ for $1 \leq i, j \leq d$, where δ_{ij} is the Kronecker delta. Let further $\phi_i \in V_h$ and $\phi_j \in V_h$ be the scalar-valued linear nodal basis functions associated with the ith and jth mesh nodes. Denote by $\mathbf{v}_h = \sum_i \mathbf{v}^{(j)} \phi_j$ and $\mathbf{w}_h = \sum_j \mathbf{w}^{(i)} \phi_i$ linear combinations of the nodal basis function with vector-valued coefficients $\mathbf{v}^{(i)} \in \mathbb{R}^d$ and $\mathbf{w}^{(j)} \in \mathbb{R}^d$. We split the bilinear form (2.2) in terms of contributions of the bilinear form a^T restricted to each macro-element $T \in \mathcal{T}_H$, i.e.,

$$
\begin{aligned}
a(\mathbf{v}_h, \mathbf{w}_h) &= \sum_{T \in \mathcal{T}_H} a^T(\mathbf{v}_h, \mathbf{w}_h) = \sum_{T \in \mathcal{T}_H} \sum_{i,j} a^T(\mathbf{v}^{(j)} \phi_j, \mathbf{w}^{(i)} \phi_i) \\
&= \sum_{T \in \mathcal{T}_H} \sum_{i,j} \sum_{l,m=1}^{d} (\mathbf{v}^{(j)})_l (\mathbf{w}^{(i)})_m \, a^T(\phi_j \boldsymbol{e}_l, \phi_i \boldsymbol{e}_m).
\end{aligned} \tag{2.3}
$$

In order to simplify notation, we introduce the operator D in place of either differential operator, i.e., $D\mathbf{u} = \epsilon(\mathbf{u})$ or $D\mathbf{u} = \boldsymbol{\nabla} \cdot \mathbf{u}$, and the coefficient placeholder k, i.e., $k = \mu$ or $k = \lambda$. For the rest of this subsection, we restrict ourselves to the general bilinear

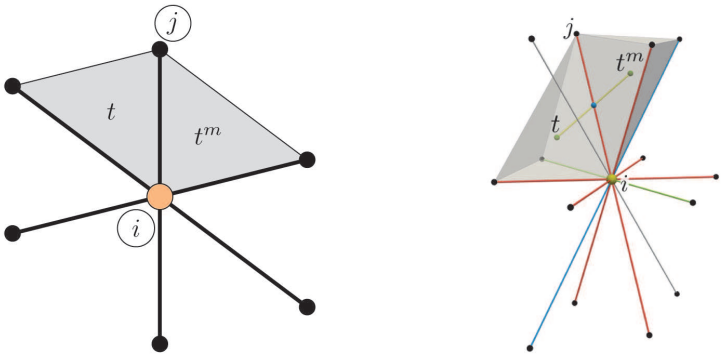


FIG. 1. *Element t and reflected element t^m along an edge between nodes i and j in the 2D case (left) and the 3D case (right).*

form

$$a(\mathbf{u},\mathbf{v}) = \langle k \cdot D(\mathbf{u}), D(\mathbf{v})\rangle_\Omega .$$

Using the standard finite element approach, this bilinear form is usually discretized in the following way. Let i_t and j_t be the local indices of an element $t \in \mathcal{T}_h$ associated with the global mesh nodes i and j. We denote by k^t the arithmetic mean over all the vertex coefficient values of an element t, i.e.,

$$\bar{k}^t = \frac{\sum_{p=1}^{d+1} k(\boldsymbol{x}_p^t)}{d+1}, \tag{2.4}$$

where $\boldsymbol{x}_p^t$ are the vertex coordinates of the local element t. Let ϕ_i^t and ϕ_j^t be the local scalar-valued linear nodal basis functions associated with the local vertices i_t and j_t. Because the derivatives of the linear basis functions are constant, employing (2.4) as a quadrature rule to approximate the bilinear form (2.3) yields

$$a_h(\mathbf{v}_h,\mathbf{w}_h) = \sum_{T\in\mathcal{T}_H}\sum_{i,j}\sum_{l,m=1}^{d} (\mathbf{v}^{(j)})_l \cdot (\mathbf{w}^{(i)})_m \sum_{t\in\mathcal{T}_h^{i,j;T}} \bar{k}^t \int_t (D(\phi_j^t \boldsymbol{e}_l), D(\phi_i^t \boldsymbol{e}_m))\,\mathrm{d}x, \tag{2.5}$$

where $\mathcal{T}_h^{i,j;T}$ is the set of all elements within a macro-element T adjacent to the edge through i and j. Note that the number of elements in this set is small due to the HHG structure. In 2D, there are only two elements adjacent to an interior edge, and in 3D there are two types of interior edges: one type with four elements and another with six elements adjacent to it. Please refer to subsection 2.4 for more details. Throughout the paper, we denote the bilinear form $a_h(\cdot,\cdot)$ defined in (2.5) by *nodal integration*. We note that the choice of $\bar{k}^t$ is natural in the case when the coefficient function is stored at the nodes. Alternative definitions, such as the evaluation of the coefficient function at the center of an element, are also suitable, and they are preferred if the coefficient function must be evaluated analytically.

With these considerations in mind, we define the reference stencil $\hat{S}_{ij}^T \in (\mathbb{R}^{d\times d})$

for $T \in \mathcal{T}_H$ as a $d \times d$ matrix for every pair of mesh nodes i and j by

$$\left(\hat{S}_{ij}^T\right)_{lm} = \int_T (D(\phi_j \boldsymbol{e}_l), D(\phi_i \boldsymbol{e}_m))\,\mathrm{d}x.$$

See Figure 1 for an illustration of stencils in 2D and 3D in the interior of a macro-element. Each edge in the stencil plots corresponds to a neighbor j of a central entry i in the mesh $\mathcal{T}_h$. Recall that, as described in the construction of HHGs, the structure in the interior of a single macro-element is always the same. It is also interesting to note that in [4] each stencil weight consists of a scalar real value, but here, each stencil weight consists of an $\mathbb{R}^{d\times d}$ matrix corresponding to the interaction between the dimensional components.

If $k \equiv 1$, the integrals in (2.5) may be replaced by the corresponding reference stencils, and the bilinear forms (2.5) and (2.3) are equal on the discrete space $V_h \times V_h$. The following lemma presents a decomposition of the bilinear form (2.5), which is better suited for matrix-free methods because it has a lower operational count while requiring a comparable amount of memory traffic. This decomposition is very similar to a decomposition of the displacement or velocity field into a symmetric strain rate part and an antisymmetric rotational part.

LEMMA 2.1. *Under the assumption that the coefficient k is affine linear on each local element patch $\omega_{i,j;T} = \bigcup_{t\in\mathcal{T}_h^{i,j;T}} \bar{t}$, the bilinear form* (2.5) *may be decomposed into a symmetric part with a scaled reference stencil and a remaining antisymmetric correction term R,*

$$\hat{a}_h(\mathbf{v}_h, \mathbf{w}_h) = \sum_{T\in\mathcal{T}_H} \sum_{i,j} \sum_{l,m=1}^{d} \left(\hat{k}_{ij}^T \cdot (\hat{S}_{ij}^T)_{lm} + \left(R^T(k)_{ij}\right)_{lm}\right) \cdot (\mathbf{v}^{(j)})_l (\mathbf{w}^{(i)})_m, \tag{2.6}$$

where $\hat{k}_{ij}^T$ is specified as in (2.8).

Proof. Let the local stiffness tensor of a local element t be given by

$$\left(a_{ij}^t\right)_{lm} = \int_t \left(D\left(\phi_j^t \boldsymbol{e}_l\right), D\left(\phi_i^t \boldsymbol{e}_m\right)\right)\,\mathrm{d}x.$$

In the following, we assume that $i \neq j$ and that k is linear on the patch $\omega_{i,j;T}$. Additionally, we introduce the symmetric part $a_{ij}^{s;t}$ and the antisymmetric part $a_{ij}^{a;t}$ of a_{ij}^t defined by

$$a_{ij}^{s;t} = \frac{1}{2}\left(a_{ij}^t + \left(a_{ij}^t\right)^\top\right) \quad \text{and} \quad a_{ij}^{a;t} = \frac{1}{2}\left(a_{ij}^t - \left(a_{ij}^t\right)^\top\right). \tag{2.7}$$

Due to our mesh structure, for each t in the interior of T, there exists a reflected element t^m; cf. Figure 1. Exploiting the fact that $\nabla\phi_i^t = -\nabla\phi_j^{t_m}$, one can show that the local stiffness tensors of these elements are related in the following way:

$$a_{ij}^{s;t} = a_{ij}^{s;t^m} \quad \text{and} \quad a_{ij}^{a;t} = -a_{ij}^{a;t^m}.$$

Before proceeding, we define the arithmetic mean of the coefficients on the patch $\omega_{i,j;T}$ as

$$\hat{k}_{ij}^T = \frac{1}{|\mathcal{T}_h^{i,j;T}|} \sum_{t\in\mathcal{T}_h^{i,j;T}} \bar{k}^t, \tag{2.8}$$

where $|\mathcal{T}_h^{i,j;T}|$ stands for the number of elements in $\mathcal{T}_h^{i,j;T}$. Using these properties, we can rewrite the last sum in (2.5) as

$$\begin{aligned}\sum_{t\in\mathcal{T}_h^{i,j;T}} \bar{k}^t (a_{ij}^t)_{lm} &= \frac{1}{2}\sum_{t\in\mathcal{T}_h^{i,j;T}} \bar{k}^t (a_{ij}^t)_{lm} + \bar{k}^{t^m} (a_{ij}^{t^m})_{lm} \\ &= \frac{1}{2}\sum_{t\in\mathcal{T}_h^{i,j;T}} \bar{k}^t (a_{ij}^{s;t})_{lm} + \bar{k}^t (a_{ij}^{a;t})_{lm} + \bar{k}^{t^m} (a_{ij}^{s;t^m})_{lm} + \bar{k}^{t^m} (a_{ij}^{a;t^m})_{lm} \\ &= \frac{1}{2}\sum_{t\in\mathcal{T}_h^{i,j;T}} (\bar{k}^t + \bar{k}^{t^m})(a_{ij}^{s;t})_{lm} + (\bar{k}^t - \bar{k}^{t^m})(a_{ij}^{a;t})_{lm} \\ &= \hat{k}_{ij}^T \cdot \hat{S}_{ij} + \frac{1}{2}\sum_{t\in\mathcal{T}_h^{i,j;T}} (\bar{k}^t - \bar{k}^{t^m})(a_{ij}^{a;t})_{lm}.\end{aligned}$$

In the last step, we exploited the facts that for an affine linear k, we have $\bar{k}^t + \bar{k}^{t^m} = 2k(\frac{x_i+x_j}{2}) = 2\hat{k}_{ij}^T$, and that

$$\sum_{t\in\mathcal{T}_h^{i,j;T}} (a_{ij}^{s;t})_{lm} = (\hat{S}_{ij}^T)_{lm}.$$

With these considerations in mind, we define the tensor $R^T(k)$ for each i and j by

$$\left(R^T(k)\right)_{ij} = \frac{1}{2}\sum_{t\in\mathcal{T}_h^{i,j;T}} (\bar{k}^t - \bar{k}^{t^m}) a_{ij}^{a;t}. \tag{2.9}$$

In the case when $i = j$, we set the correction term $\left(R^T(k)\right)_{ii}$ to zero and the scaling term $\hat{k}_{ii}^T$ to 1 and redefine the central stencil entry as

$$S_{ii}^T = -\sum_{j\neq i} \hat{k}_{ij}^T \cdot \hat{S}_{ij}^T + \left(R^T(k)\right)_{ij}.$$

This zero-row sum property ensures that translational body motions lie in the kernel of the discrete operator induced by (2.6). ∎

In addition to the bilinear form (2.6), we define the following form where the correction term R has been omitted:

$$\tilde{a}_h(\mathbf{v}_h, \mathbf{w}_h) = \sum_{T\in\mathcal{T}_H}\sum_{i,j}\sum_{l,m=1}^{d} \hat{k}_{ij}^T \cdot (\hat{S}_{ij}^T)_{lm} (\mathbf{v}^{(j)})_l (\mathbf{w}^{(i)})_m. \tag{2.10}$$

Henceforth, we refer to the bilinear form (2.6) as *physical scaling* and to the form (2.10) as *unphysical scaling*.

2.1. Interpretation of the unphysical scaling in 2D. It may be shown that the bilinear form corresponding to the unphysical form belongs to a different PDE. We illustrate this for the differential operator $-\boldsymbol{\nabla}\cdot(k\,\boldsymbol{\varepsilon}(\mathbf{u}))$ in 2D. Particularly, in 2D, a straightforward computation shows, for a differentiable coefficient k and smooth $\mathbf{u}$, that the identity

$$\boldsymbol{\nabla}\cdot(k\,(\boldsymbol{\nabla}\cdot\mathbf{u})I) = \boldsymbol{\nabla}\cdot(k\,\boldsymbol{\nabla}\mathbf{u}^\top) + \begin{pmatrix} (u_2)_{,y} & -(u_2)_{,x} \\ -(u_1)_{,y} & (u_1)_{,x}\end{pmatrix}\boldsymbol{\nabla}k = \boldsymbol{\nabla}\cdot(k\,\boldsymbol{\nabla}\mathbf{u}^\top) + (\boldsymbol{\nabla}\times O\mathbf{u})\boldsymbol{\nabla}k$$

holds true, with the identity matrix I and

$$\boldsymbol{\nabla} \times \mathbf{w} = \begin{pmatrix} (w_1)_{,y} & -(w_1)_{,x} \\ (w_2)_{,y} & -(w_2)_{,x} \end{pmatrix} \text{ and } O\mathbf{w} = \begin{pmatrix} w_2 \\ -w_1 \end{pmatrix}.$$

Using the above identity, we find

$$\begin{aligned} -\boldsymbol{\nabla} \cdot (k\, \boldsymbol{\varepsilon}(\mathbf{u})) &= -\frac{1}{2}\boldsymbol{\nabla} \cdot (k\, \boldsymbol{\nabla}\mathbf{u}) - \frac{1}{4}\boldsymbol{\nabla} \cdot (k\, \boldsymbol{\nabla}\mathbf{u}^\top) - \frac{1}{4}\boldsymbol{\nabla} \cdot (k\, \boldsymbol{\nabla}\mathbf{u}^\top) \\ &= \underbrace{-\frac{1}{2}\boldsymbol{\nabla} \cdot (k\, \boldsymbol{\nabla}\mathbf{u}) - \frac{1}{4}\boldsymbol{\nabla} \cdot (k\, (\boldsymbol{\nabla} \cdot \mathbf{u})I) - \frac{1}{4}\boldsymbol{\nabla} \cdot (k\, \boldsymbol{\nabla}\mathbf{u}^\top)}_{-\nabla \cdot \mathsf{A}(\mathbf{u})} + \underbrace{\frac{1}{4}(\boldsymbol{\nabla} \times O\mathbf{u})\boldsymbol{\nabla}k}_{\mathsf{B}(\mathbf{u})} \\ &= -\boldsymbol{\nabla} \cdot \mathsf{A}(\mathbf{u}) + \mathsf{B}(\mathbf{u}). \end{aligned}$$

The first term $\boldsymbol{\nabla} \cdot \mathsf{A}(\mathbf{u})$ corresponds to a second-order differential operator for which we have

$$\mathsf{A}\begin{pmatrix} u \\ 0 \end{pmatrix} : \boldsymbol{\nabla}\begin{pmatrix} 0 \\ v \end{pmatrix} = \mathsf{A}\begin{pmatrix} 0 \\ u \end{pmatrix} : \boldsymbol{\nabla}\begin{pmatrix} v \\ 0 \end{pmatrix}.$$

This property guarantees that the antisymmetric part of the associated local stencil term is zero, and thus it can be simply scaled. The second term $\mathsf{B}(\mathbf{u})$ is obviously equal to zero if k is constant; otherwise, it corresponds to a first-order differential operator. Here we find

$$\begin{pmatrix} v & 0 \end{pmatrix} \mathsf{B}\begin{pmatrix} 0 \\ u \end{pmatrix} = -\begin{pmatrix} 0 & v \end{pmatrix} \mathsf{B}\begin{pmatrix} u \\ 0 \end{pmatrix} \text{ and } \begin{pmatrix} v & 0 \end{pmatrix} \mathsf{B}\begin{pmatrix} u \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & v \end{pmatrix} \mathsf{B}\begin{pmatrix} 0 \\ u \end{pmatrix} = 0,$$

which yields that the symmetric part of the associated local stencil term is zero. A more detailed comparison shows that it corresponds to the antisymmetric correction term R. Therefore, omitting the correction term R in (2.6) with $D = \boldsymbol{\varepsilon}$ does not result in a discretization of the PDE $-\boldsymbol{\nabla} \cdot (k\, \boldsymbol{\varepsilon}(\mathbf{u})) = \mathbf{f}$, but of $-\boldsymbol{\nabla} \cdot (\mathsf{A}(\mathbf{u})) = \mathbf{f}$, for a general k. Only in the case when k is constant in Ω does the correction term vanish and is the original PDE recovered. We note that the splitting of the differential operator in the terms associated with the operators A and B corresponds exactly to the splitting of the stencil entries in symmetric and antisymmetric parts. Similar considerations can be worked out in 3D.

2.2. Stencil-based matrix-free methods. These newly introduced bilinear forms are very well suited for stencil-based matrix-free methods on HHGs, since the reference stencil and the correction terms are always the same for a single macro-element, and only the scaling terms depending on the coefficient need to be recomputed. In section 4, we present a short analysis of the computational cost of the standard approach by nodal integration compared to the scaling-based approaches.

In Lemma 2.1, we assume that the coefficient k is affine linear on each local patch of elements adjacent to an edge. Therefore, if k is a global affine linear function, both bilinear forms $a_h(\cdot,\cdot)$ and $\hat{a}_h(\cdot,\cdot)$ are equal. Since we use linear finite elements, optimal convergence rates may still be observed if a linear local interpolant of a nonlinear k is used. The unphysical scaling $\tilde{a}_h(\cdot,\cdot)$, however, is only equal to the other bilinear forms when the coefficient k is constant on the whole domain.

Remark 2.2. The definition in (2.8) may be replaced by $\hat{k}_{ij}^T = \frac{1}{2}\left(k(\boldsymbol{x}_i) + k(\boldsymbol{x}_j)\right)$. This approach requires fewer FLOPs, but numerical experiments suggest that using

(2.8) yields better accuracy while not having a huge impact on performance. The memory traffic for either approach is the same because the coefficients need to be loaded from memory anyway in order to compute the correction term. This assumes that the *layer condition* [19] is satisfied when traversing the HHG data structures, so the values of k need to be read from memory only once.

In practice, this scaling of the reference stencil is only done in the interior of macro-elements where, asymptotically, most computations are performed in order to evaluate the bilinear form. The physically scaled form $\hat{a}_h(\cdot,\cdot)$ is thus redefined as

$$\hat{a}_h(\phi_j \boldsymbol{e}_l, \phi_i \boldsymbol{e}_m) = \begin{cases} a_h(\phi_j \boldsymbol{e}_l, \phi_i \boldsymbol{e}_m) & \text{if } x_i \in \partial T \text{ and } x_j \in \partial T \text{ of at least one } T \in \mathcal{T}_H, \\ \hat{a}_h(\phi_j \boldsymbol{e}_l, \phi_i \boldsymbol{e}_m) & \text{otherwise.} \end{cases}$$

This definition enforces global symmetry of the matrix but requires taking into account special boundary cases when iterating over the interior of macro-elements. In practice, we therefore employ an alternative definition where we use the standard bilinear form only if $x_i \in \partial T$ of at least one $T \in \mathcal{T}_H$. This loss of global symmetry across macro-element interfaces may cause problems for iterative solvers relying on symmetric matrices. However, this symmetry loss can be regarded as higher-order perturbation and in the numerical experiments provided in section 5, no degradation of the convergence of the employed iterative solvers could be observed.

In the following two subsections, we show how to efficiently precompute most parts of the correction term (2.9) in order for them to be suitable for stencil-based codes. Since the correction term depends on the space dimension, we derive it separately for 2D and 3D.

Remark 2.3. The presented idea of scaling the reference stencils and thus obtaining an accurate enough approximation of the stiffness matrix entries is only valid for linear finite elements. However, for higher-order discretizations it is still possible to exploit the HHG structure. There, we assume that the reference basis functions, their gradients, and the coefficient are evaluated and stored at the quadrature points. Using these values, the stencils are assembled similarly to those in [24]. Let $\mathcal{Q}_t$ be the set of quadrature points in an element $t \in \mathcal{T}_h$. Due to the HHG structure, for each $q_t \in \mathcal{Q}_t$ there exists a reflected quadrature point $q_{t^m} \in \mathcal{Q}_{t^m}$. For each pair of reflected elements t and t^m attached to an edge between two nodes x_i and x_j, we need to store the matrices $(E_{q_t})_{lm} = (D(\phi_j(x_{q_t})\boldsymbol{e}_l), D(\phi_i(x_{q_t})\boldsymbol{e}_m))$ for $q_t \in \mathcal{Q}_t$. Let $\mathcal{T}_h^{i,j;T;m}$ be the set of half of the elements within a macro-element T adjacent to the edge through i and j which have a unique corresponding reflected element which is not in the set. Additionally, let ω_{q_t} be the quadrature weights corresponding to the quadrature points in $\mathcal{Q}_t$. The stencil S_{ij}^T may then by computed via

$$S_{ij}^T = \sum_{t \in \mathcal{T}_h^{i,j;T;m}} \sum_{q_t \in \mathcal{Q}_t} |t| \omega_{q_t} \left(k_{q_t} + k_{q_{t^m}}\right) E_{q_t}.$$

2.3. Correction term in 2D. In this subsection, we consider the correction term (2.9) in the case of two dimensions, i.e., $d = 2$, and present a closed form of its values. The antisymmetric part $a_{ij}^{a;t}$ of a_{ij}^t defined through (2.7) is determined by a single variable $\gamma^{(i,j);t}$ and is of the following form:

$$a_{ij}^{a;t} = \begin{pmatrix} 0 & -\gamma^{(i,j);t} \\ \gamma^{(i,j);t} & 0 \end{pmatrix}.$$

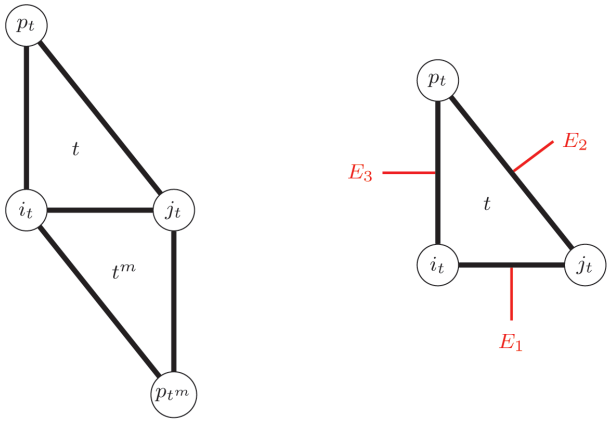


FIG. 2. *Local indices of an element t and its corresponding reflected element t^m (left). An element t with the three edges (right).*

Let $\mathbf{n}(x) = (n_1(x), n_2(x))^\top$ be the outward-pointing unit-normal of an element $t \in \mathcal{T}_h$ for $x \in \partial t$, and let $\boldsymbol{\tau}(x) = (-n_2(x), n_1(x))^\top$ be the corresponding tangential vector. In the following, we assume that the differential operator D is given by $D\mathbf{u} = \boldsymbol{\varepsilon}(\mathbf{u})$. Additionally, in the constant coefficient reference case, we have $k = 1$. Doing the same computations with $D\mathbf{u} = \boldsymbol{\nabla} \cdot \mathbf{u}$ results in the same values but now just with a flipped sign. The value $\gamma^{(i,j);t}$ can be rewritten as

$$\gamma^{(i,j);t} = \int_t \boldsymbol{\varepsilon}(\phi_j \boldsymbol{e}_1) : \boldsymbol{\varepsilon}(\phi_i \boldsymbol{e}_2) - \boldsymbol{\varepsilon}(\phi_j \boldsymbol{e}_2) : \boldsymbol{\varepsilon}(\phi_i \boldsymbol{e}_1) \, \mathrm{d}x = \frac{1}{2} \int_t \phi_{j,y}\phi_{i,x} - \phi_{j,x}\phi_{i,y} \, \mathrm{d}x$$

$$= \frac{1}{2} \int_{\partial t} \phi_i \left(\phi_{j,y} n_1 - \phi_{j,x} n_2\right) \mathrm{d}s = \frac{1}{2} \int_{\partial t} \phi_i \left(\phi_{j,y} \tau_2 + \phi_{j,x} \tau_1\right) \mathrm{d}s = \frac{1}{2} \int_{\partial t} \phi_i \boldsymbol{\nabla}\phi_j \cdot \boldsymbol{\tau} \, \mathrm{d}s.$$

We denote the three edges of an element $t \in \mathcal{T}_h$ by E_1, E_2, and E_3 as illustrated in Figure 2. Since $\phi_i = 0$ on E_2 and $\boldsymbol{\nabla}\phi_j \cdot \boldsymbol{\tau} = 0$ on E_3, the integral is reduced to

$$\gamma^{(i,j);t} = \frac{1}{2} \int_{\partial t} \phi_i \boldsymbol{\nabla}\phi_j \cdot \boldsymbol{\tau} \, \mathrm{d}s = \frac{1}{2} \int_{E_1} \phi_i \boldsymbol{\nabla}\phi_j \cdot \boldsymbol{\tau} \, \mathrm{d}s = \frac{1}{2|E_1|} \int_{E_1} \phi_i \, \mathrm{d}s = \frac{1}{4}.$$

This constant antisymmetric part $a_{ij}^{a;t}$ needs to be scaled by a difference of coefficients evaluated at the vertices. Using the notation from Figure 2, the difference is obtained by

$$\bar{k}^t - \bar{k}^{t^m} = \frac{1}{3} \left(k\left(\boldsymbol{x}_{p_t}\right) - k\left(\boldsymbol{x}_{p_{t^m}}\right)\right).$$

Finally, the correction term evaluates to

$$\left(R^T(k)\right)_{ij} = \frac{1}{12} \left(k\left(\boldsymbol{x}_{p_t}\right) - k\left(\boldsymbol{x}_{p_{t^m}}\right)\right) \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{2.11}$$

Note that in the 2D case, the correction term is independent of the geometry, and thus no extra information has to be stored in memory. As can be seen in the next subsection, this is no longer the case in 3D.

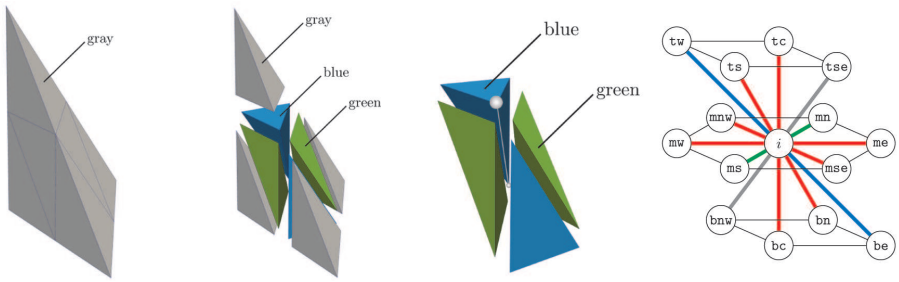


FIG. 3. *Uniform refinement of one macro-element (left) into three subclasses (second from left). Gray edge adjacent to blue and green subtetrahedra (third from left). Stencil at an interior node i of a macro-tetrahedron with off-center nodes $j \in \{\texttt{me}, \texttt{mnw}, \texttt{mn}, \texttt{ts}, \texttt{tse}, \texttt{tw}, \texttt{tc}, \texttt{bc}, \texttt{be}, \texttt{bnw}, \texttt{bn}, \texttt{ms}, \texttt{mse}, \texttt{mw}\}$ (right).*

2.4. Correction term in 3D. In the 3D case, the uniform grid refinement rule following [8] yields three subclasses of tetrahedra for each macro-element. We denote each of these classes by a color, namely gray, blue, and green; cf. left and second from left in Figure 3. We always associate the class corresponding to the macro-element to the gray color. The remaining classes are arbitrarily associated to the colors blue and green. This uniform refinement results in a stencil, which is the same for each interior node of a macro-element. The resulting stencil is illustrated on the right in Figure 3. We denote the edges adjacent to elements of classes blue and green only as edges of gray type; cf. third from left in Figure 3. The edges of green and gray types are defined similarly: An edge of blue type is adjacent only to elements of classes green and gray, whereas an edge of green type is adjacent only to elements of classes blue and gray. All other remaining edges are denoted as red-type edges.

In contrast to the 2D case, the general structure of the antisymmetric part of the local stiffness tensor $a^{a;t}$ as defined in (2.7) for an element $t \in \mathcal{T}_h^{i,j;T}$ in 3D is determined by three independent values γ, β, and δ:

$$a_{ij}^{a;t} = \begin{pmatrix} 0 & -\gamma^{(i,j);t} & -\beta^{(i,j);t} \\ \gamma^{(i,j);t} & 0 & -\delta^{(i,j);t} \\ \beta^{(i,j);t} & \delta^{(i,j);t} & 0 \end{pmatrix}. \tag{2.12}$$

Let $\mathbf{n}(x) = (n_1(x), n_2(x), n_3(x))^\top$ be the outward-pointing unit-normal of an element $t \in \mathcal{T}_h$ for $x \in \partial t$. In the case of $D\mathbf{u} = \boldsymbol{\varepsilon}(\mathbf{u})$ and $k = 1$, the nonzero components of (2.12) evaluate to

$$\begin{aligned} \beta^{(i,j);t} &= \int_t \boldsymbol{\varepsilon}(\phi_j \boldsymbol{e}_1) : \boldsymbol{\varepsilon}(\phi_i \boldsymbol{e}_3) - \boldsymbol{\varepsilon}(\phi_j \boldsymbol{e}_3) : \boldsymbol{\varepsilon}(\phi_i \boldsymbol{e}_1) \, \mathrm{d}x = \frac{1}{2} \int_t \phi_{j,z}\phi_{i,x} - \phi_{j,x}\phi_{i,z} \, \mathrm{d}x \\ &= -\frac{1}{2} \int_t \phi_{j,xz}\phi_i - \phi_{j,xz}\phi_i \, \mathrm{d}x + \frac{1}{2} \int_{\partial t} \phi_{j,z}\phi_i n_1 - \phi_{j,x}\phi_i n_3 \, \mathrm{d}s \\ &= \frac{1}{2} \int_{\partial t} \phi_i (\phi_{j,z} n_1 - \phi_{j,x} n_3) \, \mathrm{d}s. \end{aligned}$$

Similarly, the other components may be rewritten in terms of boundary integrals

$$\gamma^{(i,j);t} = \frac{1}{2} \int_{\partial t} \phi_i (\phi_{j,y} n_1 - \phi_{j,x} n_2) \, \mathrm{d}s \quad \text{and} \quad \delta^{(i,j);t} = \frac{1}{2} \int_{\partial t} \phi_i (\phi_{j,z} n_2 - \phi_{j,y} n_3) \, \mathrm{d}s.$$

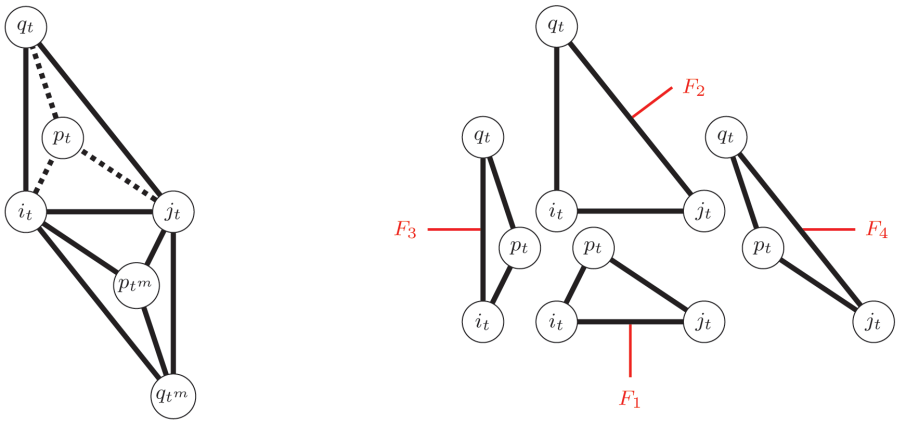


FIG. 4. *Local indices of an element t and its corresponding reflected element t^m (left). Exploded view of an element t depicting the four faces (right).*

Equivalently, as in 2D, in the case when $D\mathbf{u} = \boldsymbol{\nabla} \cdot \mathbf{u}$, the values of all three variables are the same, and only the sign is flipped.

Let i_t, j_t, p_t, and q_t be the vertex indices of a tetrahedron $t \in \mathcal{T}_h$, where i_t and j_t correspond to the global nodes i and j; see Figure 4. Additionally, the corresponding vertex coordinates of these nodes are denoted by an $\boldsymbol{x}$ with a subscript. The four faces of t are defined by the following triplets of vertices:

$$F_1 \equiv \{i_t, j_t, p_t\},\ F_2 \equiv \{i_t, j_t, q_t\},\ F_3 \equiv \{i_t, p_t, q_t\},\ \text{and } F_4 \equiv \{j_t, p_t, q_t\}.$$

Since $\phi_i = 0$ on F_4 and $(\mathbf{n} \times \boldsymbol{\nabla}\phi_j) = 0$ on F_3, applying Stokes' theorem yields

$$\begin{aligned}
\begin{pmatrix} \delta^{(i,j);t} \\ -\beta^{(i,j);t} \\ \gamma^{(i,j);t} \end{pmatrix} &= \frac{1}{2} \int_{\partial t} \phi_i \cdot (\mathbf{n} \times \boldsymbol{\nabla}\phi_j)\, \mathrm{d}s = \frac{1}{6} \sum_{f=1}^{2} \int_{F_f} \mathbf{n} \times \boldsymbol{\nabla}\phi_j \, \mathrm{d}s \\
&= \frac{1}{6} \sum_{f=1}^{2} \begin{pmatrix} \int_{F_f} \mathbf{n} \cdot (\boldsymbol{\nabla} \times \phi_j \boldsymbol{e}_1)\, \mathrm{d}s \\ \int_{F_f} \mathbf{n} \cdot (\boldsymbol{\nabla} \times \phi_j \boldsymbol{e}_2)\, \mathrm{d}s \\ \int_{F_f} \mathbf{n} \cdot (\boldsymbol{\nabla} \times \phi_j \boldsymbol{e}_3)\, \mathrm{d}s \end{pmatrix} = \frac{1}{6} \sum_{f=1}^{2} \begin{pmatrix} \int_{\partial F_f} \boldsymbol{\tau} \cdot (\phi_j \boldsymbol{e}_1)\, \mathrm{d}s \\ \int_{\partial F_f} \boldsymbol{\tau} \cdot (\phi_j \boldsymbol{e}_2)\, \mathrm{d}s \\ \int_{\partial F_f} \boldsymbol{\tau} \cdot (\phi_j \boldsymbol{e}_3)\, \mathrm{d}s \end{pmatrix} \\
&= \frac{1}{12} \left(\boldsymbol{x}_{p_t} - \boldsymbol{x}_{q_t} \right),
\end{aligned}$$

where $\boldsymbol{\tau}$ is the unit tangent. The antisymmetric part of the local stiffness tensor then reduces to

$$a_{ij}^{a;t} = \frac{1}{12} \begin{pmatrix} 0 & (\boldsymbol{x}_{q_t} - \boldsymbol{x}_{p_t})_3 & (\boldsymbol{x}_{p_t} - \boldsymbol{x}_{q_t})_2 \\ (\boldsymbol{x}_{p_t} - \boldsymbol{x}_{q_t})_3 & 0 & (\boldsymbol{x}_{q_t} - \boldsymbol{x}_{p_t})_1 \\ (\boldsymbol{x}_{q_t} - \boldsymbol{x}_{p_t})_2 & (\boldsymbol{x}_{p_t} - \boldsymbol{x}_{q_t})_1 & 0 \end{pmatrix}.$$

In Figure 5, the six elements adjacent to an edge of red type are shown. Each of these tetrahedra belongs to a class which we denote by the colors gray, green, and blue, respectively. In each color class, we have eight tetrahedra adjacent to the inner mesh node i. We define $a_{ij}^{a;t}$ to be zero if elements of the same class as t are not adjacent to an edge, which is only the case at blue-, gray-, and green-type edges. We introduce

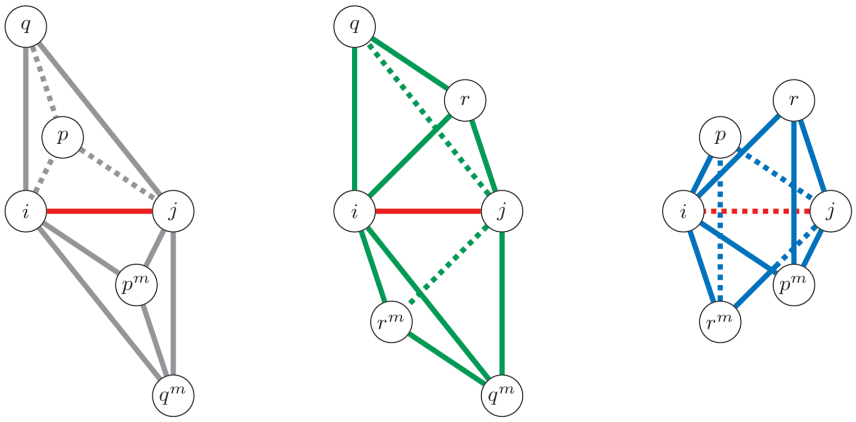


FIG. 5. *Tetrahedra adjacent to an edge of type red with local indexing of the neighboring nodes.*

a local indexing of the nodes surrounded by the edge as depicted in Figure 5 in the following considerations.

Scaling and summing over all elements adjacent to the edge through mesh nodes i and j yields

(2.13)
$$\left(R^T(k)\right)_{ij} = (\bar{k}^{t_{gray}} - \bar{k}^{t^m_{gray}})a_{ij}^{a;t_{gray}} + (\bar{k}^{t_{green}} - \bar{k}^{t^m_{green}})a_{ij}^{a;t_{green}} + (\bar{k}^{t_{blue}} - \bar{k}^{t^m_{blue}})a_{ij}^{a;t_{blue}}.$$

Since

$$\bar{k}^t - \bar{k}^{t^m} = \frac{1}{4}(k(\boldsymbol{x}_{p_t}) + k(\boldsymbol{x}_{q_t}) - k(\boldsymbol{x}_{q_{t^m}}) - k(\boldsymbol{x}_{p_{t^m}})),$$

we can rewrite (2.13) by combining terms using the index notation from Figure 5 as

$$\begin{aligned}\left(R^T(k)\right)_{ij} =& \frac{1}{4}(k(\boldsymbol{x}_p) + k(\boldsymbol{x}_q) - k(\boldsymbol{x}_{p^m}) - k(\boldsymbol{x}_{q^m}))a_{ij}^{a;t_{gray}} \\ &+ \frac{1}{4}(k(\boldsymbol{x}_q) + k(\boldsymbol{x}_r) - k(\boldsymbol{x}_{q^m}) - k(\boldsymbol{x}_{r^m}))a_{ij}^{a;t_{green}} \\ &+ \frac{1}{4}(k(\boldsymbol{x}_r) + k(\boldsymbol{x}_p) - k(\boldsymbol{x}_{r^m}) - k(\boldsymbol{x}_{p^m}))a_{ij}^{a;t_{blue}}.\end{aligned}$$

After eliminating common subexpressions, we define the three additional stencils as

$$\mathcal{S}_{ij}^{T;1} = \frac{1}{4}\left(a_{ij}^{a;t_{gray}} + a_{ij}^{a;t_{blue}}\right),\ \mathcal{S}_{ij}^{T;2} = \frac{1}{4}\left(a_{ij}^{a;t_{gray}} + a_{ij}^{a;t_{green}}\right),\ \text{and}\ \mathcal{S}_{ij}^{T;3} = \frac{1}{4}\left(a_{ij}^{a;t_{green}} + a_{ij}^{a;t_{blue}}\right).$$

To simplify the notation, we rename the following variables according to Figure 5:

$$k_{\mathcal{S}^1}^{(1)} = k(\boldsymbol{x}_p),\ k_{\mathcal{S}^1}^{(2)} = k(\boldsymbol{x}_{p^m}),\ k_{\mathcal{S}^2}^{(1)} = k(\boldsymbol{x}_q),\ k_{\mathcal{S}^2}^{(2)} = k(\boldsymbol{x}_{q^m}),\ k_{\mathcal{S}^3}^{(1)} = k(\boldsymbol{x}_r),\ \text{and}\ k_{\mathcal{S}^3}^{(2)} = k(\boldsymbol{x}_{r^m}).$$

This yields the following form of the correction term R^T in 3D:

(2.14)
$$\left(R^T(k)\right)_{ij} = \left(k_{\mathcal{S}^1}^{(1)} - k_{\mathcal{S}^1}^{(2)}\right) \cdot \mathcal{S}_{ij}^{T;1} + \left(k_{\mathcal{S}^2}^{(1)} - k_{\mathcal{S}^2}^{(2)}\right) \cdot \mathcal{S}_{ij}^{T;2} + \left(k_{\mathcal{S}^3}^{(1)} - k_{\mathcal{S}^3}^{(2)}\right) \cdot \mathcal{S}_{ij}^{T;3}.$$

Ultimately, in addition to the constant reference stencil $\hat{S}^T$, we need to store three stencils $\mathcal{S}^{T;1}$, $\mathcal{S}^{T;2}$, and $\mathcal{S}^{T;3}$ per macro-element in memory. Each of these stencils needs to be scaled appropriately to obtain a computationally cheaper approximation of the bilinear form (2.5).

3. Mapping piecewise constant coefficients to nodal values. In many physical applications, the coefficient typically depends on the strain rate $|D(\mathbf{u})|^2$ of the velocity $\mathbf{u}$ and is therefore a constant on each element when using a linear finite element discretization. Since we rely on a stencil-based implementation with coefficient values attached to the nodes in $\mathcal{T}_h$, we shall discuss efficient techniques to map piecewise constant values to nodal values. In this way, index computations that are needed to access the discrete solution can be reused to access the coefficient. Furthermore, this method is inspired by the Zienkiewicz–Zhu (ZZ) error estimator [38], where the piecewise constant gradient of a piecewise linear function is lifted to a continuous gradient. Under certain assumptions, this may improve the accuracy of the coefficient. The straightforward approach would be to find the best approximation of $|D(\mathbf{u})|^2$ in the space of piecewise linear and globally continuous functions with respect to the L^2 norm. This, however, involves solving a global linear system and thus is too costly when the coefficient changes after each iteration when solving nonlinear problems. Therefore, we refrain from a global L^2 projection and focus only on a local technique that is better suited for efficient parallel processing. One possibility is to assign to each node the volume weighted average of $|D(\mathbf{u})|^2$ over its adjacent elements. However, we present an alternative method, which we also use in our numerical experiments.

In this approach, the discrete function $\mathbf{u}$ is locally projected to an affine linear or quadratic function $\tilde{\mathbf{u}}$, and its derivative is evaluated in order to obtain an approximate value of $|D(\mathbf{u})|^2$ at a node $\boldsymbol{x}_i$. Let $\mathcal{P}_m(\omega_i)$ be the space of polynomials of order m on the patch ω_i. The jth component of $\tilde{\mathbf{u}}$ is obtained by solving the minimization problem

$$\tilde{u}_j = \operatorname*{arg\,min}_{p \in \mathcal{P}_m(\omega_i)} \sum_{i \in \mathcal{I}_i} \left(p(\boldsymbol{x}_i) - u_j(\boldsymbol{x}_i)\right)^2 \text{ for } m \in \{1, 2\}, \tag{3.1}$$

where $\mathcal{I}_i$ is the index set containing all indices of the nodal patch ω_i. Recall that in the case of a uniform refinement in 3D, this involves 15 nodes. Solving this minimization problem corresponds to solving a small least squares problem for each node $\boldsymbol{x}_i$. The approximate value of $|D(\mathbf{u})|^2$ evaluated at $\boldsymbol{x}_i$ is then given by $|D(\tilde{\mathbf{u}})(\boldsymbol{x}_i)|^2$, since by construction $D(\tilde{\mathbf{u}})$ is continuous on ω_i. As before, the coefficient k_i is obtained in a pointwise fashion according to the physical model.

In the interior of a macro-element, the quadratic and affine linear approximations are equivalent. Particularly, the quadratic minimizing polynomial p_2 of (3.1) on a patch ω_i in the interior of a macro-element may be written as $p_2(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{x}_i)^\top \mathsf{A}(\boldsymbol{x} - \boldsymbol{x}_i) + \boldsymbol{b}^\top(\boldsymbol{x} - \boldsymbol{x}_i) + c$ for some $\mathsf{A} \in \mathbb{R}^{d \times d}$, $\boldsymbol{b} \in \mathbb{R}^d$, and $c \in \mathbb{R}$. Similarly, a minimizing affine linear function on the same ω_i may be written as $p_1(\boldsymbol{x}) = \tilde{\boldsymbol{b}}^\top(\boldsymbol{x} - \boldsymbol{x}_i) + \tilde{c}$ for some $\tilde{\boldsymbol{b}} \in \mathbb{R}^d$ and $\tilde{c} \in \mathbb{R}$. Due to the symmetry of the nodes in ω_i, the quadratic and linear parts are decoupled, and it follows that $\boldsymbol{b} = \tilde{\boldsymbol{b}}$ and $c = \tilde{c}$. The derivatives of p_2 and p_1 are given by $\nabla p_2(\boldsymbol{x}) = \mathsf{A}(\boldsymbol{x} - \boldsymbol{x}_i) + \boldsymbol{b}$ and $\nabla p_1(\boldsymbol{x}) = \tilde{\boldsymbol{b}}$. Evaluating the derivatives at $\boldsymbol{x}_i$, we obtain $\nabla p_2(\boldsymbol{x}_i) = \boldsymbol{b} = \tilde{\boldsymbol{b}} = \nabla p_1(\boldsymbol{x}_i)$. Therefore, the quadratic approximation is only required on the lower-dimensional primitives, and the computationally much cheaper affine linear approximation may be used in the interior of macro-elements.

4. Computational cost analysis. Since the stencil scaling approach for vector-valued PDEs has been introduced as a means of reducing the computational cost for

matrix-free finite element implementations, we will present a concise cost analysis. Asymptotically, most of the computational work is done in the interior of macro-elements. Therefore, we restrict our performance analysis to the interior of a single macro-element. Furthermore, we ignore all performance impacts stemming from the required communication between processes and focus our analysis on multiple independent processes on a single compute node. We start with an estimation of the number of required operations to compute the residual $\mathsf{y} = \mathsf{f} - \mathsf{A}\mathsf{x}$, where the matrix A results from a discretization of the vector-valued PDEs with a single scalar coefficient. Additionally, theoretical estimates on the required memory and the memory traffic are given, which are validated by experimental measurements in subsection 5.1.1.

4.1. Number of operations. We start by counting the number of required operations to compute the residual $\mathsf{y} = \mathsf{f} - \mathsf{A}\mathsf{x}$ when using the presented method or when storing all the stencils in memory, which corresponds to storing the global matrix A. In the case of nodal integration, we assume that the local stiffness matrices (two in 2D and six in 3D) are precomputed and stored in memory. In the scaling approach, we assume that the reference stencils and the additional correction stencils are stored in memory instead. We do not precompute and store values like the average of k in (2.8) or the differences of k in (2.14). Since the stencil code is already memory bound, storing only the nodal values of k and computing the averages and differences on the fly compared to reading the precomputed values from memory increases the arithmetic intensity and reduces the required memory and the pressure on the memory buses. The required numbers of operations for the different methods are summarized in Table 2. Please note that these numbers only give estimates on the actual number of instructions performed by the processor since optimizing compilers may reorder, fuse, and vectorize FLOPs, meaning that multiple FLOPs may be performed in a single cycle. We also ignore the effect of fused multiply-add operations that are typical for most modern CPU architectures.

Let $\mathsf{y}_i \in \mathbb{R}^d$ be the target vector components at position i, let $\mathsf{x}_i \in \mathbb{R}^d$ be the input vector components at position i, and let $\mathsf{f}_i \in \mathbb{R}^d$ be the components of the right-hand-side vector at position i. Furthermore, let $S_{ij} \in \mathbb{R}^{d\times d}$ be the stencil which acts on a vector at position j in order to obtain the result at position i. The residual for all the degrees of freedom (DoFs) at position i is computed via

$$\mathsf{y}_i = \mathsf{f}_i - \sum_j S_{ij}\mathsf{x}_j. \tag{4.1}$$

Assuming that all the S_{ij} for a fixed i are already computed, the number of FLOPs for evaluating (4.1) is the same for all three approaches, as can be seen in the fifth column of Table 2. In 2D, there are seven stencils S_{ij} for a fixed i, and thus seven local matrix vector multiplications have to be performed, and the results are added for a total of 26 additions and 28 multiplications. The subtraction from the right-hand side takes two extra additions. Since there are 15 stencils in 3D, similar considerations yield that $15 \cdot 9 = 135$ multiplications need to be performed. The number of additions is made up of $15 \cdot 2 \cdot 3 = 90$ additions in the matrix-vector products, $14 \cdot 3 = 42$ additions in the sum over its results, and three additions from the subtraction of the right-hand side.

In the following, we estimate the number of required operations to compute the stencil entries S_{ij} for the matrix-free variants and begin with the physical scaling case.

4.1.1. Number of operations in the physical scaling case. Recall that in the unphysical scaling case, the stencil is defined as $S_{ij} = \hat{k}_{ij}^T \hat{S}_{ij}$ for $j \neq i$. The

TABLE 2
Operation count for residual computation.

Method	Dimension	Noncentral entries	Central entry	Residual	Total
physical scaling	2D	36 add / 36 mul	24 add / 0 mul	28 add / 28 mul	152
	3D	242 add / 220 mul	123 add / 0 mul	135 add / 135 mul	855
nodal integration	2D	33 add / 48 mul	24 add / 24 mul	28 add / 28 mul	185
	3D	754 add / 648 mul	207 add / 216 mul	135 add / 135 mul	2095
stored stencils	2D	0 add / 0 mul	0 add / 0 mul	28 add / 28 mul	56
	3D	0 add / 0 mul	0 add / 0 mul	135 add / 135 mul	270

central entry for $j = i$ is defined in a way to enforce the zero-row sum property, i.e., $S_{ii} = -\sum_{j \neq i} S_{ij}$.

Computing a stencil entry for a fixed i and j with $j \neq i$ requires computing the value of $\hat{k}_{ij}^T$ and scaling the reference stencil. The calculation of $\hat{k}_{ij}^T$ requires two multiplications and three additions in 2D, and in 3D the number of operations depends on the number of elements adjacent to the edge through the nodes i and j. Since we are interested in an upper bound only, we assume the worst case of two multiplications and seven additions. Finally, due to symmetry, the scaling requires $\frac{d(d+1)}{2}$ multiplications. These values need to be multiplied by the number of off-center stencils, which results in the numbers shown in the third column of Table 2. Computing the central entry requires 12 additions in each component, totaling 24 additions in 2D. In 3D, the number of additions per component is 41, resulting in a total of 123 additions.

In order to complete the cost consideration of the physical scaling, the cost of the correction term needs to be assessed. In 2D, the correction term (2.11) just consists of the scaled difference of two coefficient values, which results in a total of six subtractions and six multiplications. Adding the correction term to the scaled reference stencil requires 12 additional additions.

For the 3D case, recall that the physically scaled stencil is defined as

$$S_{ij} = \hat{k}_{ij}^T \cdot \hat{S}_{ij} + \left(k_{\mathcal{S}^1}^{(1)} - k_{\mathcal{S}^1}^{(2)}\right) \cdot \mathcal{S}_{ij}^{T;1} + \left(k_{\mathcal{S}^2}^{(1)} - k_{\mathcal{S}^2}^{(2)}\right) \cdot \mathcal{S}_{ij}^{T;2} + \left(k_{\mathcal{S}^3}^{(1)} - k_{\mathcal{S}^3}^{(2)}\right) \cdot \mathcal{S}_{ij}^{T;3}$$

for $j \neq i$. There, we have three correction terms with three unique nonzero entries for red edges and two correction terms with three unique nonzero entries for the remaining edges which need to be scaled. The scaling term of each correction stencil requires one addition only. This leads to $3 \cdot 3 = 9$ extra multiplications and $3 + 3 \cdot 3 = 12$ additions per red-edge stencil entry. For the edges of other colors, $2 \cdot 3 = 6$ extra multiplications and $2 + 2 \cdot 3 = 8$ additions are needed. Since there are eight red edges and six other edges per stencil, the total number of operations in the third column of Table 2 is obtained.

4.1.2. Number of operations in the nodal integration case. For the number of required operations in the nodal integration case, we recall some of the calculations from the scalar case in [4]. There, the total number of operations is reduced by eliminating common subexpressions to compute the coefficient value at the quadrature point. The number of additions required to obtain the noncentral stencil entries in the scalar case is 15 in 2D and 98 in 3D. In the vector-valued case, almost all of these numbers need to be multiplied by 4 in 2D or 9 in 3D; only the sums of the coefficients are computed once per updated node. The computation of the common subexpressions in 2D requires nine additions and 16 in 3D. Since the common subexpressions of

the coefficient only need to be computed once per node, this results in a total of $4 \cdot (15-9)+9 = 33$ in 2D and $9 \cdot (98-16)+16 = 754$ in 3D. The number of operations for the multiplications is obtained by just multiplying the number of scalar operations by 4 or 9, yielding $4 \cdot 12 = 48$ in 2D and $9 \cdot 72 = 648$ in 3D. In our case, the central entries are not computed by the computationally cheaper method of enforcing the zero row-sum property because the rigid body mode kernel is preserved in this way. These values are obtained by manually counting the number of operations for the central entries, yielding the values presented in the third and fourth rows of Table 2.

4.1.3. Number of operations in the stored stencils case. In this scenario, the whole global matrix A is stored in memory. Therefore, we assume that no costs are involved in computing the stencil entries and only the operations to compute the residual are required. Note that this scenario is the preferred one with respect to the number of operations but it consumes the most memory and it has the largest impact on memory traffic from main memory; cf. subsection 4.2.

4.1.4. Comparison of total required operations. The theoretical analysis of the required operations yields estimates of how much CPU time could be saved in the case when the memory bandwidth is not limited and when the overhead stemming from index calculations is ignored. As can be seen, the savings in FLOPs are minor in 2D, but in 3D they are quite significant. For 2D, Table 2 shows that the physical scaling requires 82% of the FLOPs that are needed by the on-the-fly nodal integration. In 3D, the physical scaling requires 41% of the FLOPs needed by the nodal integration. However, as can be seen in the measurements in subsection 5.1.1, the compiler reduces the number of theoretically estimated FLOPs. Using the values reported by the Intel Advisor,[2] we see that the physical scaling requires 44% of the FLOPs needed by the nodal integration.

4.2. Memory consumption and memory access. For the best performance, it is not only required that the number of FLOPs is small. The memory traffic from the main memory also has to be small relative to the required FLOPs. Therefore, we first give a short summary on the required number of double precision variables for a residual computation in the interior of a single macro-element in Table 3, where N is the number of scalar degrees of freedom in the interior of a single macro-element. The third column summarizes the number of variables required to store the discretized functions f, x, y, and k. The fourth column summarizes the number of variables required to store the discretized operator A. Note that only for the stored stencils approach does the memory required to store the operator grow with the mesh size. The total memory footprint is worst for the stored stencils approach. In this scenario 135 extra scalar variables must be stored, a number that would alternatively permit an extra level of refinement of the mesh when using one of the matrix-free approaches. Even if storing all stencils is cheapest in terms of FLOPs, it creates a severe restriction on the size of the problems that can be solved and leads to a very large amount of data that must be transferred from the main memory in each matrix-vector product.

In Table 4, we present estimates on the average number of bytes which need to be loaded from and stored in the main memory to compute the residual at a single mesh node in 3D. We split the estimation into two extreme cases. In the optimistic case, we assume perfect caching and that all previously loaded values stay in the fast cache levels. In the pessimistic case, we assume no caching at all and that all the data have to be loaded from the slow main memory. This analysis gives lower and

[2] https://software.intel.com/intel-advisor-xe

TABLE 3
Number of double precision variables required on a single macro-element with N scalar degrees of freedom for a residual computation.

Method	Dimension	Variables (f, x, y, k)	Operators
physical scaling	2D	$7 \cdot N$	28
	3D	$10 \cdot N$	540
nodal integration	2D	$7 \cdot N$	72
	3D	$10 \cdot N$	864
stored stencils	2D	$7 \cdot N$	$28 \cdot N$
	3D	$10 \cdot N$	$135 \cdot N$

upper bounds on the required main memory traffic, and the value observed in practice will lie somewhere between these bounds. Note that stores and loads of temporary variables required for the computation of the stencil weights are not considered in these estimates. These values only present estimates for the number of bytes that must be transferred from the main memory, but they are not necessarily proportional to the time required to load and store them. In modern architectures, data are moved in terms of cache lines that may, for example, be 64 bytes large. If numerical data are stored contiguously, successive values can be accessed more efficiently from cache lines that are already loaded. Furthermore, modern CPU micro-architectures employ prefetching that can accelerate the access to regularly strided data. A detailed analysis of such effects on the speed of numerical kernels, as presented in, e.g., [2], is beyond the scope of this article. In Table 4, we present estimated values for the bytes to be transferred in the optimistic and pessimistic scenarios.

For the matrix-free variants, the precomputed stencil values or local stiffness matrices need to be loaded. In the physical scaling case, the 15 reference stencil weights for nine block operators are required, which results in a total of 1080 bytes. Additionally, the three additional correction stencils need to be loaded, resulting in 4320 bytes. In the nodal integration case, six local stiffness matrices with 16 entries each need to be loaded for each of the nine operators, resulting in 6912 bytes. In the optimistic case, these data stay in the caches and are loaded from the main memory only in the pessimistic case.

Only one coefficient has to be loaded in the optimistic case, but in the worst case all 15 coefficients adjacent to a mesh node need to be loaded from the main memory, which results in 120 bytes per mesh node. In the stored stencils approach, for each mesh node all 15 stencil weights for nine operators need to be loaded even in the optimistic case.

Additionally, the variables f, x, and y are accessed during an iteration. In the optimistic and pessimistic cases, 24 bytes of f need to be loaded from the main memory. Additionally, because of write allocation, 24 bytes from y need to be loaded before they are stored, resulting in traffic of 48 bytes. Reusing cached values of x in the optimistic case requires loading 24 bytes, but in the pessimistic case all 15 neighboring values need to be loaded, resulting in 360 bytes.

These estimates show that with poor cache reuse, the matrix-free approaches must be expected to produce even more main memory traffic than the stored stencils approach. However, when the layer condition is satisfied for the data traversal, and the caches are used efficiently, the matrix-free methods may lead to reduced main memory traffic.

TABLE 4
Average number of bytes required to load from and store to main memory when computing the residual at a mesh node in 3D assuming the usage of 64-bit double precision floating-point variables.

Method	Optimistic	Pessimistic
physical scaling	$8 + 96 = 104$	$4320 + 120 + 432 = 4872$
nodal integration	$8 + 96 = 104$	$6912 + 120 + 432 = 7464$
stored stencils	$1080 + 96 = 1176$	$1080 + 432 = 1512$

5. Numerical results and applications. In this section, we provide numerical results to illustrate the accuracy and run-time of the new scaling approaches in comparison to the assembly by nodal integration in a matrix-free framework. Throughout this section, we denote the time-to-solution by tts, and by relative tts we denote the ratio of the time-to-solution of the stencil scaling approach with respect to the nodal integration. The numerical solutions obtained by the corresponding bilinear forms $a_h(\cdot,\cdot)$ and $\hat{a}_h(\cdot,\cdot)$ are always denoted by $\mathbf{u}_h$ and $\hat{\mathbf{u}}_h$, respectively.

We use two machines to obtain the run-time measurements presented in the following subsections. Most of the measurements are conducted on the newer SuperMUC-NG system equipped with Skylake nodes. The following values are taken from [28]. Each node has two Intel Xeon Platinum 8174 processors with a nominal clock rate of 3.1 GHz. Each processor has 24 physical cores, which results in 48 cores per node. Each core has a dedicated L1 (data) cache of size 32 kB and a dedicated L2 cache of size 1024 kB. Each of the two processors has an L3 cache of size 33 MB shared across all its cores. The total main memory of 94 GB is split into equal parts across two NUMA domains with one processor each. We use the Intel 19.0 compiler, together with the Intel 2019 MPI library, and specify the compiler flags `-O3, -march=native, -xHost`.

The second machine we use for some measurements is the older SuperMUC Phase 2 system equipped with Haswell nodes. The following values are taken from [29]. Each node has two Intel Xeon E5-2697 v3 processors with a nominal clock rate of 2.6 GHz. Each processor has 14 physical cores, which results in 28 cores per node. Each core has a dedicated L1 (data) cache of size 32 kB and a dedicated L2 cache of size 256 kB. The theoretical bandwidths are 343 GB/s and 92 GB/s, respectively. The CPUs are running in cluster-on-die mode. Thus, each node represents four NUMA domains each consisting of seven cores with a separate L3 cache of size 18 MB and a theoretical bandwidth of 39 GB/s. On top of this, each NUMA domain has 16 GB of main memory with a theoretical bandwidth of 6.7 GB/s available. On this second machine, we use the Intel 18.0 compiler, together with the Intel 2018 MPI library, and specify the compiler flags `-O3, -march=native, -xHost`. Note that the serial runs using only a single compute core are not limited to running on large machines such as SuperMUC but can also be run on usual modern desktop workstations with enough memory.

All the following experiments were implemented in the HHG framework [5, 6, 7]. If not otherwise specified, we solve the linear systems by applying geometric multigrid V-cycles directly to the system until a specified relative tolerance of the norm of the residual is obtained. The transfers from a coarser to a finer grid are performed by a matrix-free linear interpolation, and the restriction is performed by the corresponding transposed matrix-free operation. As a smoother, we employ the hybrid Gauss–Seidel method, meaning that in the interior of macro-elements standard Gauss–Seidel iterations are performed. Across the interfaces not all dependencies are updated, which results in a Jacobi-like method. On the coarse grid, we perform iterations of the diagonally preconditioned conjugate gradient method up to a fixed large relative tolerance or for a fixed number of iterations in order to avoid unnecessary oversolving.

5.1. Linear elastostatics. In this subsection, we first perform benchmarks to verify the performance models from section 4, and we consider two problems in linear elasticity. The first is a benchmark problem where we have an analytical solution at hand and can compute the discretization errors directly. In the second, a more relevant problem is investigated, where an external force is applied to a metal foam.

5.1.1. Memory traffic and roofline analysis. In subsection 4.2, we presented theoretical estimates on the number of FLOPs and the memory accesses required to compute the residual of a linear system using different strategies to obtain the matrix entries. In this subsection, we verify these results experimentally using a specially designed benchmark, executed on a single compute node of SuperMUC-NG. With this benchmark, we compare the performance of the methods analyzed in subsection 4.2, i.e., physical stencil scaling, standard nodal integration, and stored stencils approach. The floating-point performance measurements were conducted using the Intel Advisor 2019 [21], and the memory traffic measurements were conducted by accessing the hardware performance counters using the Intel VTune Amplifier 2019 [22].

The benchmark computes the residual $y = f - Ax$ for a vector-valued operator A in 3D with the same sparsity pattern as the discretized linear elasticity operator. As in the theoretical analysis, we only consider the DoFs in the interiors of macro-tetrahedra. The residual computation is iterated 500 times in order to obtain an averaged value, reducing errors stemming from small fluctuations in the run-time. The benchmark is executed using 48 MPI ranks, pinned to the 48 physical cores of a single node. This is essential to avoid optimistic bandwidth values when only a single core accesses the memory. Measurements with the Intel Advisor and VTune Amplifier are carried out solely on rank 0. Moreover, all measurements are restricted to the innermost update loop, i.e., where the actual nodal updates take place. This does not influence the results since the outer loops are identical in all variants. Note that in each update, the DoFs corresponding to a single mesh node are updated all at once, i.e., three DoFs per update. We choose $L = 5$ as the refinement level, which yields $1.09 \cdot 10^6$ DoFs per macro-element. The computation involves three vector-valued variables x, y, and f where each of them requires about 8.4 MiB of storage per macro-tetrahedron. In addition to this, the scalar-valued coefficient k requires about 2.8 MiB of storage.

We assign three macro-elements to each MPI rank, which is the maximum possible for the stored stencils approach on SuperMUC-NG. In practice, the memory limit of a compute node would be reached even faster, since all the lower-dimensional primitives, the multigrid hierarchy, and the communication buffers require extra memory. Using these settings, each innermost loop is executed 500,062,500 times per MPI rank.

In Figure 6, we summarize the recorded performance results of the three approaches. In the leftmost plot of Figure 6, the FLOPs per update are shown, which are close to the theoretically estimated values from Table 2. The second-from-left plot presents the total number of transferred bytes from the main memory per update. The total memory consumption is shown in the third-from-left plot. The stored stencils approach requires almost 15 times more memory than the matrix-free approaches. At first sight, the stored stencils approach appears to be the most attractive with respect to the required operations when enough main memory is available. However, the rightmost plot shows that the physical scaling approach has a slightly lower time per update than the stored stencils approach. This is due to the large amount of data which needs to be transferred from the main memory in each update; cf. second-from-left plot in Figure 6. This means that, in fact, the caches are more efficiently used in the matrix-free approaches. Note that these measured values are also close to the

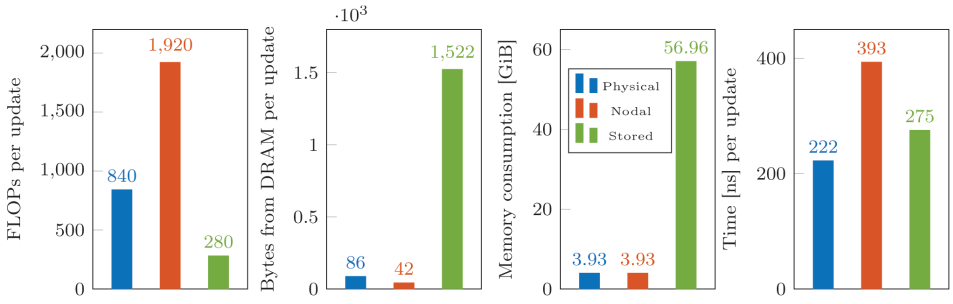


FIG. 6. *Bar plots for comparing the performance and memory consumption of the nodal integration, physical stencil scaling, and stored stencils approaches on SuperMUC-NG. Three macro-tetrahedra are assigned to each MPI rank.*

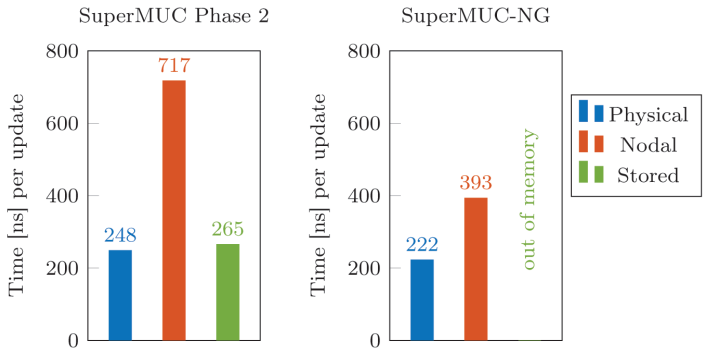


FIG. 7. *Time per update on different machines with four macro-tetrahedra attached to each MPI rank. Left: SuperMUC Phase* 2. *Right: SuperMUC-NG.*

theoretically estimated values reported in Table 4.

The same benchmark was conducted on SuperMUC Phase 2 but with four macro-tetrahedra assigned to each of the 28 MPI ranks. This was possible, since this machine has more memory available per core than SuperMUC-NG. In Figure 7, we contrast the time per update on SuperMUC Phase 2 with the time per update on SuperMUC-NG using equal problem sizes per MPI rank. Note that both of the matrix-free methods worked on SuperMUC-NG, but the stored stencil approach required too much memory. Furthermore, the time per update of the physical scaling did not improve much, but the time per update of the nodal on-the-fly integration was reduced by about 45%. This is due to the increased clock rate of SuperMUC-NG compared to SuperMUC Phase 2 and the larger arithmetic intensity of the on-the-fly integration.

In order to further visualize these results, we present a roofline analysis in Figure 8 conducted on SuperMUC-NG; see [20, 36]. The abscissa shows the arithmetic intensity, i.e., the number of FLOPs divided by the number of bytes loaded and stored in the innermost loop. The ordinate gives the measured performance as FLOPs performed per second. For reference, we added measured saturated memory bandwidth rooflines as reported by the Intel Advisor. Obviously, these measured values are smaller than the theoretically optimal ones given in the hardware specifications. The maximum performance for double precision vectorized fused multiply-add operations is reported

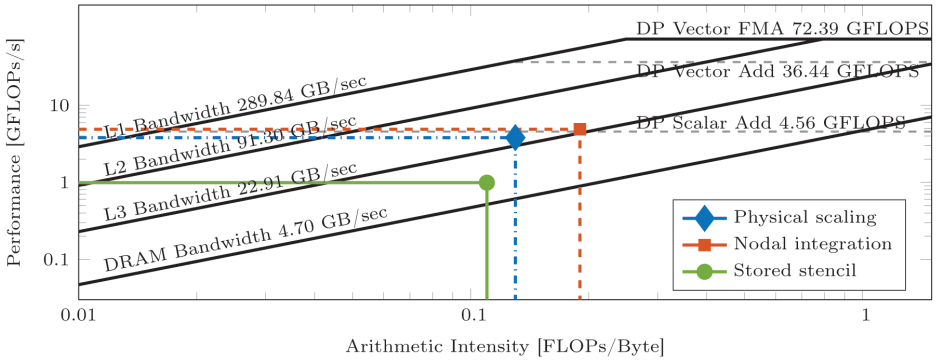


FIG. 8. *Roofline analysis of the residual computation using nodal integration, physical stencil scaling, and stored stencils approaches.*

by the Intel Advisor tool as 72.39 GFLOPs/s. From the roofline analysis one can see that the nodal integration yields the best performance with respect to FLOPs per second but is still slower in practice, since it requires more than twice the number of operations compared to the physical scaling. The physical scaling has a smaller arithmetic intensity and a slightly worse performance in GFLOPs/s, while the stored stencils approach has the lowest arithmetic intensity with the worst performance. The compiler could not autovectorize the physical scaling and stored stencils kernels. In the nodal integration kernel, however, one of the fixed-length inner loops could be autovectorized. This explains why the performance of the nodal integration is slightly better than the double precision scalar add performance. Of course, the roofline analysis constitutes only a first quantitative evaluation of the performance. Other performance models, such as the execution-cache-memory performance model [34], can give deeper insight.

Remark 5.1. As can be seen in Figure 8, the physical scaling approach reaches about 5.25% of the peak performance based on double precision vector fused multiply-add instructions. However, considering other rooflines, the physical scaling almost reaches the double precision scalar add performance and reaches about 10.43% of the double precision vector add performance. Further performance optimizations are difficult because of the less than ideal mix of multiplies and adds and the challenging vectorization due to the index calculations in tetrahedral elements. These investigations and performance optimizations are beyond the scope of this paper but are part of future work and ongoing development of software structures in HyTeG [23].

5.1.2. Linear elastostatics benchmark problem. As a first benchmark problem, we consider a compressible linear elasticity problem on the unit cube $\Omega = (0,1)^3$ modeled by (2.1) with $\Gamma_D = \partial\Omega$ and $\Gamma_N = \emptyset$. The material of the block is assumed to be isotropic and heterogeneous with a varying elastic modulus E but constant Poisson's ration ν. In this scenario, the stress tensor $\boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon} + \lambda \operatorname{tr}(\boldsymbol{\varepsilon})I$ is given by Hooke's law, and the Lamé constants μ and λ are

$$\mu(E) = \frac{E}{2(1+\nu)} \quad \text{and} \quad \lambda(E) = \frac{\nu E}{(1+\nu)(1-2\nu)}.$$

Since the stress tensor $\boldsymbol{\sigma}$ depends linearly on E, we factor it out and rewrite the stress tensor such that it depends only on the single spatially variable coefficient E, i.e.,

$$\boldsymbol{\sigma} = E(x,y,z) \cdot \left(\frac{1}{(1+\nu)} \boldsymbol{\varepsilon} + \frac{\nu}{(1+\nu)(1-2\nu)} \mathrm{tr}(\boldsymbol{\varepsilon}) I \right).$$

The associated bilinear form in the *constant case* $E = 1$ is thus given by a linear combination of the discussed forms, yielding

$$a^{E=1}(\mathbf{u},\mathbf{v}) = \frac{1}{(1+\nu)} \left\langle \boldsymbol{\varepsilon}(\mathbf{u}), \boldsymbol{\varepsilon}(\mathbf{v}) \right\rangle_\Omega + \frac{\nu}{(1+\nu)(1-2\nu)} \left\langle \boldsymbol{\nabla} \cdot \mathbf{u}, \boldsymbol{\nabla} \cdot \mathbf{v} \right\rangle_\Omega . \tag{5.1}$$

The scaling is then performed on the bilinear form (5.1) with E as the varying scalar coefficient. In the following, we perform a quantitative comparison of the three approaches by investigating their accuracy and run-time. For this purpose, we let $\mathbf{u}^*$ be a manufactured solution and set the right-hand side $\mathbf{f}$ of (2.1) accordingly to $\mathbf{f} = -\boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(\mathbf{u}^*)$. The Dirichlet boundary condition is set to $\mathbf{g} = \mathbf{u}^*\big|_{\partial\Omega}$. This allows for a direct computation of errors and a quantitative study on accuracy of the different methods. By $\mathcal{I}_h$, we denote the interpolation operator of a function on the mesh $\mathcal{T}_h$ and by $\|\cdot\|_2$ the discrete L^2 norm defined as

$$\|\mathbf{u}\|_2 = \left(h^3 \sum_{i \in \mathcal{N}_h} \|\mathbf{u}(\mathbf{x}_i)\|_2^2 \right)^{\frac{1}{2}},$$

where $\mathcal{N}_h$ is the set of all vertices in the mesh $\mathcal{T}_h$. As material parameters, we choose the Poisson's ratio of aluminum, i.e., $\nu = 0.34$, and a Young's modulus of the following: form

$$E(x,y,z) = \cos(m\,\pi\,x\,y\,z) + 2,\ m \in \{1,2,3,8\}.$$

The manufactured solution $\mathbf{u}^*$ is chosen as

$$\mathbf{u}^*(x,y,z) = \frac{1}{xyz+1} \begin{pmatrix} x^3 y + z^2 \\ x^4 y + 2z \\ 3x + y z^3 \end{pmatrix}.$$

It is important to note that the coefficient and exact solution do not lie in the ansatz spaces and therefore cannot be exactly reproduced.

We discretize the computational domain by 384 tetrahedra on the coarsest level $\ell = 0$. The finest level considered in this subsection is $L = 6$. Each system of equations is solved using a single rank on SuperMUC-NG and by employing a geometric $V(3,3)$ multigrid solver until a relative residual of 10^{-8} is obtained. As a smoother, we employ the hybrid Gauss–Seidel method, and on the coarsest level we employ a diagonally preconditioned conjugate gradient method, since the problem is symmetric and positive definite.

In Table 5, we report on the errors, convergence rates, number of required V-cycle iterations, and run-times for different refinement levels ℓ and coefficient parameters m. The error on level ℓ is defined as $\|\mathcal{I}_{h_L} \mathbf{u}^* - \mathcal{I}_{h_L} \mathbf{v}_{h_\ell}\|_2$, where $\mathbf{v}_{h_\ell}$ denotes the numerical solution obtained with one of the two matrix-free approaches, i.e., $\mathbf{u}_h$ and $\hat{\mathbf{u}}_h$. We do not consider the stored stencil approach in this comparison, since the problem sizes on the finest level are too large, and the matrices could not be stored in memory. We observe quadratic convergence in the discrete L^2 norm for the assembly through

nodal integration and the physical scaling. On the last level $L = 6$, the convergence rate is higher, since here we compare two discrete approximations on the same level. We still keep these rows in the table for completeness and in order to compare the relative tts even if the error is not directly comparable to the errors on the coarser levels. Independent of the coefficient frequency, we observe a relative tts of about 61% for $m \in \{1, 2, 3, 8\}$.

In order to emphasize that using the unphysical scaling results in a wrong solution, we computed the errors in this benchmark using the unphysical scaling (2.10) without reporting them in Table 5. In this case the errors on level $\ell = 6$ were $3.95 \cdot 10^{-3}$ for $m = 1$, $1.25 \cdot 10^{-2}$ for $m = 2$, $1.83 \cdot 10^{-2}$ for $m = 3$, and $2.32 \cdot 10^{-3}$ for $m = 8$.

For a performance comparison, we performed the same experiment on a compute node of the older SuperMUC Phase 2. The number of V-cycles and errors are the same as on SuperMUC-NG, and thus we only present the tts for both matrix-free approaches in Table 6. On this machine, we observe a relative tts of about 45% independent of the coefficient frequency. This larger speedup is due to the lower clock rate of the processor, which has already been discussed in subsection 5.1.1 and illustrated in Figure 7.

5.1.3. Linear elastostatics with external forces. In this subsection, we present an application of our scaling approach where an external force is applied to an isotropic and heterogeneous material. As before, we consider the stress tensor of Hooke's law and model the problem by (2.1), where $\partial\Omega = \Gamma_{\mathrm{D}} \cup \Gamma_{\mathrm{N}}$ and $\Omega = (0, 4) \times (0, 2) \times (0, 1)$; cf. Figure 9 (left). The Dirichlet boundary is chosen as $\Gamma_{\mathrm{D}} = \{(x, y, z) \in \Omega \mid z = 0\}$ and the Neumann boundary as $\Gamma_{\mathrm{N}} = \partial\Omega \backslash \Gamma_{\mathrm{D}}$. In this scenario, we ignore volume forces, and thus we set $\mathbf{f} = \mathbf{0}$. The material block is clamped at the bottom, and therefore we set $\mathbf{g} = \mathbf{0}$. Further, the following planar force $\hat{\mathbf{t}}$ is applied to the top plane of the foam:

$$\hat{\mathbf{t}}(x, y, z) = \begin{cases} (0, 0, -1)^\top, & z = 1 \\ (0, 0, 0)^\top & \text{else} \end{cases} \text{GPa}.$$

We assume that the material of interest is a metal foam, and thus we apply the *Gibson and Ashby model* [37, 17] which assumes the following relationship between the elastic modulus of the metal foam E_f and of the matrix E_m:

$$\frac{E_f}{E_m} \approx \phi^2 \left(\frac{\rho_f}{\rho_m} \right)^2 + (1 - \phi) \frac{\rho_f}{\rho_m}, \tag{5.2}$$

where ρ_f is the foam's density, ρ_m the matrix density, and ϕ the porosity of the foam. Again, we assume the matrix to consist of aluminum with a Poisson's ratio of $\nu = 0.34$ and the elastic modulus $E_m = 70\,\text{GPa}$. Additionally, we assume that the ratio of foam- and matrix-density is given by a radially symmetric function of the form

$$\frac{\rho_f}{\rho_m} = \frac{1}{16} x(4 - x)z(2 - y) + \frac{1}{2},$$

and $\phi = 1 - \frac{\rho_f}{\rho_m}$. The foam's elastic modulus E_f is then obtained by relationship (5.2).

We discretize the block with 3072 tetrahedra on the coarsest level $\ell = 0$. The finest level considered in this subsection is $L = 5$. Each system of equations is solved using 48 compute cores with the same multigrid solver as in the previous subsection. Since no analytical solution is available, we assume that the solution obtained with the reference bilinear form $a_h(\cdot, \cdot)$ is the true solution and compare it to the solutions

TABLE 5

Errors for the linear elastostatics benchmark problem in the discrete L^2 norm, convergence rates, number of V-cycle iterations, and time-to-solution and relative time-to-solution for nodal integration and physical scaling recorded for different refinement levels ℓ and parameters m. The measurements were conducted on SuperMUC-NG.

		Nodal integration				Physical scaling				Rel.
ℓ	DoFs	error	eoc	iter	tts [s]	error	eoc	iter	tts [s]	tts
					$m = 1$					
1	$1.52 \cdot 10^3$	$8.11 \cdot 10^{-3}$	0.00	12	0.07	$8.07 \cdot 10^{-3}$	0.00	12	0.07	0.93
2	$1.21 \cdot 10^4$	$2.12 \cdot 10^{-3}$	1.94	18	0.35	$2.10 \cdot 10^{-3}$	1.94	18	0.29	0.81
3	$9.72 \cdot 10^4$	$5.43 \cdot 10^{-4}$	1.97	25	2.97	$5.39 \cdot 10^{-4}$	1.97	25	2.03	0.68
4	$7.77 \cdot 10^5$	$1.38 \cdot 10^{-4}$	1.97	29	26.50	$1.37 \cdot 10^{-4}$	1.97	29	17.13	0.65
5	$6.22 \cdot 10^6$	$3.53 \cdot 10^{-5}$	1.97	32	230.78	$3.51 \cdot 10^{-5}$	1.97	32	143.04	0.62
6	$4.97 \cdot 10^7$	$4.71 \cdot 10^{-6}$	2.91	32	1848.95	$4.65 \cdot 10^{-6}$	2.92	32	1125.28	0.61
					$m = 2$					
1	$1.52 \cdot 10^3$	$8.26 \cdot 10^{-3}$	0.00	12	0.08	$8.23 \cdot 10^{-3}$	0.00	12	0.07	0.93
2	$1.21 \cdot 10^4$	$2.16 \cdot 10^{-3}$	1.93	19	0.37	$2.15 \cdot 10^{-3}$	1.94	19	0.30	0.81
3	$9.72 \cdot 10^4$	$5.54 \cdot 10^{-4}$	1.97	25	2.96	$5.50 \cdot 10^{-4}$	1.97	25	2.02	0.68
4	$7.77 \cdot 10^5$	$1.41 \cdot 10^{-4}$	1.97	29	26.40	$1.40 \cdot 10^{-4}$	1.97	29	16.88	0.64
5	$6.22 \cdot 10^6$	$3.59 \cdot 10^{-5}$	1.97	32	233.40	$3.57 \cdot 10^{-5}$	1.97	32	142.98	0.61
6	$4.97 \cdot 10^7$	$4.92 \cdot 10^{-6}$	2.87	33	1910.68	$4.86 \cdot 10^{-6}$	2.87	33	1173.80	0.61
					$m = 3$					
1	$1.52 \cdot 10^3$	$8.28 \cdot 10^{-3}$	0.00	12	0.07	$8.30 \cdot 10^{-3}$	0.00	12	0.07	0.95
2	$1.21 \cdot 10^4$	$2.16 \cdot 10^{-3}$	1.94	19	0.37	$2.17 \cdot 10^{-3}$	1.94	19	0.30	0.81
3	$9.72 \cdot 10^4$	$5.54 \cdot 10^{-4}$	1.97	25	2.95	$5.54 \cdot 10^{-4}$	1.97	25	2.01	0.68
4	$7.77 \cdot 10^5$	$1.41 \cdot 10^{-4}$	1.97	30	27.58	$1.41 \cdot 10^{-4}$	1.97	29	16.87	0.61
5	$6.22 \cdot 10^6$	$3.59 \cdot 10^{-5}$	1.97	32	230.73	$3.59 \cdot 10^{-5}$	1.97	32	145.07	0.63
6	$4.97 \cdot 10^7$	$4.99 \cdot 10^{-6}$	2.85	33	1902.81	$5.04 \cdot 10^{-6}$	2.84	33	1160.50	0.61
					$m = 8$					
1	$1.52 \cdot 10^3$	$8.67 \cdot 10^{-3}$	0.00	11	0.07	$9.04 \cdot 10^{-3}$	0.00	11	0.06	0.94
2	$1.21 \cdot 10^4$	$2.26 \cdot 10^{-3}$	1.94	18	0.36	$2.44 \cdot 10^{-3}$	1.89	18	0.29	0.79
3	$9.72 \cdot 10^4$	$5.75 \cdot 10^{-4}$	1.98	24	2.83	$6.36 \cdot 10^{-4}$	1.94	24	1.96	0.69
4	$7.77 \cdot 10^5$	$1.46 \cdot 10^{-4}$	1.98	28	25.69	$1.63 \cdot 10^{-4}$	1.96	28	16.46	0.64
5	$6.22 \cdot 10^6$	$3.72 \cdot 10^{-5}$	1.97	31	223.97	$4.14 \cdot 10^{-5}$	1.98	31	138.66	0.62
6	$4.97 \cdot 10^7$	$5.54 \cdot 10^{-6}$	2.75	32	1844.60	$6.90 \cdot 10^{-6}$	2.58	32	1124.61	0.61

obtained using the form $\hat{a}_h(\cdot,\cdot)$. We denote the solutions by $\mathbf{u}_h$ and $\hat{\mathbf{u}}_h$, respectively. Again, we do not consider the stored stencil approach in this comparison, since the problem sizes on the finest level are too large, and the matrices could not be stored in memory. The error on level ℓ is defined by $\|\mathcal{I}_{h_L}\mathbf{v}_{h_\ell} - \mathbf{u}_{h_L}\|$ for $\mathbf{v} \in \{\mathbf{u}, \hat{\mathbf{u}}\}$ and $\ell \leq L$. See Figure 9 (right) for an illustration of the deformed metal foam computed on level $\ell = 4$.

In Table 7, we report on the errors, convergence rates, number of V-cycle iterations, and run-times for different refinement levels ℓ. We do not observe optimal quadratic convergence in the discrete L^2 norm, even in the nodal integration case, because of the lower regularity of the problem. The solution obtained by the physical scaling, however, has the same convergence rate but with a relative tts of about 64%.

5.2. Generalized incompressible Stokes problem. In order to show that the new approach is also applicable to indefinite problems, we consider a generalized incompressible Stokes problem with a variable viscosity. The stress tensor of a generalized Newtonian fluid with viscosity μ is given by $\boldsymbol{\sigma}(\mathbf{u}, p) = 2\mu\boldsymbol{\varepsilon}(\mathbf{u}) - pI$ and

TABLE 6

Time-to-solution and relative time-to-solution in the linear elastostatics benchmark for nodal integration and physical scaling recorded for different refinement levels ℓ and parameters m. The measurements were conducted on SuperMUC Phase 2.

ℓ	Nodal integration tts [s]	Physical scaling tts [s]	Rel. tts
	$m = 1$		
1	0.10	0.08	0.88
2	0.84	0.34	0.40
3	4.48	2.39	0.53
4	42.36	20.62	0.49
5	381.03	176.84	0.46
6	3065.13	1394.86	0.46
	$m = 2$		
1	0.09	0.09	0.93
2	0.53	0.36	0.68
3	4.79	2.35	0.49
4	44.31	20.73	0.47
5	381.06	174.61	0.46
6	3170.62	1420.91	0.45
ℓ	Nodal integration tts [s]	Physical scaling tts [s]	Rel. tts
	$m = 3$		
1	0.09	0.08	0.91
2	0.58	0.35	0.61
3	4.76	2.43	0.51
4	45.51	21.02	0.46
5	379.64	174.09	0.46
6	3159.07	1420.09	0.45
	$m = 8$		
1	0.09	0.08	0.91
2	0.49	0.42	0.85
3	4.53	2.26	0.50
4	42.36	20.35	0.48
5	369.05	168.28	0.46
6	3079.40	1374.16	0.45

depends not only on the velocity $\mathbf{u}$ but also on the pressure p. The problem considered in this section is modeled by the equations

$$\begin{aligned} -\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} &= \mathbf{f} \quad \text{in } \Omega, \\ \boldsymbol{\nabla} \cdot \mathbf{u} &= 0 \quad \text{in } \Omega, \\ \mathbf{u} &= \mathbf{g} \quad \text{on } \Gamma_{\mathrm{D}}, \\ \boldsymbol{\sigma} \cdot \mathbf{n} &= \hat{\mathbf{t}} \quad \text{on } \Gamma_{\mathrm{N}} \end{aligned}$$

on a domain $\Omega \subset \mathbb{R}^3$ with a Dirichlet boundary Γ_{D} and Neumann boundary Γ_{N}. For the well posedness of the problem, the finite element spaces need to meet a uniform inf-sup condition, which is not the case for an equal-order $P1$ discretization. Therefore, we add a level-dependent residual-based stabilization term c_ℓ [10] to the mass conservation equation, i.e., $c_\ell(p, q) = -\frac{h_\ell^2}{12} \langle \boldsymbol{\nabla} p, \boldsymbol{\nabla} q \rangle_\Omega$. If $\Gamma_{\mathrm{N}} = \emptyset$, then the pressure is not unique

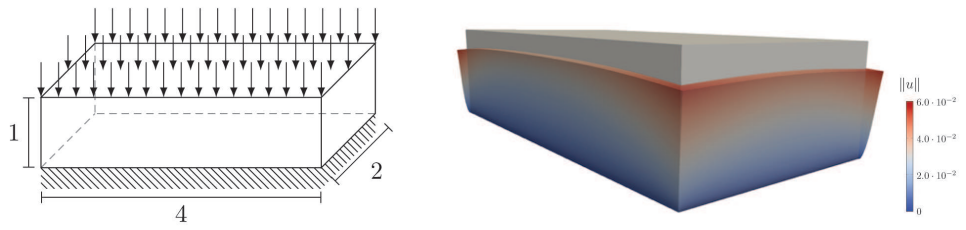


FIG. 9. *Experimental setup and dimensions of the metal foam (left). Initial (gray) and displaced (colored) foam after applying the force on top. The displacement is magnified by a factor of 5. The numerical solution was computed on refinement level $\ell = 4$ with standard nodal integration (right).*

TABLE 7

Errors of the linear elastostatics example with external forces in the discrete L^2 norm, convergence rates, number of V-cycle iterations, time-to-solution, and relative time-to-solution for nodal integration, and physical scaling recorded for different refinement levels ℓ.

		Nodal integration				Physical scaling				Rel.
ℓ	DoFs	error	eoc	iter	tts [s]	error	eoc	iter	tts [s]	tts
1	$1.23 \cdot 10^4$	$4.46 \cdot 10^{-4}$	0.00	12	0.38	$4.44 \cdot 10^{-4}$	0.00	12	0.42	1.11
2	$9.86 \cdot 10^4$	$1.70 \cdot 10^{-4}$	1.39	16	0.64	$1.69 \cdot 10^{-4}$	1.39	16	0.67	1.05
3	$7.89 \cdot 10^5$	$6.45 \cdot 10^{-5}$	1.40	18	1.27	$6.42 \cdot 10^{-5}$	1.40	18	1.20	0.94
4	$6.31 \cdot 10^6$	$2.31 \cdot 10^{-5}$	1.48	19	4.87	$2.30 \cdot 10^{-5}$	1.48	19	3.74	0.77
5	$5.05 \cdot 10^7$	$6.59 \cdot 10^{-6}$	1.81	19	30.02	$6.57 \cdot 10^{-6}$	1.81	19	19.22	0.64

up to a constant, and we enforce uniqueness by demanding that the mean value of the pressure be zero. This equal-order discretization is inconsistent and does not conserve the mass locally; however, the discrete solutions still converge with the optimal order. There are possibilities of obtaining local mass conservation by a postprocess, which is discussed in [32].

5.2.1. Stationary geophysics example. To demonstrate that the presented method is also suitable for solving geophysical problems, we present an example inspired by convection in Earth's mantle. The domain is chosen as $\Omega = (0,1)^3$ with $\Gamma_D = \partial\Omega$ and $\Gamma_N = \emptyset$. In this scenario, the viscosity and the volume forces depend on the temperature. Therefore, we construct a temperature field ϑ, resembling a temperature plume in Earth's mantle given by the formula

$$\vartheta(x,y,z) = \frac{89\,\mathrm{e}^{-30\left(z+\left(\frac{3r}{2}+\frac{3}{4}\right)\left(r-\frac{1}{2}\right)-\frac{3}{10}\right)^2-10\,r^2}}{100} + \frac{49\,\mathrm{e}^{-100\,r^2}}{50\left(\mathrm{e}^{17\,z-\frac{1819}{200}}+1\right)},$$

with $r(x,y,z) = \sqrt{\frac{13\,(x-\frac{1}{2})^2}{10} + \frac{27\,(y-\frac{1}{2})^2}{10}}$. Note that the temperature field is not radially symmetric, and therefore no problem reduction due to symmetry is possible. The viscosity μ of the fluid is then given by an exponential law with a jump across a horizontal plane, i.e.,

$$\mu(x,y,z) = \mathrm{e}^{-\vartheta(x,y,z)} \cdot \begin{cases} 10^{-2}, & z > \frac{3}{4}, \\ 1 & \text{else.} \end{cases}$$

See Figure 10 (left) for an illustration. In cases like this, where the location of a jump is known a priori, it is possible to resolve the jump via the macro-mesh, since the

standard on-the-fly integration is performed across these interfaces. If the locations of jumps are not known beforehand, it is still possible to locally mark elements where the standard on-the-fly integration should be performed. This solution will, of course, reduce the performance because of the additional branches in the code and possible load imbalances between processes. Stokes flow problems with larger viscosity jumps or a highly heterogeneous viscosity require more sophisticated preconditioners than the geometric multigrid solver used here. See, e.g., [33] for how to construct a robust iterative solver in this case. Additionally, we assume a gravitational source term $\mathbf{f} = \vartheta \cdot (0, 0, 10)^\top$ arising from a Boussinesq approximation [32, 31]. Figure 10 (right) shows the velocity streamlines of the numerical solution using nodal integration computed on a mesh with 50,331,648 tetrahedra.

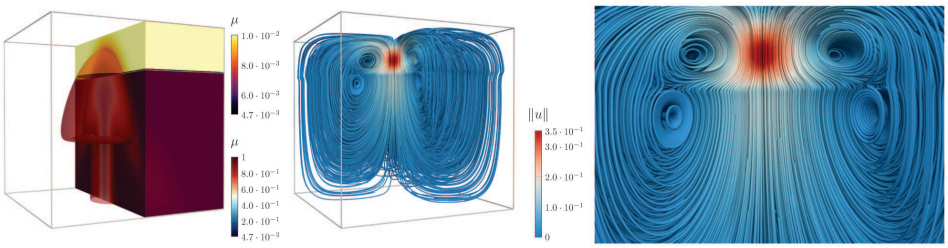


FIG. 10. *Viscosity μ depending on the given plume temperature field ϑ with isosurface $\mu = 0.85$ in the lower part and $\mu = 0.0085$ in the upper part (left). Velocity streamlines of the numerical solution computed on a mesh with* 50,331,648 *tetrahedra using nodal integration (middle). Zoom on the velocity streamlines at the center (right).*

In the following scenario, the coarsest level $\ell = 0$ is discretized by 786,432 tetrahedra, and each system is solved using 12,288 compute cores on SuperMUC-NG. The finest level $\ell = 6$ involved solving a system with about $1.03 \cdot 10^{11}$ DoFs. In order to solve the systems, we employ the inexact Uzawa solver presented in [14], with variable $V(3, 3)$ cycles where two smoothing steps are added to each coarser refinement level which enforces convergence of the method. As a smoother, we again employ the hybrid Gauss–Seidel method but with a relaxation parameter of 0.3 in the pressure part. On the coarsest level, we employ the diagonally preconditioned MINRES method, since the problem is not positive definite but symmetric. The solutions obtained with both approaches do not show any visual differences on all levels. In Table 8, we report on the number of inexact Uzawa iterations and tts for different refinement levels ℓ. The relative tts of the physical scaling is based on the tts of the nodal integration. The worse relative tts in comparison to the linear elasticity examples is due to the fact that the cost of the divergence matrices and the stabilization matrix need to be taken into account. Since the stencils of these matrices are constant on each macro-primitive and do not depend on a coefficient, we employ specialized kernels for them in all of the approaches. This part of the cost remains unchanged for both approaches. Therefore, assuming an optimal relative tts of 61% for the velocity block results in a theoretical optimal relative tts of only about 78% for the total Stokes operator. This value is close to the value observed for $\ell = 6$ in Table 8. We observe that the number of V-cycle iterations required to obtain the relative residual tolerance is larger for multigrid hierarchies with fewer levels compared to a larger number of levels. Furthermore, the tts for the levels $\ell = 1$ to $\ell = 4$ are very close to each other. Since the number of MPI ranks is fixed and the coarse grid solver does not scale optimally, most of the time is

spent on the coarse grid if the number of multigrid levels is small. This cost of the coarse grid solver decreases relatively if the number of multigrid levels increases and more time is spent for smoothing on the finer levels.

TABLE 8
Velocity and pressure errors of the stationary geophysics example in the discrete L^2 norm, convergence rates, number of inexact Uzawa iterations, time-to-solution, and relative time-to-solution for nodal integration and physical scaling recorded for different refinement levels ℓ.

		Nodal integration		Physical scaling		Rel.
ℓ	DoFs	iter	tts [s]	iter	tts [s]	tts
1	$4.19 \cdot 10^6$	12	36.58	12	39.30	1.07
2	$3.35 \cdot 10^7$	11	32.58	11	34.98	1.07
3	$2.68 \cdot 10^8$	10	31.69	10	32.22	1.02
4	$2.15 \cdot 10^9$	9	32.60	9	32.38	0.99
5	$1.72 \cdot 10^{10}$	9	62.84	9	54.91	0.87
6	$1.03 \cdot 10^{11}$	8	262.74	8	197.58	0.75

5.2.2. Nonlinear generalized Stokes problem. In this section, we consider the scenario of a nonlinear incompressible Stokes problem where the fluid is assumed to be of generalized Newtonian type, modeled by a shear-thinning Carreau model,

$$\mu(\mathbf{u}) = \eta_\infty + (\eta_0 - \eta_\infty)\left(1 + \kappa|\boldsymbol{\varepsilon}(\mathbf{u})|^2\right)^r.$$

The considered parameters in dimensionless form are specified in Figure 11 (left). These parameters stem from experimental results; cf. [16, Chapter II].

The computational domain Ω is depicted in Figure 11 (right), discretized by 14,208 tetrahedra on the coarsest level $\ell = 0$. The boundary $\partial\Omega$ is composed of Dirichlet and Neumann parts, i.e., $\partial\Omega = \Gamma_\mathrm{D} \cup \Gamma_\mathrm{N}$ with $\Gamma_\mathrm{D} = \{(x, y, z) \in \partial\Omega \mid x < 5\}$ and $\Gamma_\mathrm{N} = \partial\Omega \backslash \Gamma_\mathrm{D}$. The volume force term $\mathbf{f}$, the external forces $\hat{\mathbf{t}}$, and the Dirichlet boundary term $\mathbf{g}$ are set to

$$\mathbf{f} = (0, 0, 100)^\top,\ \hat{\mathbf{t}} = (0, 0, 0)^\top,\ \text{and}\ \mathbf{g} = 16\, y\,(1 - y)\, z\,(1 - z) \cdot (1, 0, 1)^\top.$$

We solve this nonlinear system by applying an inexact fixed-point iteration similar to the nonlinear solver described in [13], where the underlying linear systems are only solved approximately to prevent oversolving. The pseudocode of our approach is presented in Algorithm 5.1. The inexact Uzawa multigrid solver described in the previous subsection is used for the computations in this subsection. The problem is solved using a total of 111 MPI ranks with three compute nodes on SuperMUC-NG. Following the standard notation, the discretized saddle-point problem in a single fixed-point iteration reads

$$\begin{pmatrix} \mathsf{A}\left(\mu^{(n)}\right) & \mathsf{B}^\top \\ \mathsf{B} & \mathsf{C} \end{pmatrix} \begin{pmatrix} \mathsf{u}^{(n+1)} \\ \mathsf{p}^{(n+1)} \end{pmatrix} = \begin{pmatrix} \mathsf{f} \\ \mathsf{0} \end{pmatrix}. \tag{5.3}$$

In Figure 12, we plot the final viscosity profile and y-component of the velocity along the line $\theta = [0, 5] \times \{0.5\} \times \{0.42\}$ for both approaches computed on $\ell = 4$. We see that the solutions of the nodal integration and physical scaling approaches coincide. For the sake of completeness, we also present the unphysical scaling results, which yield different curves in both the top and bottom of Figure 12.

Parameter	Value
η_0	140.764
η_∞	1.0
κ	212.2
r	−0.325

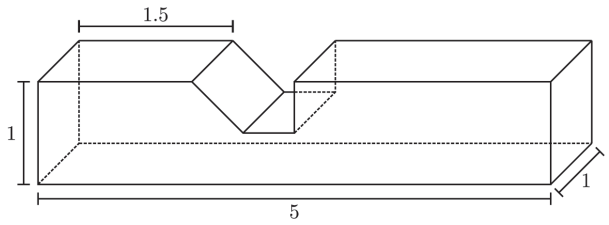


FIG. 11. *Dimensionless parameters for the Carreau viscosity model (left). Experimental setup and dimensions of channel (right).*

Algorithm 5.1 Fixed-point iterations coupled with a multigrid solver.

Set $\mathsf{u}^{(0)} = \mathsf{0}$, $\mathsf{p}^{(0)} = \mathsf{0}$, $n = 0$
Set $\mu^{(0)} = \mu(\mathsf{u}^{(0)})$ by employing the local approximation from section 3
repeat
  Solve system (5.3) for $\mathsf{u}^{(n+1)}$ and $\mathsf{p}^{(n+1)}$ by applying an inexact Uzawa V(3,3)-cycle
  Set $\mu^{(n+1)} = \mu\left(\mathsf{u}^{(n+1)}\right)$ by employing the local approximation from section 3
  Set $n = n + 1$
until $\max_i\{\|\mathsf{u}_i^{(n)} - \mathsf{u}_i^{(n-1)}\|_2\} < 10^{-3} \cdot \max_i\{\|\mathsf{u}_i^{(n)}\|_2\}$
Solve system (5.3) for $\mathsf{u}^{(n+1)}$ and $\mathsf{p}^{(n+1)}$ by applying final Uzawa V(3,3)-cycles until a relative residual of 10^{-3} is obtained
Set $\mu^{(n+1)} = \mu\left(\mathsf{u}^{(n+1)}\right)$ by employing the local approximation from section 3
return $\mathsf{u}^{(n+1)}$, $\mathsf{p}^{(n+1)}$, and $\mu^{(n+1)}$

In Table 9, we report on the number of fixed-point iterations, the number of final iterations, and the absolute and relative tts for the nodal integration and physical scaling. We observe a relative tts of about 87% on the finest level $\ell = 6$.

Since the coefficient μ changes after each multigrid V-cycle, the caching of face stencils as was done in the previous sections is not possible. This has a large impact on the run-time for lower levels in the hierarchy, since the cost may be dominated by the face primitives. Only asymptotically, for fine levels, the cost of the face primitives is small compared to the cost of the element primitives. In this numerical experiment, the solver performance is worse than in the previous examples, because of the inherent difficulty of the nonlinear problem and the expensive on-the-fly nodal integration of the bilinear form on the macro-faces.

6. Conclusion. We have presented a new method to improve the performance of matrix-free operator applications for vector-valued second-order elliptic PDEs. Although we restricted ourselves to linear finite elements on hierarchical hybrid grids, the idea of exploiting structure and symmetry in the mesh applies for higher orders as well, which we described in Remark 2.3. The method is based on scaling reference stencils originating from a constant coefficient discretization by variable coefficients. We showed that in theory a correction term is required in cases where the coefficient is not constant. Furthermore, we presented how to precompute these correction terms in the case of HHGs and how to rescale them using the coefficient. This new approach was aimed at reducing memory traffic and at reducing the number of required FLOPs. In order to show this, we first derived theoretical models about the required number of

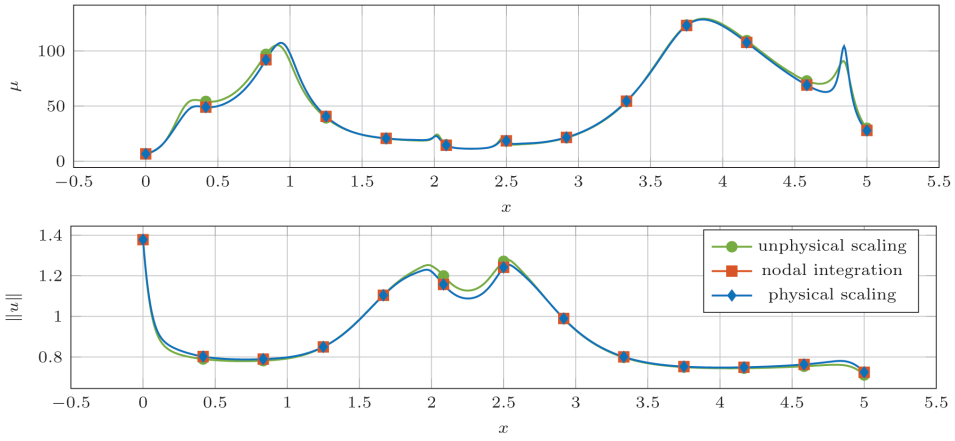


FIG. 12. *Viscosity profile (top) and profile of the velocity magnitude (bottom) plotted over θ for different assembly approaches.*

TABLE 9
Relative time-to-solution comparison of the nodal integration and physical scaling approach for the nonlinear generalized Stokes problem.

		Nodal integration			Physical scaling			Rel.
ℓ	DoFs	fixed-point iter	final iter	tts [s]	fixed-point iter	final iter	tts [s]	tts
3	$4.69 \cdot 10^6$	26	14	42.50	26	14	41.80	0.98
4	$3.82 \cdot 10^7$	29	7	127.98	29	7	121.87	0.95
5	$3.08 \cdot 10^8$	25	8	571.84	29	8	578.57	1.01
6	$2.47 \cdot 10^9$	24	7	3349.39	25	6	2925.09	0.87

FLOPs and the memory traffic. We validated these estimates using benchmarks and numerical experiments on the supercomputer SuperMUC-NG. The results have shown that specially designed matrix-free methods such as ours may be beneficial compared to matrix-based methods not only for higher-order discretizations but also for low-order discretizations, where the arithmetic intensity is lower. The numerical benchmarks and experiments involving linear elasticity and Stokes flow also showed that our method is faster than and comparably accurate to standard methods. Although we applied this method using only a geometric multigrid solver, it can also be used with more scalable coarse grid solvers, such as algebraic multigrid, or combined with preconditioners designed for large viscosity jumps or heterogeneous viscosities in Stokes flow problems. This makes our method attractive for highly resolved simulations on current and future machines where the available memory is limited compared to the compute power.

Acknowledgments. The authors wish to thank the referees for the detailed feedback they gave during the review process. Their insights significantly improved the quality of the final manuscript. The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (GCS, https://www.gauss-centre.eu/) for funding this project by providing computing time on the GCS supercomputer SuperMUC at Leibniz Supercomputing Centre (LRZ, www.lrz.de).

REFERENCES

[1] A. Afzal, *The Cost of Computation: Metrics and Models for Modern Multicore-Based Systems in Scientific Computing*, Master's thesis, Department Informatik, Friedrich Alexander Universität Erlangen-Nürnberg, 2015.

[2] C. L. Alappat, J. Hofmann, G. Hager, H. Fehske, A. R. Bishop, and G. Wellein, *Understanding HPC Benchmark Performance on Intel Broadwell and Cascade Lake Processors*, preprint, https://arxiv.org/abs/2002.03344, 2020.

[3] P. Arbenz, G. H. van Lenthe, U. Mennel, R. Müller, and M. Sala, *A scalable multi-level preconditioner for matrix-free μ-finite element analysis of human bone structures*, Int. J. Numer. Methods Engrg., 73 (2008), pp. 927–947.

[4] S. Bauer, D. Drzisga, M. Mohr, U. Rüde, C. Waluga, and B. Wohlmuth, *A stencil scaling approach for accelerating matrix-free finite element implementations*, SIAM J. Sci. Comput., 40 (2018), pp. C748–C778, https://doi.org/10.1137/17M1148384.

[5] B. Bergen, *Hierarchical Hybrid Grids: Data Structures and Core Algorithms for Efficient Finite Element Simulations on Supercomputers*, SCS Publishing House, 2005.

[6] B. Bergen and F. Hülsemann, *Hierarchical hybrid grids: Data structures and core algorithms for multigrid*, Numer. Linear Algebra Appl., 11 (2004), pp. 279–291.

[7] B. Bergen, G. Wellein, F. Hülsemann, and U. Rüde, *Hierarchical hybrid grids: Achieving TERAFLOP performance on large scale finite element simulations*, Int. J. Parallel Emergent Distrib. Syst., 22 (2007), pp. 311–329.

[8] J. Bey, *Tetrahedral grid refinement*, Computing, 55 (1995), pp. 355–378.

[9] J. Bielak, O. Ghattas, and E.-J. Kim, *Parallel octree-based finite element method for large-scale earthquake ground motion simulation*, CMES Comput. Model. Eng. Sci., 10 (2005), pp. 99–112.

[10] F. Brezzi and J. Pitkäranta, *On the stabilization of finite element approximations of the Stokes equations*, in Efficient Solutions of Elliptic Systems, Springer, 1984, pp. 11–19.

[11] J. Brown, *Efficient nonlinear solvers for nodal high-order finite elements in 3D*, J. Sci. Comput., 45 (2010), pp. 48–63.

[12] G. F. Carey and B.-N. Jiang, *Element-by-element linear and nonlinear solution schemes*, Comm. Appl. Numer. Methods, 2 (1986), pp. 145–153.

[13] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408, https://doi.org/10.1137/0719025.

[14] D. Drzisga, L. John, U. Rüde, B. Wohlmuth, and W. Zulehner, *On the analysis of block smoothers for saddle point problems*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 932–960, https://doi.org/10.1137/16M1106304.

[15] C. Flaig and P. Arbenz, *A highly scalable matrix-free multigrid solver for μFE analysis based on a pointer-less octree*, in Large-Scale Scientific Computing: 8th International Conference (LSSC 2011), Sozopol, Bulgaria, Revised Selected Papers, I. Lirkov, S. Margenov, and J. Waśniewski, eds., Springer, 2012, pp. 498–506.

[16] G. P. Galdi, R. Rannacher, A. M. Robertson, and S. Turek, *Hemodynamical Flows: Modeling, Analysis and Simulation*, Birkhäuser, 2008.

[17] L. J. Gibson and M. F. Ashby, *Cellular Solids: Structure and Properties*, Cambridge University Press, 1999.

[18] B. Gmeiner, T. Gradl, H. Köstler, and U. Rüde, *Highly parallel geometric multigrid algorithm for hierarchical hybrid grids*, in NIC Symposium 2012 - Proceedings, Jülich, Germany, NIC Ser. 45, 2012, pp. 323–330.

[19] G. Hager and G. Wellein, *Introduction to High Performance Computing for Scientists and Engineers*, CRC Press, 2010.

[20] A. Ilic, F. Pratas, and L. Sousa, *Cache-aware Roofline model: Upgrading the loft*, IEEE Comput. Architecture Lett., 13 (2013), pp. 21–24.

[21] Intel Corp., *Intel Advisor*, https://software.intel.com/en-us/intel-advisor-xe, 2019.

[22] Intel Corp., *Intel VTune Profiler*, https://software.intel.com/en-us/vtune, 2019.

[23] N. Kohl, D. Thönnes, D. Drzisga, D. Bartuschat, and U. Rüde, *The HyTeG finite-element software framework for scalable multigrid solvers*, Int. J. Parallel Emergent Distrib. Syst., 34 (2019), pp. 477–496.

[24] M. Kronbichler and K. Kormann, *A generic interface for parallel cell-based finite element operator application*, Comput. & Fluids, 63 (2012), pp. 135–147.

[25] K. Ljungkvist, *Matrix-free finite-element computations on graphics processors with adaptively refined unstructured meshes*, in Proceedings of the 25th High Performance Computing Symposium (HPC '17), Society for Computer Simulation International, 2017, pp. 1:1–1:12.

[26] K. Ljungkvist and M. Kronbichler, *Multigrid for Matrix-Free Finite Element Computations*

on Graphics Processors, Tech. report 2017-006, Department of Information Technology, Uppsala University, 2017.

[27] J. Loffeld and J. Hittinger, *On the arithmetic intensity of high-order finite-volume discretizations for hyperbolic systems of conservation laws*, Int. J. High Performance Comput. Appl., 33 (2019), pp. 25–52.

[28] LRZ, *Hardware of SuperMUC-NG*, https://doku.lrz.de/display/PUBLIC/Hardware+of+SuperMUC-NG (retrieved 25 February 2020).

[29] LRZ, *SuperMUC Petascale System*, https://www.lrz.de/services/compute/supermuc/systemdescription/ (retrieved 29 November 2018).

[30] D. A. May, J. Brown, and L. L. Pourhiet, *A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow*, Comput. Methods Appl. Mech. Engrg., 290 (2015), pp. 496–523.

[31] Y. Ricard, *Physics of mantle convection*, Treatise on Geophysics, 7 (2007), pp. 31–81.

[32] U. Rüde, C. Waluga, and B. Wohlmuth, *Mass-corrections for the conservative coupling of flow and transport on collocated meshes*, J. Comput. Phys., 305 (2016), pp. 319–332.

[33] J. Rudi, G. Stadler, and O. Ghattas, *Weighted BFBT preconditioner for Stokes flow problems with highly heterogeneous viscosity*, SIAM J. Sci. Comput., 39 (2017), pp. S272–S297, https://doi.org/10.1137/16M108450X.

[34] H. Stengel, J. Treibig, G. Hager, and G. Wellein, *Quantifying performance bottlenecks of stencil computations using the execution-cache-memory model*, in Proceedings of the 29th ACM International Conference on Supercomputing, ACM, 2015, pp. 207–216.

[35] B. van Rietbergen, H. Weinans, R. Huiskes, and B. Polman, *Computational strategies for iterative solutions of large FEM applications employing voxel data*, Int. J. Numer. Methods Engrg., 39 (1996), pp. 2743–2767.

[36] S. Williams, A. Waterman, and D. Patterson, *Roofline: An insightful visual performance model for multicore architectures*, Commun. ACM, 52 (2009), pp. 65–76.

[37] Y. Zhang, D. Rodrigue, and A. Ait-Kadi, *High density polyethylene foams.* II. *Elastic modulus*, J. Appl. Polymer Sci., 90 (2003), pp. 2120–2129.

[38] O. C. Zienkiewicz and J. Z. Zhu, *The superconvergent patch recovery and a posteriori error estimates: Part* 1*: The recovery technique*, Int. J. Numer. Methods Engrg., 33 (1992), pp. 1331–1364.