New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
5 December 2016

REVISED
19 February 2017

ACCEPTED FOR PUBLICATION
27 March 2017

PUBLISHED
16 May 2017

PAPER

Kassiopeia: a modern, extensible C++ particle tracking package

Daniel Furse[1], Stefan Groh[2], Nikolaus Trost[3], Martin Babutzka[2], John P Barrett[1], Jan Behrens[4], Nicholas Buzinsky[1], Thomas Corona[5,6], Sanshiro Enomoto[7], Moritz Erhard[2], Joseph A Formaggio[1], Ferenc Glück[3,8], Fabian Harms[3], Florian Heizmann[2], Daniel Hilk[2], Wolfgang Käfer[2], Marco Kleesiek[2], Benjamin Leiber[2], Susanne Mertens[9], Noah S Oblath[1], Pascal Renschler[2], Johannes Schwarz[3], Penny L Slocum[10], Nancy Wandkowsky[3], Kevin Wierman[5,6] and Michael Zacher[4]

[1] Laboratory for Nuclear Science, Massachusetts Institute of Technology, Cambridge, MA 02139, United States of America
[2] Institute of Experimental Nuclear Physics (IEKP), Karlsruhe Institute of Technology (KIT), Wolfgang-Gaede-Str. 1, D-76131 Karlsruhe, Germany
[3] Institute for Nuclear Physics (IKP), Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, D-76344 Eggenstein-Leopoldshafen, Germany
[4] Institut für Kernphysik, Westfälische Wilhelms-Universität Münster, D-48149 Münster, Germany
[5] Department of Physics and Astronomy, University of North Carolina, Chapel Hill, NC 27599, United States of America
[6] Triangle Universities Nuclear Laboratory, Durham, NC 27708, United States of America
[7] Center for Experimental Nuclear Physics and Astrophysics, Department of Physics, University of Washington, Seattle, WA 98195, United States of America
[8] Wigner Research Institute for Physics, Budapest POB 49, Hungary
[9] Max Planck Institute for Physics, Munich & Technical University Munich, D-80333 Munich, Germany
[10] Department of Physics, Yale University, PO Box 208120, New Haven, CT 06520, United States of America

E-mail: nikolaus.trost@kit.edu

Keywords: simulation, software, electromagnetic fields, particle tracking, C++

Abstract

The KASSIOPEIA particle tracking framework is an object-oriented software package using modern C++ techniques, written originally to meet the needs of the KATRIN collaboration. KASSIOPEIA features a new algorithmic paradigm for particle tracking simulations which targets experiments containing complex geometries and electromagnetic fields, with high priority put on calculation efficiency, customizability, extensibility, and ease-of-use for novice programmers. To solve KASSIOPEIA's target physics problem the software is capable of simulating particle trajectories governed by arbitrarily complex differential equations of motion, continuous physics processes that may in part be modeled as terms perturbing that equation of motion, stochastic processes that occur in flight such as bulk scattering and decay, and stochastic surface processes occurring at interfaces, including transmission and reflection effects. This entire set of computations takes place against the backdrop of a rich geometry package which serves a variety of roles, including initialization of electromagnetic field simulations and the support of state-dependent algorithm-swapping and behavioral changes as a particle's state evolves. Thanks to the very general approach taken by KASSIOPEIA it can be used by other experiments facing similar challenges when calculating particle trajectories in electromagnetic fields. It is publicly available at https://github.com/KATRIN-Experiment/Kassiopeia.

1. Introduction

KASSIOPEIA is a software package for the purpose of tracking particles in complex geometries and electromagnetic fields. It has been developed in order to meet the simulation needs of the KATRIN collaboration, which endeavors to measure the absolute neutrino mass scale through tritium β-decay. Strong evidence for the existence of non-zero neutrino mass follows from the legion of experiments demonstrating flavor oscillation phenomena [1–7]. The discovery of neutrino oscillations (hence, neutrino mass) is the first demonstration of neutrino properties beyond the Standard Model prescription. However, oscillation phenomena depend only on the differences of the squares of neutrino mass eigenvalues $\Delta m_{ij}^2 \equiv m_j^2 - m_i^2$; the

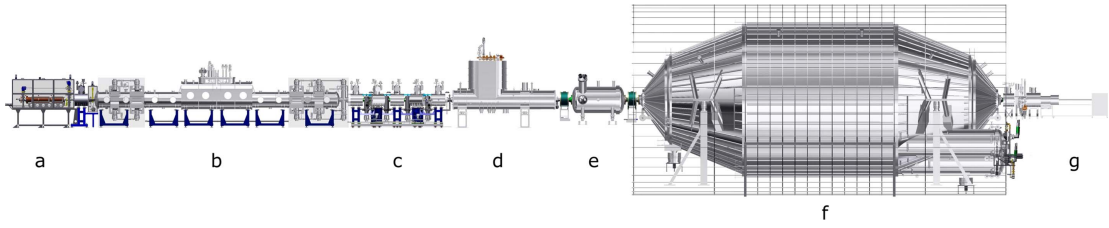


Figure 1. Schematic overview of the 70 m KATRIN experimental beam-line: (a) the rear section, (b) the tritium source, (c) the DPS, (d) the CPS, (e) the pre-spectrometer, (f) the main spectrometer and air-coil framework, and (g) the focal-plane detector system.

absolute neutrino mass scale does not enter into the description of oscillation phenomena. As such, the absolute neutrino mass remains one of the foremost open questions in neutrino physics at the present time.

The most sensitive direct searches for the electron neutrino mass to date are based on the investigation of the electron spectrum of tritium β-decay. The electron energy spectrum of tritium β-decay for a neutrino with component masses m_1, m_2, and m_3 (with mixing amplitudes U_{e1}, U_{e2}, and U_{e3}, respectively) is given (with some simplifications[11]) by

$$\frac{\mathrm{d}N}{\mathrm{d}E} \propto F(Z, E)pE(E_0 - E)\sum_{i=1,3}|U_{ei}|^2[(E_0 - E)^2 - m_i^2]^{\frac{1}{2}}\Theta(E_0 - E - m_i), \quad (1)$$

where $E(p)$ denotes the electron's kinetic energy (momentum), E_0 corresponds to the total decay energy, $F(Z, E)$ is the Fermi function, taking into account the Coulomb interaction of the outgoing electron in the final state, and $\Theta(E_0 - E - m_i)$ is the step function that ensures energy conservation. As both the matrix elements and $F(Z, E)$ are independent of m_ν, the dependence of the spectral shape on m_ν is given by the phase space factor only. The bound on the neutrino mass from tritium β-decay is independent of whether the electron neutrino is a Majorana or a Dirac particle.

Although the history of beta spectroscopy spans a variety of different magnetic and electrostatic spectrometers, the technique that has demonstrated the greatest sensitivity to neutrino mass has been MAC-E-Filters (Magnetic Adiabatic Collimation with Electrostatic Filtering). This type of spectrometer, originally based on the work by Kruit [8] was later utilized by the Mainz [9] and Troitsk [10] experiments to set a limit on the neutrino mass on the level of 2 eV. The MAC-E filter technique demands a smoothly varying magnetic field and has an energy resolution that is dictated by the ratio of the minimum to maximum magnetic field strength. This dictates that better energy resolution be accompanied by an increase in the physical size of the spectrometer. The KATRIN experiment's massive size and other design optimizations [11] will allow it to reach an energy resolution of 0.93 eV, and should allow it to place a limit on the neutrino mass approximately an order of magnitude better than the current state of the art [9, 10].

Figure 1 illustrates the overall components of the experiment, which include: (a) a rear section, used for calibrating the response of the detector and monitoring the source strength, (b) a windowless gaseous tritium source (WGTS), where 10^{11} electrons are produced per second by the decay of molecular tritium gas at a temperature of 30 K, (c) an electron transport and tritium elimination section, comprising (c) an active differential pumping (DPS) followed by (d) a passive cryo-pumping section (CPS), where the tritium flow is reduced by more than 14 orders of magnitude, (e) the electrostatic pre-spectrometer that offers the option to pre-filter the low-energy part of the tritium decay spectrum, (f) the large electrostatic main spectrometer of MAC-E-filter type that represents the precision energy filter for electrons, and (g) a segmented Si-PIN diode array to count the transmitted electrons. Further details on the experiment's design and performance parameters can be found elsewhere [11].

A suitable computational model of such a complex experiment is indeed necessary if one is to properly assess the results obtained by the experiment. Such a tool is essential for many tasks, from the estimation of background rates and systematic effects to modeling signal electron energy loss and backscattering at the silicon detector. Ultimately, detailed simulations based on measurements performed during installation are the most accurate tool for evaluating the sensitivity of the entire experiment, giving them a position of central importance. However, such performance goals often impose strict (and often contradictory) conditions on the simulation software. For one, the software needs to be able to accurately propagate electrons through the complex fields and geometries found in the experiment, while at the same time retain high efficiency and flexibility for such propagation. In addition, KATRIN needs to be able not only to calculate such trajectories, but also compute the

[11] Note that we have neglected modifications of the energy spectrum due to the nuclear matrix element and the molecular final state distribution.

electromagnetic fields *in situ*. Though tools exist for each task separately (e.g. GEANT4 [12] for particle tracking, COMSOL [13] for electromagnetic field calculations), none provide a unified treatment of fields and particle tracking that is adequate for the specific needs of KATRIN. COMSOL is a proprietary, closed source software package, and thus cannot be extended or adapted by the end-user. GEANT4 does provide an extensive list of particle interactions and material properties, and is unrivaled in its treatment of high energy physics, but KATRIN does not benefit from these features as it is mainly concerned with the navigation of low-energy particles traversing complex fields in a low to ultra-high vacuum. Furthermore, the tracking algorithms of GEANT4 were not designed with the expectation of needing to track many low-energy particles in the adiabatic limit, as this is not a common situation in high energy physics. Such particles undergo many cyclotron orbits and require excessively short step sizes to solve the equations of motion, unless they are treated appropriately (i.e. by way of the guiding center limit [14]). Tracking and navigation algorithms designed without this situation in mind often suffer from unacceptably long computation times, and this difficultly is typically compounded further in navigationally complex situations (such as in KATRIN's wire array modules) where the particle's position must be known with fine precision on the order of the smallest feature of nearby geometric components.

However, in direct comparison with long-established packages such as GEANT4 or COMSOL, it is only fair to mention that KASSIOPEIA, at the moment, contains only a limited number of interaction models which are in the focus of the KATRIN experiment: electron interactions from the kilo-electronvolt range down to the electronvolt regime in hydrogen, residual gases and silicon, all modeled in great detail, including not merely ionization but also elastic and excitation contributions. The need for these detailed models presented another argument to build a new package, as existing ones again could not be used. To broaden the scope of KASSIOPEIA, ion interactions are under development at present and in the future the additional usage of existing third party interaction modules is planned. Along the same lines, the interoperability with common formats for data output and simulation input (geometry formats, field representations) will be improved. At present geometries need to be described in KASSIOPEIA'S Extensible Markup Language (XML) format and (non-constant) fields should stem either from electrodes and coils implemented this way, or a (plain-text) field map. Generally it can be said that the overhead for introducing a new module such, as a reader for special expansions of—potentially even time-dependent—electric or magnetic fields (when the option to use a field map and interpolation does not suffice) or a new interaction, is fairly limited and can be done without full knowledge of all the internal workings of KASSIOPEIA—a developer guide will follow soon. Proven use-cases include but are not limited to: electron detection in silicon detectors, eigen-frequency calculation of static Penning traps, cooling of magnetically trapped electrons in residual gas, and studies of transmission properties of magnetic spectrometers. Future applications—given the integration of the appropriate interaction models—could also extend to germanium detectors, liquid xenon/argon time projection chambers, or further.

Furthermore, existing software tools did not provide the required flexibility sought by the KATRIN experiment. For instance, a researcher trying to design a new component of the spectrometer electrode system needs quick feedback on its influence on the fields, and ultimately needs to understand the influence it has on the dynamical properties of the spectrometer. Another might need to understand tritium ion propagation and scattering in the source of the experiment. Even two people working on the same topic might be interested in completely different aspects of the same physics, requiring different output from an otherwise identical simulation. Such examples are inexhaustible, which indicates a need for a simulation package that is very granular and allows for a large space of possible module combinations and arrangements, in a user-configurable way. The combined requirements of modularity, extensibility, and ease-of-use for novice programmers unfortunately have ruled out the majority of existing candidates.

In light of these facts, development began on KASSIOPEIA, a new general particle and field simulation package designed to meet the diverse needs outlined above. Since modularity, encapsulation and speed are essential, the development team decided to implement the simulation as an object-oriented design written in C++, with a user-interface based primarily on configuration files written in XML. We believe that the current version of the code, which is now publicly available, broadly satisfies the requirements set out above and will present a valuable and complementary approach to particle simulation packages available to the wider physics community.

The organization of this paper is as follows. Section 2 details the underlying design of the KASSIOPEIA software package and its configuration. Sections 3 and 4 describe the geometry model and the electromagnetic field calculations, respectively. Sections 5 through 8 provide a description of the generation of particle states (section 5), their propagation through simulation geometries by solving of their equations of motion (section 6), the treatment of stochastic interactions (section 7), and termination (section 8). Of subsequent interest are section 9, which outlines the various data output options, section 10, which describes the command structure responsible for maintaining the simulation state machine, and section 11, which demonstrates the visualization capabilities of this software. Finally, we conclude with some example use-cases and validation in sections 12 and 13.

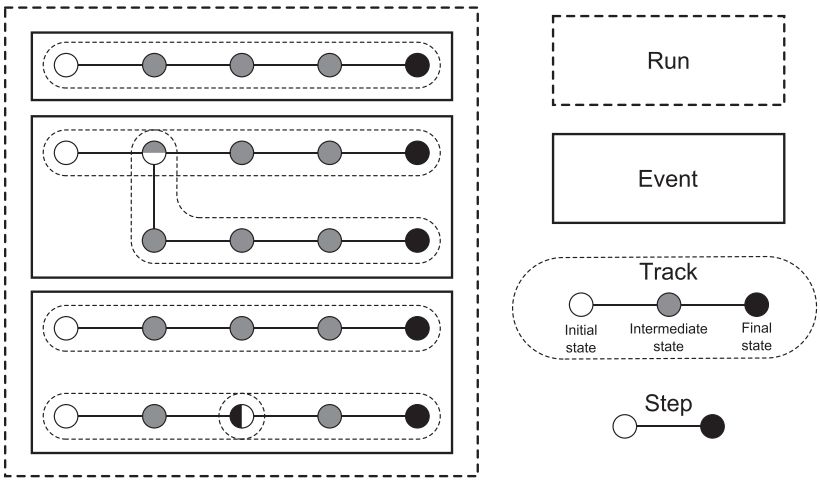


Figure 2. Schematic representation of a run with three events and a total of six tracks. Each track starts with an initial state (white) and ends with a final state (black). It consists of multiple steps—four for most of the given examples. Via interaction processes such as ionization within a track, a new particle and therefore a new track can be generated (track 3). Thus an event can consist out of multiple tracks, which is also the case for an event of a radioactive decay chain, for example, creating multiple initial particles and therefore multiple tracks within one event (tracks 4 and 5). A track can be split if it crosses a surface or changes a geometrical space, thus ending the old track and starting a new one (tracks 5 and 6). Figure from [15].

2. General design

The goal of particle tracking software and therefore also of KASSIOPEIA is to simulate the evolution of the physical state of multiple particles with very high precision and efficiency. The particle therefore represents a fundamental object, whose properties are to be modified by the algorithms of the simulation software. The inherent properties of a particle, its mass m and electric charge q, are fixed during initialization, while the dynamic properties, such as its position $\mathbf{x}$, and momentum $\mathbf{p}$, will evolve as the simulation progresses.

2.1. Organizational structure

KASSIOPEIA's data structure, which is filled by output from the simulation, is organized into four intuitive levels of detail: Step, Track, Event and Run. A schematic representation of this classification is visualized in figure 2. The individual levels are detailed as shown in figure 2.

- **Step:** The lowest level of organization in the simulation is a step. It represents the evolution of a particle over a small amount of time and space from an initial to a final state. The propagation of the particle is achieved by solving the equations of motion and by considering a variety of interactions with the surrounding matter and fields. Additionally, navigation within the defined geometry is performed to detect the crossing of surfaces or space boundaries.
- **Track:** The complete evolution of a particle from its point of origin to its termination is called a track, which can be seen as a sequential collection of steps. A particle and therefore a track, is typically created within an event generator or through an interaction like ionization. It can be terminated by a collection of terminators depending on specific states of the simulation. Additionally, a particle can also be terminated and a new one generated by the navigation when crossing a surface or changing a space, which thereby splits the track into two.
- **Event:** The next level of organization is an event, which is a collection of causally related tracks. Each event typically has one primary track corresponding to the primary particle created by a generator, and optionally additional secondary tracks created by splitting of the primary track or by new particles being generated during an interaction process. There are also specific generators which produce multiple causally related primary particles, for example in a radioactive decay sequence. Within one event, the primary particles created and all of their descendants are tracked step-by-step until they are terminated.
- **Run:** The highest level of organization within KASSIOPEIA is the run, which is a collection of events, whose number is pre-defined by the user in the configuration file. It represents one execution of the simulation for a fixed experimental setup. Multiple runs can be realized by running multiple KASSIOPEIA instances and merging the produced output files at the end.

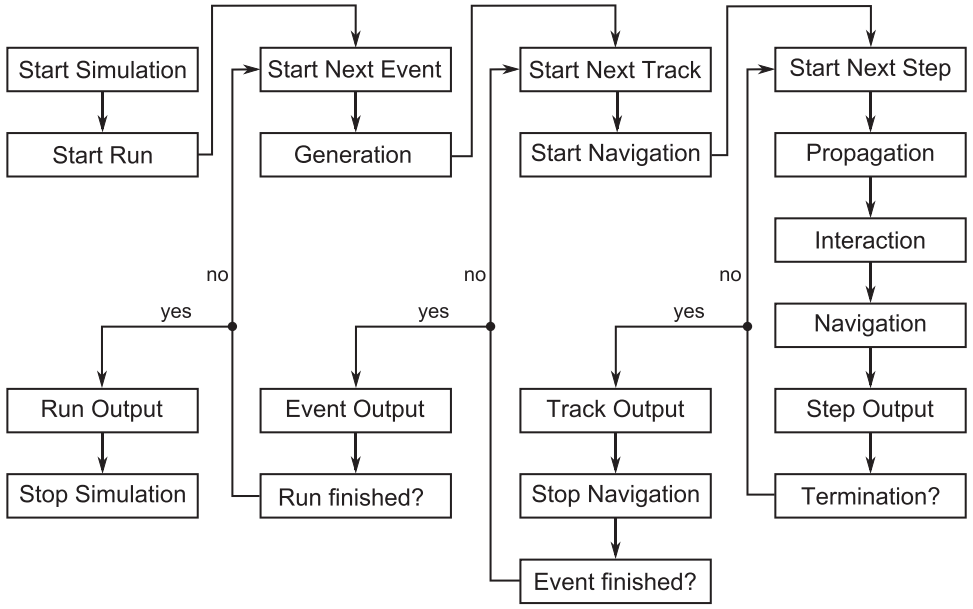


Figure 3. Simplified schematic representation of KASSIOPEIA's simulation algorithm, composed out of three loops over events, tracks and steps. Figure reprinted with permission from [15].

2.2. Simulation work-flow

The introduced data structures need to be populated by the simulation algorithm. A simplified and schematic chart of the simulation work-flow is visualized in figure 3. When the simulation is started, first the XML configuration file is parsed and the defined objects used in the simulation will be built and initialized, as detailed in section 2.4. Then the event loop is executed n times and in each loop a user-specified generator will produce one or multiple initial particles. For each of these particles a track is created and consecutive steps are performed until the track is terminated. User-defined quantities of the track including the initial and final particle state can then be written to disk before the next track is executed. If the tracking of all particles of the event is completed, including secondary particles created within the tracks, the specific event being executed is finished and the corresponding event output is written. After all n events have been completed, the run output is written and the simulation ends after the deinitialization of all created objects.

The most important part during particle tracking is the step loop, which is typically repeated a large number of times for each track. The schematic representation displayed in figure 3 corresponds to a simplification of a more sophisticated algorithm. In each step the particle is propagated by integrating its equations of motion over the user-defined step size. This is typically the most expensive part of the simulation as it involves many calculations of complex electric and magnetic fields, gradients and potentials. After the propagation step has been evaluated, the particle's mean free path length for each of the given interaction processes is calculated and the length at which the process will occur is determined probabilistically. If the randomly generated interaction length is less than the length of particle step's trajectory, then the terminal position of the particle is adjusted accordingly and the interaction process is executed on the modified final state of the particle. Additionally, the navigator checks if the particle has crossed any geometrical boundaries within the calculated trajectory. If this is the case, the final state of the particle is adjusted to lie on the crossing position of its trajectory with the given geometrical boundary. Interactions or commands associated with the boundary may then modify the particle state or induce a change in the configuration of the simulation, as will be detailed in the next section. After the propagation, interaction and navigation of the step is done, and the particle has reached its final state for that step, the information about the step and its initial and final particle states can be written to the output. Subsequently, the active terminators are called to check if the particle has reached a certain physical state where the user has defined it must stop. If this is the case, the track is finished, if not, then the step loop is repeated.

2.3. Modularity

The most powerful feature of KASSIOPEIA is its flexibility and modularity. The user can not only define the type of modules to be used in the simulation such as generators, field calculators, or interactions, but the whole composition of the simulation algorithm can be changed depending on the particle's geometrical state. This is achieved through the combination of the concept of a toolbox in conjunction with a collection of container classes and a dynamic list of commands. The toolbox exists to maintain all of objects that the user might use

during the course of simulation. The container classes (called root classes in KASSIOPEIA), serve to localize physics processes with similar attributes (discrete interactions, propagation, field calculation, etc) and execute the activated processes during the main simulation loop. The command list modifies the container classes and is responsible for the addition, removal or replacement of activated parts of the simulation. It is updated in a dynamic way depending on the current location of the particle. The details of these three aspects of the simulation are as follows:

- **Toolbox:** All objects of the simulation specified by the user are instantiated and stored in a so-called toolbox when the XML configuration file is parsed (see next section 2.4). This is the case for physics modules such as particle generators, field calculators or interaction processes, but also for completely different objects such as output components (see section 9).
- **Root classes:** The simulation algorithm works with container classes for the different types of processes displayed in figure 3. These container classes within KASSIOPEIA are called root classes. There are two different types, divided by whether multiple objects of the same kind are allowed to be active at the same time. The root terminator, for example, can contain a list of multiple terminator objects, so when being called by the simulation algorithm, it will call all its 'child' terminators. The root trajectory, however, being responsible for the propagation of the particle, can contain only one representation of the equations of motion, although these equations can be composed of multiple terms. The root classes are typically filled with user-specified default objects at the beginning of the simulation, but they may also be left completely empty. They will then be filled by the navigator depending on the geometrical state of the tracked particle.
- **Commands:** The manner in which root classes are filled by the navigator is completely exposed to the user. In the configuration file, a single or multiple nested navigation geometries can be defined. For geometric objects (see section 3) a basic distinction is made between a navigation space, a navigation side, (which is some subset of the boundary of a space), and a navigation surface (which is a free surface, not associated with a space). For each navigation geometry, a set of commands can be defined that are executed when the particle enters the corresponding geometry and are reversed, when the particle leaves it. A prerequisite of this command method is that nested geometries must be completely contained within their parent space. This is because processes which are associated with the parent space remain active inside the nested geometry, unless otherwise specified. This helps to avoid unnecessary deactivation and reactivation of objects upon traversal of the geometry. Commands are typically defined to add, remove or replace objects from the root classes. An example of this being the addition of an interaction to the root interaction for a certain surface or the replacement of the step size control algorithm for a certain trajectory representation.

All objects in the toolbox that are referred to in one of the given commands (and therefore may be used at some time during the simulation) are initialized at the beginning before the start of the simulation. When an object is added to the simulation, it is activated and subsequently deactivated upon removal. At the beginning of each track, the navigation is started and the simulation is put in a state which depends on the geometry and position of the particle. At each step, the navigator checks if the geometrical state of the particle has changed, and if so, will activate the corresponding navigation geometry and execute the associated commands. After each track is finished, the navigation is stopped and the state of the simulation is put back into the default mode.

With this flexible command concept it is possible to track particles through a variety of different lengths or physical processes with KASSIOPEIA using a single configuration file. A prime example is the entirety of the KATRIN experiment where electrons are generated in a gaseous tritium source, and propagate through the approximately 70 m long beam line until finally entering a silicon detector. In this case, the dominant physical processes change over the course of a particle's path through each separate region of the experiment. For example, in the source it is important to consider scattering off tritium molecules, whereas in the ultra-high vacuum of the main spectrometer there is very little scattering, but precision integration of the equations of motion in the complex electromagnetic fields becomes necessary. An even more drastic change must be executed upon entry of the silicon material of the detector, since the solid-state physics dictating electron interactions there proceed on a scale of μm. This whole journey of the particle can be described as a single track in KASSIOPEIA, as the simulation algorithm is adapted according to the correct underlying physics of each region of the experiment.

2.4. Initialization

A simulation run with KASSIOPEIA is completely defined by a configuration file. In this file all simulation input data are defined and created. This includes the entire geometry of the experiment, all kinds of different physics processes that may be executed in the simulation, as well as the level of detail in the output and recorded

```
<!-- definition of path variable -->
<define name="path" value="/path/to/file"/>
<!-- inclusion of geometry xml file -->
<include name="[path]/CoilGeometry.xml"/>

<!-- condition if coils should be used -->
<if condition="[use_coils]">
    <!-- loop to add an individual current to a total of 5 coils -->
    <loop variable="i" start="1" end="5" step="1">
        <!-- the total current added to each coil is the product -->
        <!-- of its real current and its number of turns -->
        <electromagnet spaces="coil_[i]" current="{[[i]_current]*[[i]_turns]}"/>
    </loop>
</if>
```

Figure 4. Example of an XML configuration file, showing the features of variables, includes, conditions, loops, and formula evaluations.

quantities. The geometry navigation commands and also basic properties of the simulation, such as the value of the initial random seed and the number of generated events, are also defined in the configuration. The configuration files for KASSIOPEIA are based on XML as specified in [16], with some additional features and extensions. The XML parser of KASSIOPEIA is composed of a chain of multiple XML processors and the parsing of the information is performed sequentially in the so-called SAX style. First an XML tokenizer creates tokens out of the data stream from the file, which are then fed into the chain of XML processors. These processors may modify the stream of tokens until the last processor in the chain finally creates the desired object. The modifications of the stream by the individual processors allows one the ability to perform advanced operations while the XML file is parsed, such as variable definitions, evaluation of mathematical formulas, or even loops and conditional statements. Furthermore, it is also possible to break the configuration into separate files through an include mechanism or write down a serialized version of a given configuration file. A snippet of an example configuration file is shown in figure 4.

3. Geometry

The geometry module of KASSIOPEIA comprises geometrical classes for a large number of different shapes, linear algebra methods, structures for the relation between geometrical elements, and an extension system to add arbitrary properties to shapes.

3.1. Shapes

The available shapes are divided into different types of surfaces and spaces. Both can be constructed from an XML configuration file by defining the necessary attributes that are required for the specific geometrical element. Each single shape is created with its own coordinate system depending on the attribute values chosen by the user. In example shown in figure 5, a box space, a cylinder space and a disk surface are constructed. The origin of the box is located in one of its corners, while the origin for the cylinder is in its center.

Aside from the above given examples of basic shapes, it is also possible to construct more complex arbitrary shapes, from a set of points connected either by lines or arcs. The resulting poly-line can be rotated or extruded to create non-trivial surfaces or spaces. All the features of the XML system as described in section 2.4 can be used, such as loops, variable definitions, and mathematical operations. This significantly reduces the amount of text needed to describe a large number of similar geometrical objects. If the available general purpose shapes cannot easily represent the full complexity of an experiment, special purpose spaces and surfaces can be incorporated by a user following the inheritance mechanism of the geometry package. As an example of the capability and flexibility of the geometry system, figure 6 shows a comparison between the geometry model and a photograph of the KATRIN main spectrometer interior.

3.2. Structure

Surface and spaces have to be placed and related to each other to form a nested relationship. Spaces automatically contain a set of boundary surfaces and may also contain other spaces or surfaces. The policy of KASSIOPEIA's geometry module is that child spaces or child surfaces need to be completely contained within their nesting parent space with no protrusion allowed. The user is responsible for ensuring this condition is satisfied since as of yet no automatic collision detection between geometry objects is performed.

```
<!-- shapes -->
<box_space name="box_A" xa="0.0" ya="0.0" za="0.0" xb="1.0" yb="1.0" zb="1.0"/>
<cylinder_space name="cylinder_C" z1="-0.4" z2="0.4" r="0.3"/>
<disk_surface name="disk_a" z="0.0" r="0.1"/>

<!-- structure -->
<space name="outer_box" node="box_A">
    <space name="inner_cylinder" node="cylinder_C">
        <transformation rotation_euler="90 60 -90"/>
        <transformation displacement="0.5 0.5 0.5"/>
    </space>
    <surface name="inner_disk_a" node="disk_a">
        <transformation rotation_euler="90 45 -90"/>
        <transformation displacement="0.5 0.5 0.8"/>
    </surface>
</space>

<!-- extension -->
<electromagnet spaces="outer_box/inner_cylinder" current="50"/>
```

Figure 5. In the geometry XML file, first the definitions of the involved shapes are given, which in this example includes a box, a cylinder, and a disk surface. Then the relation between the shapes is structured. Here the cylinder and the disk are placed inside the box, and are rotated and displaced in the process. Finally, additional information is added to the cylinder inside the box. In this case, it is made into an electromagnet by specifying some associated current. Units of length are in meters, rotation angles are in degrees, and current is given in amperes.

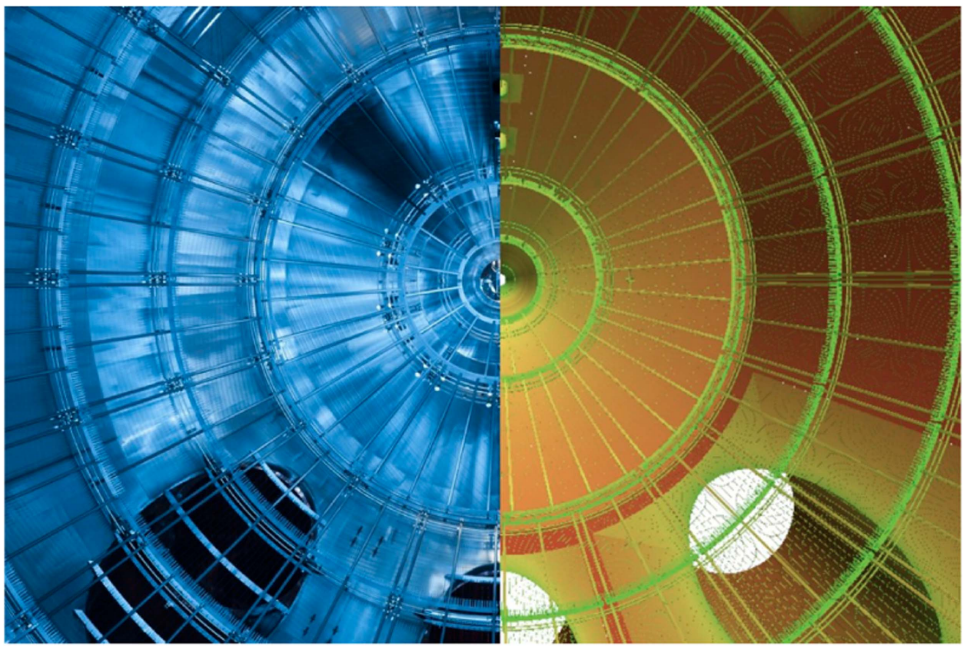
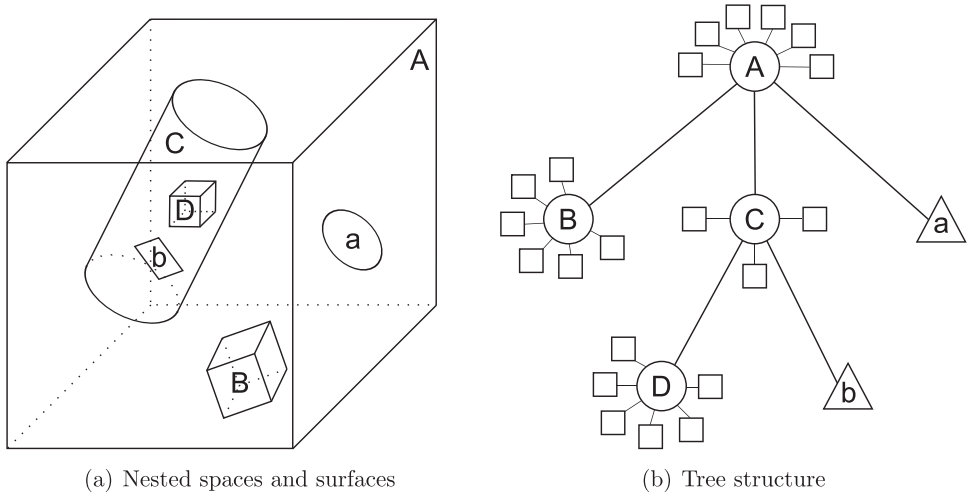


Figure 6. A photograph (left) and 3D electromagnetic model (right) of KATRIN's main spectrometer. Visible in both the model and the photograph are the pumping ports, the wire frames and mesh of the inner vessel.

When a space or surface element is placed inside another space, multiple transformations can be applied to rotate or displace the child space with respect to the coordinate system of the parent space. In the XML example in figure 5 the defined shapes are structured in a nested relationship and in figure 7 a visualization of this example is drawn together with its corresponding geometry tree. The tree is used to store the geometry relation internally. When referring to a specific space or surface (for example when adding an extension to it), its address in the tree is required, this can be specified using an XPath-like syntax as shown in figure 5.

3.3. Extensions

The shapes and surfaces of the geometry module feature an extension system which allows a user to append arbitrary information to an object. These extensions may contain different types of data for surfaces and spaces. This can be, for example, the color of a geometrical shape, to be used by a visualization module, or an extension with electromagnetic attributes such as the current in a space or the potential on a surface. The electromagnetic



(a) Nested spaces and surfaces (b) Tree structure

Figure 7. A set of nested spaces and surfaces (a) and its representation as a tree structure (b). In this example circles represent spaces, squares represent boundary surfaces, and triangles represent free surfaces. Figures from [15].

extensions are of prime importance for the field calculation module in order to compute electromagnetic fields for the given geometry. In the XML configuration of figure 5 an example of an electromagnet is given, where a current of 50 A is added as an extension of a cylinder object in order to form a solenoid coil.

4. Electromagnetic field computations

4.1. Magnetic field

For the magnetic field calculation of axisymmetric coils we use the zonal harmonic expansion method [17, 18]. This is a special version of the spherical harmonic method, applied for axisymmetric systems. It can be 100–1000 times faster than the more widely known elliptic integral method, and it is more general than the similar, but more widely known, radial series expansion [19–21]. It has not only high speed but also high accuracy, which makes the method particularly appropriate for trajectory calculations of charged particles. Due to these properties, no interpolation is necessary when computing the magnetic field during particle tracking.

The zonal harmonic expansions are convergent at field points within the central and remote regions, which have spherical boundaries. Their centers, (referred to as the source point), can be chosen arbitrarily along the axis of symmetry. The rate of convergence of the field series depends on the distance between the field evaluation point and the source point; the smaller distance for central field points, and conversely, the larger distance for remote field points, produces a correspondingly higher rate of convergence. The slower calculation requiring elliptic integrals can be avoided, unless the field point is very close or inside the coil windings. The zonal harmonic method can also be applicable for general three-dimensional coil systems, as long as the current distribution of each coil is axially symmetric within its own local coordinate system. See [17, 22] and [23] for more details.

The field of non-axisymmetric coils (e.g. coils which produce transverse dipole fields, which are rectangular, rather than solenoidal) are computed by directly integrating the Biot–Savart formula.

4.2. Electric field

In KEMField, the boundary element method (BEM) is used for static electric field computations. In the case of metal electrodes, the (known) boundary conditions given at an arbitrary point on the electrode surface may be expressed by a Coulomb integral over the unknown charge density of the whole surface of the electrode system. Thus an integral equation is obtained for the charge density function. To handle this problem numerically, the surface of the electrodes is discretized into many small boundary elements, and a system of linear equations is obtained, for which the charge densities of each mesh element serve as the unknown vector. To solve this system of equations, either a direct or an iterative method is used. Once the charge densities are known, the potential and field at an arbitrary point can then be computed by summing the potential and field contributions over all boundary elements. In KEMField, both metal surfaces (with Dirichlet boundary conditions) and also dielectrics (with Neumann boundary conditions) can be used for electric field computations [24].

The BEM has several advantages relative to the finite difference method (FDM) and the finite element method (FEM). First, there is no need to discretize the whole three-dimensional volume, only the two-

dimensional surfaces, thus the number of degrees-of-freedom is usually smaller. Second, for a given computation time, the BEM produces more accurate potential and field values than the FDM and FEM [25]. In addition, with the BEM, the potential and field at arbitrary points can be computed directly from the charged sources (rather than interpolated from values known at fixed points), which leads to very high accuracy and yields smooth field solutions. The BEM is also ideal for an electrode system with large differences in size scales (e.g. very small electrodes in a large-volume vessel) and can easily handle open systems, in contrast to the FDM and FEM.

In the special case of applying the BEM for axially symmetric electrodes, the potential-field contribution of a boundary element is usually evaluated with the help of the first and second complete elliptic integrals; often, a numerical integration of the elliptic integral formulas may also be necessary. To compute the potential and field of the whole electrode system, these contributions have to be summed over all elements. This summation can be rather time-consuming. Fortunately, it is possible to use the previously mentioned zonal harmonic expansion method within a large region of space of an axisymmetric electrode system [26], and this method is 2–3 orders of magnitude faster than the calculation using elliptic integrals.

In the case of axisymmetric electrode system, a few hundred or few thousand elements are usually enough to get an appropriate discretization. A direct method, like Gauss–Jordan elimination or LU decomposition, can be used in that case to compute the unknown charge densities of the elements. Direct methods can also be used for the special case of a discrete rotationally symmetric electrode system, where the number of elements could be high (e.g. few millions) but the number of different charge densities is still small (e.g. a few thousand).

In the more general case of a complicated three-dimensional electrode system, a few hundred thousand or few million elements might be necessary for a good discretization of the original surfaces. Typically, a three-dimensional electrode system is discretized by a combination of triangles, rectangles, or wire elements. These basic geometric elements approximate the charge density on the modeled surface using a piecewise constant function for the charge density. With large linear systems of this size, memory requirements render direct methods inapplicable and iterative methods become necessary. KEMField deals with large systems through the use of either the Robin Hood method, or Krylov subspace methods. The Robin Hood method, which is a special version of the Gauss–Seidel iteration, allows one to solve the linear system with a memory cost proportional to $\mathcal{O}(N)$ and a computational cost which scales like $\mathcal{O}(N^{\alpha})$ (with $1 < \alpha < 2$) [27]. On the other hand, Krylov subspace methods such as GMRES [28], if used with straightforward matrix-vector multiplication, would by themselves generally be insufficient to efficiently solve problems of this size. However, when combined with fast multipole techniques [29, 30] they are a powerful tool which can greatly reduce the time to solution and reduce memory cost. While a full description of the specific algorithm used by the KEMField library is beyond the scope of this paper, KEMField provides a modified multipole method which is a hybrid of the traditional fast multipole method (FMM) [31] and a Fourier transform based approach known as the fast Fourier transform on multipoles (FFTM) [32, 33], which is described in detail in [34]. Krylov subspace methods benefit greatly from preconditioning when dealing with the three-dimensional Laplace BEM and KEMField provides several simple choices such as Jacobi and Block–Jacobi, as well as an implicit preconditioner which acts by solving the same problem at reduced accuracy at each iteration in order to very effectively reduce the number of full accuracy iterations needed.

Once the charge densities are known, the potential and field of an element can then be computed directly though the use of either analytical or numerical integrations. Typically, analytical integration formulas are used to deal with the singular integrals when the evaluation point is close to the element, while Gaussian quadrature, which exhibits greater numerical stability, is used for field points which are far from the element (relative to the element size). Direct calculation of the potential and field by summing over all elements during charged particle tracking is very time-consuming. To reduce the computation time, one method that is provided is an interpolating field map grid. For a fixed electrode configuration, the potential and field values at the many grid points have to be evaluated only once, and during charge particle tracking the field calculation by interpolation is much faster than by the direct summation method. In order to increase the accuracy and to reduce the memory requirements, we use a cubic Hermite interpolation procedure [23, 35, 36]. This method is very effective when particle tracking is performed within small volumes which are well removed from boundaries, and fast and accurate field evaluation is needed. Another method which can sometimes require greater memory usage but can map the field to within a user-defined tolerance everywhere is provided by the FMM. This technique constructs a large collection of spherical multipole expansions (of the boundary elements in the far-field) covering the volume of interest, while near-field boundary elements have their field contributions evaluated directly. Several parameters exposed by this method, such as the expansion degree and spatial resolution, allow the user great flexibility to fine-tune a compromise between the accuracy, memory usage, and the speed of field evaluation at run time.

In order to make full exploitation of modern computing resources, KEMField has been designed to take advantage of parallelization whenever possible. In order to do this KEMField can make use of graphics

Table 1. Selection of value generators with description.

Value generator	Description
Fix	A fixed value defined by the user.
Uniform	The value is drawn from a uniform distribution between a defined minimum and maximum value.
Gauss	The value is drawn from a Gaussian distribution with a mean μ and standard deviation σ. Generated values may optionally be limited between a defined minimum and maximum value with the normalization adjusted accordingly.
Formula	The value is drawn between a minimum and maximum value according to a density distribution of a user-defined formula.
Set	A specific number of values is generated equally spaced between a start and an end value.
List	A list of values is used where each value in the list can be specified by the user.
Cylindrical	A position value is drawn uniformly from a cylindrical volume.
Spherical	A position or direction value is drawn uniformly from the surface of a sphere.

processing units (GPUs) using OpenCL [37] to accelerate both field calculation as well as the solving of BEM problems. Additionally, KEMField can be compiled with MPI [38], for use on distributed computing platforms making use of either CPU or CPU+GPU architectures.

5. Generation of particles

At the beginning of each event a particle generator needs to produce the initial state for a set of particles. In addition to the definition of a particle's intrinsic nature via its mass and charge, it needs to be fully characterized by seven parameters: position (x, y, z), momentum (p_x, p_y, p_z), and time (t). While the type of the particle and therefore its inherited properties can easily be specified by an ID number, following the PDG particle numbering pattern [39], the generation of its dynamic properties is broken up into a independent substructure consisting of four generators for the particle properties: time, position, energy, and direction. This choice of quantities was motivated by the particular use-case of KATRIN, which needs to examine particle motion in an MAC-E filter, for which it is generally advantageous to initialize a particle state by setting its energy and momentum direction, rather than setting the momentum vector directly.

For each of the four independent quantities (time, position, energy, and direction) the generated values can be set independently by specifying so-called value generators, which can drawn numbers from a user-specified distribution. Any combination of these value generators can be used to initialize the four basic quantities, leading to a large number of possibilities. A selection of value generators that are available within KASSIOPEIA is presented in table 1.

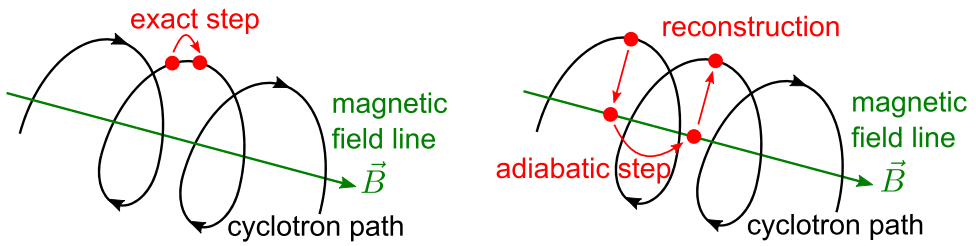
These value generators can be used separately or combined to form a composite generator for time, position, energy, and direction. In addition, there exist some special generators, which do not make use of the composite value mechanism. For example, these can be special position generators which can create random initial positions uniformly in a space or over a surface, or energy generators which reflect the radioactive decay sequences of specific unstable isotopes of radon, krypton or lead.

6. Propagation of particles

The propagation of particles, or more specifically the calculation of their corresponding trajectories, is one of the most important parts of the KASSIOPEIA software. This module is responsible for integrating the equations of motion which are represented as first order ordinary differential equations. Within KASSIOPEIA all continuous physics processes are represented as terms in the overall equation of motion. This includes not only the propagation of the particle in conservative force fields, but also processes such as radiative synchrotron losses, which can separately be included in the equation of motion. In this way, when numerically solving the particle motion, all of the included terms can treated together on equal footing. Depending on the choice of variables used for the full physical state of a particle, the terms present the equations of motion can adopt different representations. These are referred to as 'trajectories' in KASSIOPEIA. Each of the available trajectory types will be introduced in the following subsections, along with the specific differential equations for each physics process. We will also outline the integrators used to solve the differential equations in conjunction with the step size controls required to mitigate numerical error.

6.1. Exact trajectory

When dealing with an exact trajectory, the physical state of the propagating particle is described by a pair of variables which are implicitly a function of the time coordinate t. These are the position vector, $\mathbf{r}$, and the



(a) Exact trajectory (b) Adiabatic trajectory

Figure 8. Illustration of the step taken when computing the exact trajectory (a) and the adiabatic trajectory (b). In the adiabatic trajectory calculation the guiding center position is propagated, which allows a much larger step size. The particle's exact position can be reconstructed afterwards if the gyration is included. Figures from [15].

momentum vector, $\mathbf{p}$. The representation of the propagation term in the equation of motions of a particle with charge q in an electric and magnetic field $\mathbf{E}$ and $\mathbf{B}$ is given by the Lorentz equation. The terms for the ordinary differential equations for the variables of the exact representation are therefore:

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \frac{\mathbf{p}}{\gamma m}, \tag{2}$$

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = q\left(\mathbf{E} + \frac{\mathbf{p} \times \mathbf{B}}{\gamma m}\right), \tag{3}$$

where m is the rest mass of the particle, and γ is the relativistic Lorentz factor.

6.2. Adiabatic trajectory

If the magnetic and electric fields are nearly constant within a cyclotron radius of the particle, the first adiabatic invariant $\gamma\mu$, with μ being the magnetic moment, remains conserved along the trajectory of the particle. Under this approximation, the physical state of the particle can be represented by its time, t, the guiding center of the motion, $\mathbf{r}_\mathrm{c}$, the components of the particle's momentum which are parallel and perpendicular to the magnetic field $p_\parallel$ and $p_\perp$, and the gyration phase ϕ [14]. The advantage of using this adiabatic trajectory is the much larger step size that is possible compared to the case of calculating an exact trajectory, since the curvature in the path of the guiding center position is much smaller than in the propagation of the real particle position. The exact position of the particle can be reconstructed from the guiding center after the propagation step, as visualized in figure 8.

In case of the adiabatic representation, the propagation terms in the ordinary differential equation assume the following form:

$$\frac{\mathrm{d}\mathbf{r}_c}{\mathrm{d}t} = \frac{p_\parallel}{m\gamma}\frac{\mathbf{B}_\mathrm{c}}{B_\mathrm{c}}, \tag{4}$$

$$\frac{\mathrm{d}p_\parallel}{\mathrm{d}t} = -\frac{p_\perp^2}{2\gamma m B_\mathrm{c}}\nabla B_\mathrm{c} + q\mathbf{E}_\mathrm{c}\cdot\frac{\mathbf{B}_\mathrm{c}}{B_\mathrm{c}}, \tag{5}$$

$$\frac{\mathrm{d}p_\perp}{\mathrm{d}t} = \frac{p_\perp p_\parallel}{2\gamma m B_\mathrm{c}}\nabla B_\mathrm{c}\cdot\frac{\mathbf{B}_\mathrm{c}}{B_\mathrm{c}}. \tag{6}$$

Two additional terms need to be added to these propagation terms to account for gyration and drift caused by the magnetron motion. The gyration term can be derived from the cyclotron frequency of the particle, and since it completely decouples from the guiding center motion is simply

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = \frac{qB_\mathrm{c}}{m\gamma}. \tag{7}$$

The terms accounting for the drift motion are more complicated as they modify both guiding center position as well as the momentum components, their contributions can be written as

$$\left.\frac{\mathrm{d}\mathbf{r}_c}{\mathrm{d}t}\right|_{\mathrm{drift}} = \frac{\mathbf{E}_c \times \mathbf{B}_c}{B_c^2} + \frac{2p_\parallel^2 + p_\perp^2}{qm(\gamma+1)B_c^3}\mathbf{B}_c \times \nabla B_c \frac{\mathbf{B}_c}{B_c}, \tag{8}$$

$$\left.\frac{\mathrm{d}p_\parallel}{\mathrm{d}t}\right|_{\mathrm{drift}} = \frac{q\gamma m}{p_\parallel}\mathbf{E}_c \cdot \frac{\mathrm{d}\mathbf{r}_c}{\mathrm{d}t} - \frac{p_\perp^2}{2p_\parallel B_c}\left(\nabla \mathbf{B}_c \cdot \frac{\mathrm{d}\mathbf{r}_c}{\mathrm{d}t}\right) \cdot \frac{\mathbf{B}_c}{B_c}, \tag{9}$$

$$\left.\frac{\mathrm{d}p_\perp}{\mathrm{d}t}\right|_{\mathrm{drift}} = \frac{p_\perp}{2B_c}\left(\nabla \mathbf{B}_c \cdot \frac{\mathrm{d}\mathbf{r}_c}{\mathrm{d}t}\right) \cdot \frac{\mathbf{B}_c}{B_c}. \tag{10}$$

6.3. Magnetic trajectory

An additional trajectory type within KASSIOPEIA is the 'magnetic trajectory' which can be used to visualize magnetic field lines. This is achieved by creating a pseudo-particle which is represented only by its time t and position $\mathbf{r}$. Particle properties such as kinetic energy or momentum are undefined. In this regard, it is interesting to note that in the context of KATRIN, electrons with a kinetic energy in the kilo-electronvolt regime posses relatively small cyclotron radii. Therefore, on occasion, a magnetic trajectory can be a good approximation for the path of such an electron. This simplification can be advantageous since a magnetic trajectory can be calculated with considerably less computational effort than that which is required to track a charged particle, either exactly or adiabatically. The ordinary differential equation in this case is simply

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \frac{\mathbf{B}}{B}. \tag{11}$$

6.4. Synchrotron losses

Besides the previously mentioned terms for the propagation of the particle, other terms can be added as well, such as the synchrotron term, which handles free-space radiative losses by accelerated particles. This radiated power needs to be expressed as a force term for use in trajectory calculations. The well known expression for this is the Abraham–Lorentz–Dirac radiation reaction force, which suffers from several pathologies such as violation of energy conservation and causality, as pointed out by Dirac [40]. Within KASSIOPEIA a comparatively new approach is implemented that avoids these problems and follows the equation of motion proposed by Ford and O'Connell [41], which reads as

$$m\frac{\mathrm{d}(\gamma \mathbf{v})}{\mathrm{d}t} = \mathbf{F} + \tau_e\left[\gamma\frac{\mathrm{d}\mathbf{F}}{\mathrm{d}t} - \frac{\gamma^3}{c^2}\left(\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} \times (\mathbf{v} \times \mathbf{F})\right)\right], \tag{12}$$

where $\tau_e = 2e^2/3m_e c^3$ and $\mathbf{F}$ is the known Lorentz force,

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}), \tag{13}$$

while the second term is the radiation reaction force. This is the force term that is used in KASSIOPEIA for the exact trajectory. Notice that the total time derivative of the fields requires us to calculate the field gradient and these are in general very costly to evaluate. However, the term accounting for radiative losses in the adiabatic representation is simpler and faster, and the full derivation can be found in [42]. The radiated power can be described in terms of external forces acting on the particle as follows:

$$P = \frac{\mu_0}{6\pi c}\frac{q^2}{m^2}(f_\parallel^2 + \gamma^2 f_\perp^2), \tag{14}$$

where $f_\parallel$ is the component of the force acting tangential to the particle's motion and $f_\perp$ acts normal to its motion. Equation (14) needs to treated approximately by making the assumption that the motion of the gyration is completely responsible for the radiation. This assumption is completely consistent with the adiabatic approximation, and yields the complete synchrotron term in this representation:

$$\left.\frac{\mathrm{d}\mathbf{r}_c}{\mathrm{d}t}\right|_{\mathrm{sync}} = 0, \tag{15}$$

$$\left.\frac{\mathrm{d}\phi_c}{\mathrm{d}t}\right|_{\mathrm{sync}} = 0, \tag{16}$$

$$\left.\frac{\mathrm{d}p_\perp}{\mathrm{d}t}\right|_{\mathrm{sync}} = -\frac{\mu_0}{6\pi c}\frac{q^4}{m^3}\,|\mathbf{B}(\mathbf{r}_c, t)|^2 p_\perp \gamma, \tag{17}$$

Table 2. List of supplied Runge–Kutta integrators.

Name	Solution order	Function evaluations per step	Embedded local error estimate	Dense output order	Reference
RKF54	5	6	Yes	NA	[43]
RKDP54	4	7	Yes	5	[44, 45]
RK65	6	8	Yes	NA	[46]
RK8	8	13	No	NA	[47]
RK86	8	12	Yes	NA	[48]
RK87	8	13	Yes	NA	[46]
RKDP853	8	16	Yes	7	[45]
SYM4[a]	4	4	No	NA	[49]

[a] This integrator can only be used with the exact trajectory.

$$\left.\frac{\mathrm{d}p_{\parallel}}{\mathrm{d}t}\right|_{\mathrm{sync}} = 0. \tag{18}$$

This description is free of field gradients and as such can only be an approximation. However, the adiabatic approximation to hold, the gradients need to be negligible anyway and therefore no new assumption is made here.

6.5. Integrators

All of the mentioned trajectories can be written as ordinary differential equations of the form $y' = f(x, y)$. These differential equations have to be integrated numerically. The numerical integration is carried out by integrators of the Runge–Kutta type. KASSIOPEIA provides a variety of Runge–Kutta integrators which may be chosen by the user depending on their preference for accuracy or speed. Table 2 lists the available integrator types and some of their properties. These integrators can be categorized by several properties. These are: the order of truncation error on the solution as a function of the step size; the number of function evaluations required to propagate the solution at each step; and the availability of a local error estimate. Since straightforward Runge–Kutta routines typically do not provide information about the solution between steps, some integrators have been equipped with a continuous interpolant (dense output). All Runge–Kutta integrator routines which do not provide an embedded dense output can optionally be equipped with a first or third order Hermite interpolant for dense output which requires no additional function evaluations. For problems where extremely long tracking times or excellent energy conservation is required, a symplectic integrator is also provided, as detailed in table 2. Since KASSIOPEIA is object-oriented and easily extensible, users may also chose to add their own integrators using alternative Runge–Kutta, predictor–corrector, or symplectic algorithms. Any additional integrators can be accommodated as long as they originate from from the KSMathIntegrator interface.

6.6. Controls

The computation of a step is critical in terms of performance. Since it usually requires the calculation of the electromagnetic field at multiple positions, the number of calculated steps for each track should be limited as much as possible. This must be done carefully, as the accuracy in calculating the particle's motion will decrease for larger step sizes, which can lead to a violation of energy conservation. The step size of the particle therefore needs to be adjusted to meet the specific requirements of the user. This customization is handled by step size controls, which are part of the tracking module of KASSIOPEIA. A step size control suggests a specific step size (with the dimension of time) to the integrator at the beginning of each step. After the step has been performed the step size control may accept or reject the current step, and suggest a new step size. It is also possible to use multiple step size controls simultaneously with the smallest suggestion being used. The simplest step size controls within KASSIOPEIA are user-defined constraints on a maximum time step or a maximum step length. More advanced step size controls are also available. For regions where a magnetic field is present the step size can be chosen based on a fixed fraction of the current cyclotron period of the particle, this control results in small steps in high magnetic fields and large steps in low ones. It also possible to adjust the step size dynamically to keep the violation of conserved quantities, such as the total energy, within a user-defined range. Additionally, dynamic step size adjustment is also available if a Runge–Kutta integrator with embedded error estimation is used. In this case step size can be adjusted to keep the local error on the position and/or momentum of the particle within a range of user-defined absolute or relative tolerances.

7. Interaction of particles

The interaction processes within KASSIOPEIA are grouped into space interactions and surface interactions. Space interactions occur stochastically as a function of a given probability during particle propagation within a dense medium, while surface interactions are triggered only when the particle reaches a surface which has a specific interaction attached to it.

7.1. Space interactions

For particles moving with a velocity v through a medium with a target number density of n, the probability for an interaction with cross section of σ to occur after time t can be calculated according to:

$$P(t) = 1 - \exp\left(-\frac{t \cdot v}{\lambda}\right), \tag{19}$$

where the mean free path is defined as

$$\lambda = \frac{1}{n \cdot \sigma}. \tag{20}$$

In KASSIOPEIA the parameters n, v, and σ are calculated as mean values for the initial and final position of the particle on the step. The density of the medium is calculated by a separate density module. The simplest example being defined by a constant density, which may be related to properties specified by user such as partial gas pressure and temperature. The cross section σ corresponds to the sum of all individual cross sections of interaction processes for this scattering module. The time for the next scattering to take place can be generated stochastically according to:

$$t_{\text{scat}} = -\ln(1 - P) \cdot \frac{\lambda}{v}, \tag{21}$$

where P is drawn from a uniform distribution between 0 and 1. If this scattering time is larger than the time it takes the particle to complete the current step, no scattering will take place. On the other hand, if $t_{\text{scat}} < t_{\text{step}}$, the trajectory of the particle is recalculated with a step size identical to the scattering time so that the properties of the particle are exactly computed at the time just before scattering takes place. After that, a decision on the specific scattering process to be executed (typically elastic or inelastic scattering), is made based on their individual cross section contributions. Finally, the scattering process is executed, thereby modifying the properties of the particle and optionally creating new particles.

7.2. Scattering types

Scattering processes are treated by the simulation in a modular way so that each process can be handled individually as a so-called scattering calculator. This unit is responsible for calculating the cross section as a function of the particle's state, as well as executing the interaction process which modifies the particle. Multiple calculators with the same species can then be grouped into a scattering module.

The dominant gas species in KATRIN's main spectrometer is molecular hydrogen, while in the source region it is molecular tritium, with the scattering properties of both isotopes being very similar. The corresponding cross sections, energy losses and angular changes for elastic [50–52], excitation [53, 54], and ionization processes [55] are implemented within different scattering calculators, one for each process.

Additional scattering processes for different species can be implemented easily. For example, this can be done by importing data from the LXcat database [56], which has already been accomplished for the process of electron scattering off argon [57].

In addition, specific scattering calculators to describing electron interactions in silicon are available. These were adapted from the KESS package [58] and now are completely integrated into KASSIOPEIA.

7.3. Surface interactions

Surface interactions differ from space interactions as they are only enforced when the particle crosses a specific surface. The associated surface interaction has to decide whether the particle is transmitted to the next space or reflected back into the previously occupied space. This will modify the properties of the particle accordingly, and can result in an angular or energy change when crossing the interface between different materials. Of particular importance to KATRIN are processes where particles enter from vacuum into a solid silicon detector.

8. Termination of particles

The trajectory of a particle has to be terminated once a specific condition from a user-defined set is met. The condition to terminate the propagation of the actual particle can be defined in a very flexible way. For example,

Table 3. Selection of terminators with description.

Terminator name	Description
min/max z	upper/lower bound on z component of position
min/max r	upper/lower bound on r component of position
min/max kinetic energy	upper/lower bound on kinetic energy
max time	upper bound on track time
max length	upper bound on track length
max steps	upper bound on track step count
death	stops track if this terminator is active
secondaries	stops track if particle is a secondary
min distance	lower bound on distance to specific space or surface
trapped	upper bound on longitudinal momentum sign change count
output	allowed range for the value of a specific output variable

the termination condition can be determined to occur when the particle hits a detector after having propagated through an experiment, or it may be made by identifying a specific particle property that makes further tracking meaningless, such as when a certain minimal kinetic energy is reached, or if an undesired propagation direction of the particle manifests. A selection of terminators available within KASSIOPEIA are presented in table 3.

As a terminator module is a small class with a simple structure, additional terminators as required by the user can easily be added. Like all other modules, terminators can be attached to specific navigation spaces or surfaces of the simulation and will only be activated once the particle enters that specified geometry.

9. Output

The output that will be written to disk for a given simulation depends on the users needs. A static output system that writes down fixed particle properties after each track or even step is therefore not desirable. This becomes evident when running Monte Carlo simulations with millions of tracks, each track containing millions of steps, as the disk space required to save the simulated information can easily reach problematic dimensions.

Therefore, the output system for the recent version of KASSIOPEIA was designed in a highly flexible way that allows the user to define each individual output component in the XML configuration for the four levels of detail: run, event, track, and step. Therefore, it is possible to store exactly the amount of information that is required, and the output can be made to include very specific information which is desired from relevant objects in the simulation.

9.1. Output components

A single output component can be configured through a chain-like system which starts with an object that has been put into the simulation toolbox. By calling a getter function of the simulation object and repeating the procedure for the resulting object, a chain can be produced with the desired output variable at the end. For example, if a user wants to write down the magnitude of the magnetic field at the end of each step. This can be done by retrieving the step object from the toolbox, getting the final particle object from the step, getting the magnetic field vector from the particle, and finally, calculating the magnitude of the magnetic field vector.

Multiple output components with the desired information about the simulation need to be grouped together before they can be added to the object writing the output. Each group corresponds to a tree in the ROOT [59] data structure, which is written to disk en bloc. When adding an output group to the writer object, the user can choose the level at which this group should be written, i.e. each step, track, event, or run. Output groups at the step level can be connected to the navigation geometry as well as any other simulation module, and will only be turned-on when the particle is in an active region. A typical step output group might consist of time, position, energy, and electromagnetic field values at each step, while a typical track output group may be composed of initial and final positions, energies of tracks, and information about the generation or the termination process.

9.2. Analysis logic

In addition to the flexibility of the output system, where the user can define exactly which variable at which interval or geometrical state should be recorded, it is also possible to apply a simple analysis logic to the output stream. This is highly useful, if for example, a user is only interested in the maximal kinetic energy of each track, but does not want to record the energy information for each step. The alternative of storing all this information and performing the analysis after the Monte Carlo is finished would increase the required disk space significantly.

The available analysis logic for output components within KASSIOPEIA allows for the determination of a minimal, maximal, or integral value. These output components can be used at any level, but the resulting output should be collected at least one level higher than the component it depends. For example, the maximal value of a variable which is updated at step level should only be recorded at the track level or higher. Additionally, there is a math output component, which can combine arbitrary output components at the same level with a user-defined function.

9.3. Writers and readers

Apart from a simple ASCII writer, the main writer for KASSIOPEIA is based on the binary data format of a ROOT [59] file with a tree structure. For each selected group of output components of the four organization levels of run, event, track, and step, a tree is created with the data objects as branches. Additional meta-information is also stored which permits the correct reconstruction of the data. For a simple analysis of the data, the files can be viewed and plotted with a few clicks using the ROOT TBrowser. For a more advanced analysis, the user has the possibility to write their own analysis tools by linking against the KASSIOPEIA package. With the provided reader classes, saved data can be reconstructed automatically for user-friendly access. Additionally, geometry data and simulation data such as steps and tracks can also be written using the VTK format [60], which enables the creation of three-dimensional visualizations of geometries and tracks, as detailed in section 11.

10. Navigation

The task of the navigation module is to make sure that the simulation algorithm always follows the state defined by the current geometry object which houses the particle, as defined by the user in the configuration file. At the beginning of each track the navigator needs to check the location of the initial particle position and adjust the simulation algorithm accordingly. After each step the navigator checks if the current space was exited, a child space was entered, or a surface was crossed. This is done by calculating the distance to all relevant navigation geometries. As a caching system is used, these distances are not computed at each step, but only if the length of the particle's trajectory exceeds the cached limit. If a crossing of any navigation boundary is detected, the position of the final particle is adjusted to the exact geometrical position of the intersected surface.

If this surface is associated with a change of the simulation algorithm, the corresponding commands are executed and the next step of the particle is a surface step, meaning no propagation takes places and only the interaction associated with this surface is executed. Afterwards, the navigator checks if the particle has been reflected or transmitted, depending on the result of the surface interaction. Subsequently, the appropriate commands of the space associated with the particle's next step are executed.

If the crossed surface is part of a space change, without (surface) commands attached to it, the navigator will adjust the simulation algorithm according to the corresponding commands. If a space was exited, the inverse of the commands attached to this space are executed, and if a new space is entered, the commands attached to this navigation space are applied. When entering a nested space, the particle still remains inside the encompassing outer space, therefore the inverse commands associated with an exit condition are not executed. This ability to nest spaces requires that the user ensure the set of commands that are applied when entering a space are defined in a way which respects the spatial hierarchy (to avoid repetition or the removal of objects which are no longer active).

The change of the simulation algorithm's state is achieved by commands, which can be used to modify the root classes, as detailed in section 2. The definition of these commands in the XML configuration file is shown in the following example of figure 9, where a simulation is set up in a world space with a cylindrical inner volume. When the particle enters this inner volume, the propagation calculation should switch from the adiabatic to the exact method, the step output should be activated, and the particle should be terminated if it reaches the bottom surface.

The example shown consists of two navigation spaces and a navigation side. The main navigation space is the world space, which possesses an attached command which activates the adiabatic trajectory within the root trajectory object, and several more commands which add terminators to the root terminator. The world space also contains an inner volume which is of special interest to the user in the given example. In this volume, the trajectory is replaced by first defining commands for removing the old adiabatic trajectory and subsequently adding the exact trajectory to the root trajectory object. An additional terminator is added within this new space and also the step output is activated by adding an output group to the root writer. The particle should be terminated when reaching the bottom surface of the inner volume. As this particular surface is part of a navigation space it is referred to as navigation side, in this case a command is attached to it which adds a death terminator, which kills the particle track.

```
<ksgeo_space name="space_world" spaces="world">
    <!-- set trajectory -->
    <set_trajectory child="trajectory_adiabatic"/>
    <!-- set terminators -->
    <add_terminator child="term_min_z"/>
    <add_terminator child="term_max_z"/>
    <!-- define which modules should be used in the inner volume -->
    <geo_space name="inner_volume" spaces="world/inner_volume">
        <!-- replace adiabatic trajectory with exact trajectory -->
        <clear_trajectory child="trajectory_adiabatic"/>
        <set_trajectory child="trajectory_exact"/>
        <!-- add another terminator -->
        <add_terminator child="term_min_energy"/>
        <!-- add step output -->
        <add_step_output child="output_step_basics"/>
        <!-- define modules on the bottom surface of the inner volume -->
        <geo_side name="bottom_surface" surfaces="world/inner_volume/bottom">
            <add_terminator child="term_death"/>
        </geo_side>
    </geo_space>
</ksgeo_space>
```

Figure 9. Example for a navigation definition in an XML file. The shown example is explained in full detail in the text.

The available command pattern is not limited to the modification of the root classes only, but can be used to modify any object registered in the toolbox. For example, changing the step size control of a trajectory object or adding a scattering calculator to a scattering module is easily accomplished in this paradigm.

11. Visualization

In order to aid the user in reducing and understanding the Monte Carlo data produced by KASSIOPEIA, two visualization options are available for generating graphical output: the three-dimensional VTK visualization [60] and the two-dimensional ROOT [59] visualization. These two options each depend on their corresponding external libraries. Both visualization options are completely configurable in the XML configuration file, and some options include the ability to color track or step elements by dynamic variables, change the viewing angle, and select and plot geometric objects.

The first depends on the use of the external library VTK [60]. As detailed in section 9 various data components at track and step scope can be saved to a VTK polydata file (.vtp). The user may then process the polydata files using the visualization tool of their choice. In addition, data concerning the geometry, such as the size and aspect ratio of BEM mesh elements, can also be exported and visualized in this way. Figure 10 demonstrates some of the capabilities of the KASSIOPEIA package and the use of the VTK output format in conjunction with the visualization software Paraview. In this example the electron tracks are colored according to their kinetic energy while the photomultiplier tube (PMT) surfaces (consisting of roughly 150 K triangular elements) are colored according to their electrostatic potential. The PMT simulation depicted in figure 10, has been performed primarily as a demonstration of the visualization, field solving, and particle navigation capabilities of KASSIOPEIA, it thus uses a simple toy model for the secondary electron production. This simplified model draws the number of secondary electrons from a Poisson distribution, and after some fractional energy loss, randomly partitions the primary particle's energy among the secondaries and ejects them each at a randomly chosen angle. While this particular model of the surface physics may not be an accurate representation, even such simple models can sometimes provide useful information. In this case, to provide an example to the reader, we speculate that the relative gain variation of a PMT according to the strength of some stray magnetic field could perhaps be modeled. It is important to note that, while the existing interactions models in KASSIOPEIA cannot handle every possible end use, it is a fairly simple matter for the end-user with knowledge (e.g. scattering cross sections) of the physical process that they wish to model to insert any discrete surface or volume interaction into the KASSIOPEIA framework.

The second visualization option depends on the use of the external library ROOT [59], which it uses for rendering and display. This is typically used to produced a two-dimensional cross section of a selected geometry where particle tracks can be plotted and colored according to the user's preferences. In the example of figure 11, the particle tracks are colored according to the pitch angle of the electron with respect to local magnetic field. While visually less impressive than the three-dimensional option afforded by VTK, this type of visualization can be very useful for detailed geometric investigations and is simple to use.

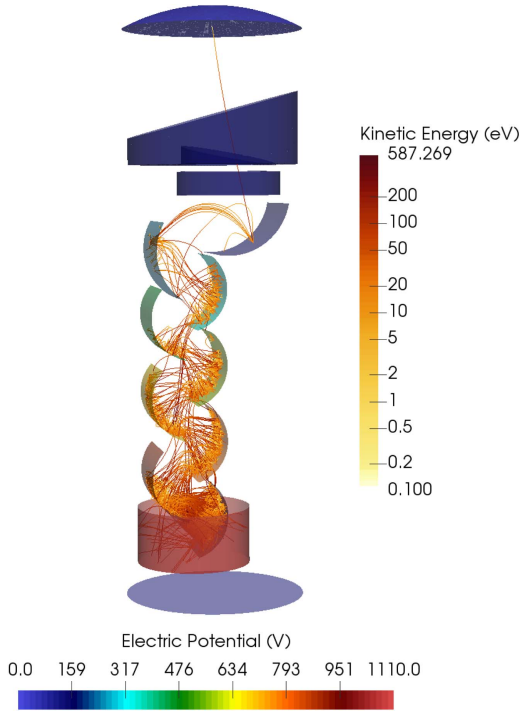


Figure 10. A series of electron tracks generated from a single photo-electron event in a simplified PMT demonstration. The outer wall of the PMT has been removed for visualization purposes. The field calculation and tracking in this example are accurate, the interaction model for the electron scattering is simplified. The image was rendered using Paraview [61].

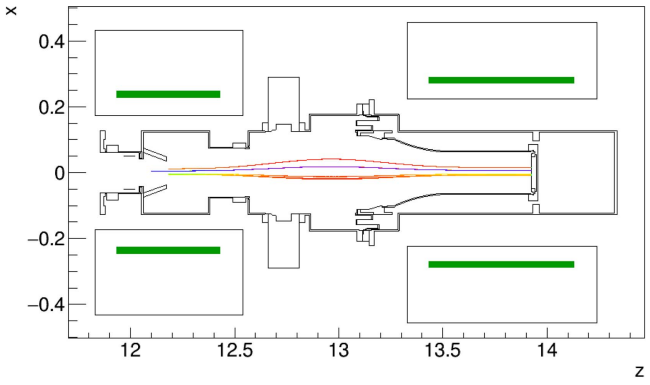


Figure 11. Cross section of the KATRIN Detector system with electrons simulated as entering from the left and terminating on the detector at right. The superconducting coils are shown in green providing a guiding magnetic field. The trumpet-shaped post acceleration electrode sits immediately before the detector and is held on +10 kV potential.

12. Validation and use

Two very important issues of any scientific software are generality and correctness. Therefore it is of great importance to provide example use-cases and validate the results that KASSIOPEIA produces. One very basic feature of the KASSIOPEIA development cycle is the existence of a set of unit and integration tests that check the basic functionality of separate and combined parts of the software. This tool aids in the continual development and simultaneous use of the software and is critical to keep the software in working order.

To establish the utility of KASSIOPEIA we provide a list of published real-world applications as follows:

- 'Radon induced background processes in the KATRIN pre-spectrometer' [62]: KASSIOPEIA tracking simulations are used.
- 'Background due to stored electrons following nuclear decays at the KATRIN experiment and its impact on the neutrino mass sensitivity' [63]: KASSIOPEIA is used for stored particle tracking with interactions.

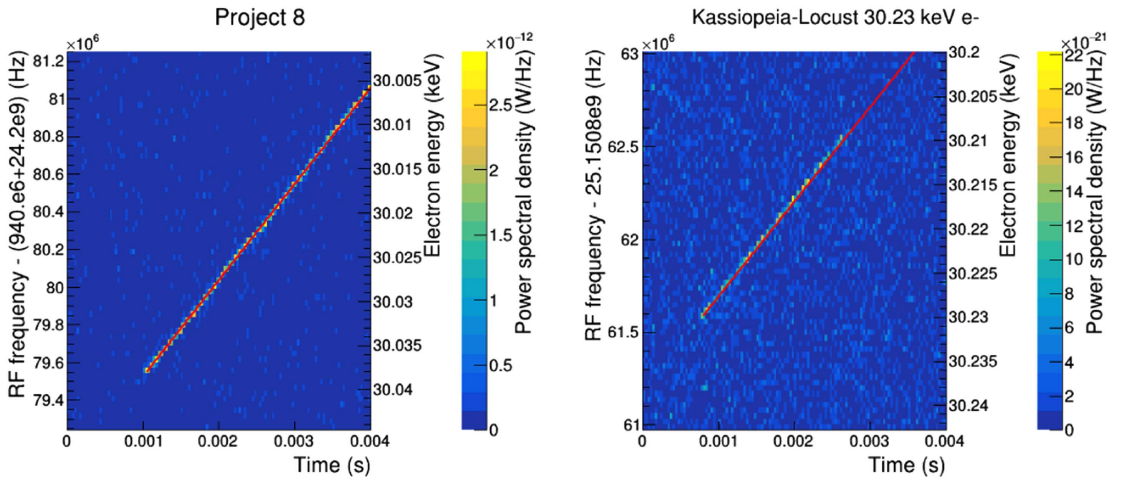


Figure 12. Side by side comparison showing cyclotron radiation from a single electron measured in the Project8 experiment (left) and that from a 30.23 keV electron simulated in KASSIOPEIA (right). The solid red lines denote the cyclotron frequency expected from equation (22). The difference of eight orders of magnitude in the absolute power between the left and right panels derives from the gain of the receiver, which is present in the experiment but is not necessary in the simulation. The power in the simulated noise background is 10×10^{-22} W/Hz. In the left panel the noise background at the input to the receiver is expected to be lower than the above after replacing a termination with a reflecting short in the experiment, resulting in the higher signal to noise ratio in the left panel. The plots demonstrate that the energy losses to cyclotron radiation calculated by KASSIOPEIA are consistent with those observed in an empirical measurement.

- 'Stochastic heating by ECR as a novel means of background reduction in the KATRIN spectrometers' [64]: KASSIOPEIA is used for stored particle tracking with interactions.
- 'The KATRIN pre-spectrometer at reduced filter energy' [65]: KASSIOPEIA is used for tracking simulations.
- 'Electromagnetic design of the large-volume air-coil system of the KATRIN experiment' [66]: KASSIOPEIA is used for magnetic field calculation and field line tracking.

In addition to the aforementioned studies, there are also several examples that provide validation through a direct comparison of the output of a KASSIOPEIA simulation with actual experimental data. These include:

- 'Validation of a model for radon-induced background processes in electrostatic spectrometers' [67]: Stored particle tracking with interactions.
- 'Modeling of electron emission processes accompanying radon-α-decays within electrostatic spectrometers' [68]: validation of a radon generator.
- 'High voltage monitoring with a solenoid retarding spectrometer of the KATRIN experiment' [69]: magnetic field calculation and field line tracking.
- 'Investigation of the passage of electrons from vacuum into the active volume of a p–i–n diode charged particle detector' [70]: electron tracking and diffusion in silicon.
- Project8, an experiment trying to measure the neutrino mass by taking tritium β spectra through detection of cyclotron radiation, relies on KASSIOPEIA to model the radiative losses by electrons in a magnetic trap.

One particularly interesting comparison between KASSIOPEIA and experimental data is provided by the Project8 experiment [71], the results of which are shown in figure 12. In this example a magnetic field of approximately 1.0 T is modeled in KASSIOPEIA. A trapping region is implemented by distorting the main field by –40 G in a parabolic shape that is similar to that used by the Project8 experiment. Electrons with pitch angles, ϕ, where $|\phi - 90°| < 6°$, are trapped magnetically for up to 1 ms. Their tracks are calculated in KASSIOPEIA using adiabatic trajectory integrated by an eighth order Runge–Kutta method with a step size limited to 1.125 cyclotron periods. The electrons move back and forth in the trap at a frequency near 100 MHz while circling the field lines in cyclotron motion, loosing energy due to cyclotron radiation. Further demonstrating the flexibility of KASSIOPEIA and the ease by which it may be extended, the simulated energy losses calculated by KASSIOPEIA are sampled at 2 GHz and passed to an external software package called Locust that simulates the voltages induced in an antenna and receiver.

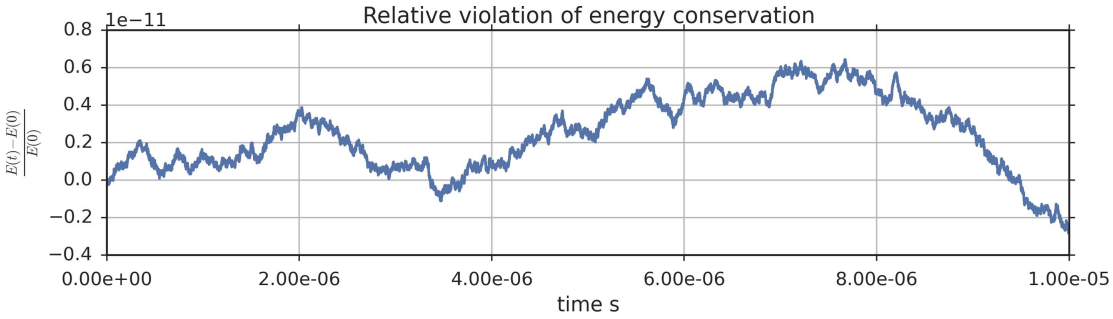


Figure 13. The relative violation of energy conservation for an electron stored over ≈10 μs. This corresponds to roughly 1.7 million cyclotron periods in a perfect static quadrupole trap with a magnetic field of 1 T and a voltage of 10 V. This calculation was performed using the exact tracking method with a step size limited to less than 10^{-11}s. A full description is shipped with KASSIOPEIA in the corresponding configuration file QuadrupoleTrapSimulation.xml, which enables any user to re-run this reference simulation.

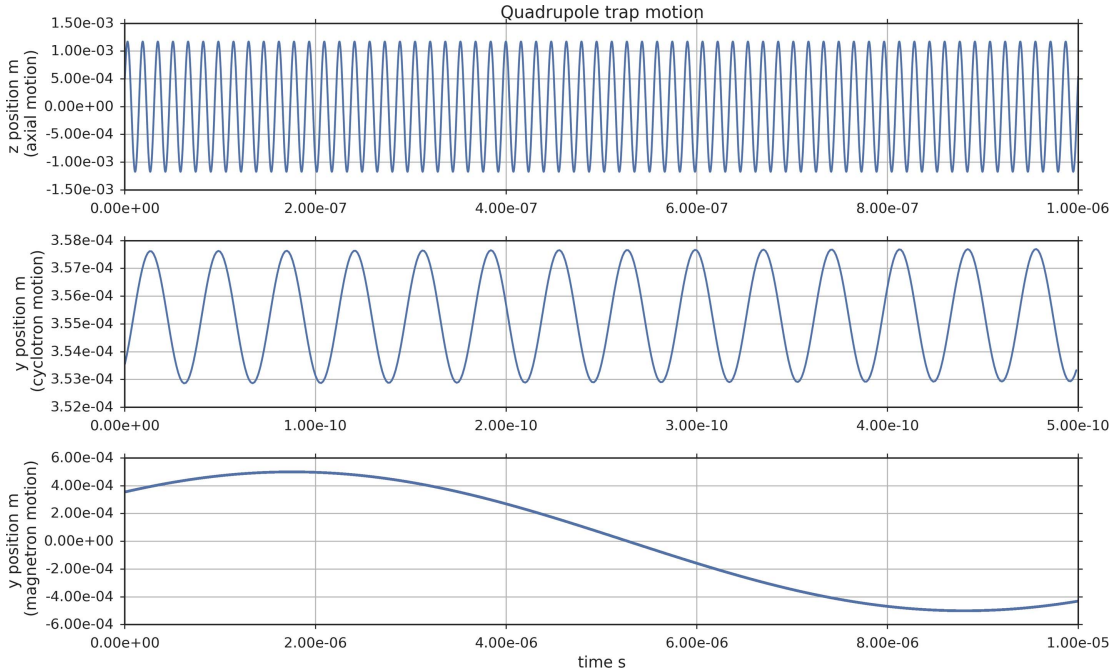


Figure 14. From the motion of the electron in the trap the frequencies are derived by calculating the mean time between two zero-crossings. For the magnetron motion $y = a_m \cdot \cos(\omega_m \cdot t + \phi_m)$ was fitted to the data and afterwards subtracted to allow for proper detection of zero-crossings for the cyclotron motion. The top plot displays the motion in the z-direction, below that the cyclotron motion in the y-component, and at the bottom the magnetron motion is shown. The data are generated by the same QuadrupoleTrapSimulation.xml as in figure 13.

The right panel of figure 12 shows the power from a simulated 30.23 keV electron located at the magnetic trap minimum with pitch angle 90°. The slope of the track is in agreement with the corresponding empirical measurement from the Project8 experiment in the left panel to about 1%, confirming the calculations of energy loss. The starting frequencies of the tracks agree to within 200 eV, which is consistent with absolute uncertainty on the field at the magnetic minimum of the trap. The difference in magnetic fields is also responsible for the 1% deviation of simulated and measured slope in the event. The solid red lines in figure 12 denote the cyclotron frequency predicted from the ideal analytic expression

$$f_\gamma = \frac{eB}{2\pi\gamma m_e}, \tag{22}$$

where e and m_e are the charge and mass of the electron, γ corresponds initially to 30.23 keV, and B is 0.9583 T minus 39.97 G (42.68 G) in the left (right) panel. At $t = 1$ ms, the electron starts to lose energy to cyclotron radiation with approximate power

$$P = \frac{2}{3}\frac{e^2}{c^3}|\dot{v}|^2\gamma^4. \tag{23}$$

The rate of the energy loss determines the slope of the line and is in good agreement with both the simulation and empirical data in 12.

Complementary to the comparison to real-world data, some simple physical test cases that have analytical reference solutions have been incorporated into several stand-alone binaries. For example, charge density calculations can be performed on several simple geometries such as a spherical or unit-cube capacitor, which shown the field solvers are both accurate and scalable. For additional results discussing the validity and accuracy of the field computation the reader is referred to [24] and [34].

In addition, to test a complete case involving particle tracking, the simulation of a (static) quadrupole Penning trap has been performed. This test case was chosen for several reasons. It exhibits long storage times of charged particles, which is a prominent use-case for KASSIOPEIA and allows one to evaluate its performance in this task. Its fields and also the trajectories of stored particles are well described analytically, which makes it possible to evaluate tracking and field computation independently against known solutions. On the other hand this system is not trivial, in real-world applications it can be affected by electric and magnetic field disturbances that may be further explored in simulation. For this particular example, a plot showing the degree of energy conservation violation is shown in figure 13.

Similar to the law of energy conservation, another invariance theorem is very suitable to assess the numerical errors in this particular system. The Brown–Gabrielse theorem [72] states

$$\omega_m^2 + \omega_z^2 + \omega_{c'}^2 = \omega_c^2. \tag{24}$$

It connects the frequencies observed in a quadrupole trap to the free cyclotron frequency $\omega_c = \frac{e}{m_e\gamma}B$, where ω_m is the magnetron frequency, ω_z is the axial frequency, and $\omega_{c'}$ is the cyclotron frequency each as observed from the motion of the particle in the trap. For the same simulation used in figure 13 a relative error of

$$\sqrt{\frac{\omega_m^2 + \omega_z^2 + \omega_{c'}^2}{\omega_c^2}} - 1 = -1.86 \cdot 10^{-8} \tag{25}$$

is observed. Here the frequencies were determined by measuring the length of half-periods for the y- and z-components of the trajectory. Projections of the three oscillation modes can be seen in figure 14. The gamma factor entering the calculation of ω_c was derived from the mean kinetic energy of the electron of about 0.8 eV, whereas neglecting this leads to an error of $1.5 \cdot 10^{-6}$. A commented Python notebook containing the analysis to reproduce these results is shipped with KASSIOPEIA.

13. Conclusion

We have presented the KASSIOPEIA particle tracking framework developed within the KATRIN collaboration. It was design to enable the fast and accurate computation of three-dimensional and axial-symmetric static electromagnetic fields created by complex electrode and magnet geometries, using KEMFIELD's fast multipole and zonal harmonic methods, respectively. In these fields particles can be tracked using various Runge–Kutta integrators (available up to eighth order) over extended periods of time using either an adiabatic approximation or by exactly solving the full Lorentz equation. The effects of synchrotron radiation can be taken into account in both cases. Furthermore, interaction models for electrons scattering on gaseous molecular hydrogen as well as helium, argon, and in silicon detectors are available. A major feature of the software is its flexibility, which stems from the XML configuration language that is used to describe all aspects of a simulation. If the pre-existing modules are sufficient for an end-user, no C++ code needs to be written and even novices can quickly devise complex and interesting simulations. The incorporation of new effects and models does require C++ knowledge, but their inclusion is made comparatively simple given the modular structure of the software. By now it is heavily used within the KATRIN collaboration to study all aspects of the corresponding experiment. Beyond KATRIN it is used by the aSPECT [73] and Project8 [71] collaborations. Therefore, we believe that this tool will be useful to a wider scientific community and hope for new users. The source code along with instructions for installing dependencies, compilation, and the first steps of configuration is available for download from: https://github.com/KATRIN-Experiment/Kassiopeia

The continued development of KASSIOPEIA will be on-going and it is expected that many improvements and additional features will be available in the future. Upcoming features will include the ability to track neutral particles with spin, and particles which have an internal quantum state. Futhermore, the field code will eventually incorporate additional integration techniques [74] to improve numerical stability and may be extended to non-static problems using time or frequency domain BEM techniques.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under Award Numbers FG02-97ER41041 and DE-FG02-06ER-41420. In addition, this work was supported by the German BMBF (05A14VK2), HAP, KHYS, and KCETA.

References

[1] Fukuda Y *et al* (Super-Kamiokande Collaboration) 1998 *Phys. Rev. Lett.* **81** 1562–7
[2] Ahmad Q R *et al* (SNO Collaboration) 2001 *Phys. Rev. Lett.* **87** 071301
[3] Ahmad Q R *et al* (SNO Collaboration) 2002 *Phys. Rev. Lett.* **89** 011301
[4] Aharmim B *et al* (SNO Collaboration) 2008 *Phys. Rev. Lett.* **101** 111301
[5] Eguchi K *et al* (KamLAND Collaboration) 2003 *Phys. Rev. Lett.* **90** 021802
[6] An F P *et al* (Daya Bay Collaboration) 2012 *Phys. Rev. Lett.* **108** 171803
[7] Abe K *et al* (T2K Collaboration) 2011 *Phys. Rev. Lett.* **107** 041801
[8] Kruit P and Read F H 1983 *J. Phys. E: Sci. Instrum.* **16** 313
[9] Kraus C *et al* (Mainz Collaboration) 2005 *Eur. Phys. J.* C **40** 447–68
[10] Aseev V N *et al* (Troitsk Collaboration) 2011 *Phys. Rev.* D **84** 112003
[11] The KATRIN Collaboration 2004 KATRIN Design Report 2004 FZKA Scientific Reports 7090 Forschungszentrum Karlsruhe http://bibliothek.fzk.de/zb/berichte/FZKA7090.pdf
[12] Agostinelli S *et al* (GEANT4) 2003 *Nucl. Instrum. Meth.* A **506** 250–303
[13] COMSOL Multiphysics Package http://comsol.com accessed: 2016-05-30
[14] Northrop T G 1961 *Ann. Phys., NY* **15** 79–101
[15] Groh S 2015 Modeling of the response function and measurement of transmission properties of the KATRIN experiment *PhD Thesis* Karlsruhe Institute of Technology http://digbib.ubka.uni-karlsruhe.de/volltexte/1000046546
[16] Bray T, Paoli J, Sperberg-McQueen C M, Maler E and Yergeau F 2008 *Extensible Markup Language (xml) 1.0* http://w3.org/TR/xml/
[17] Glück F 2011 *Progress In Electromagnetics Research* B **32** 351–88
[18] Garrett M W 1951 *J. Appl. Phys.* **22** 1091–107
[19] Paszkowski B 1968 *Electron Optics* (London: Iliffe)
[20] Szilagyi M 2012 *Electron and ion Optics* (New York: Plenum)
[21] Hawkes P W and Kasper E 1996 *Principles of Electron Optics* vol 3 (London: Academic)
[22] Glück F 2012 Axisymmetric Electric and Magnetic Field Calculations with ZonalHarmonic Expansion *PIER Proc.* **March** 1698–702 http://piers.org/piersproceedings/download.php?file=cGllcnMyMDEyS3VhbGFMdW1wdXJ8NEE1XzE2OTgucGRmfDExMTAxOTExMTgxNw==
[23] Corona T J 2009 Tools for electromagnetic field simulation in the KATRIN experiment *Master's Thesis* Massachusetts Institute of Technology
[24] Corona T 2014 Methodology and application of high performance electrostatic field simulation in the KATRIN experiment *PhD Thesis* University of North Carolina at Chapel Hill
[25] Cubric D, Lencova B, Read F and Zlamal J 1999 *Nucl. Instrum. Methods Phys. Res.* A **427** 357–62
[26] Glück F 2011 *Prog. Electromagn. Res.* B **32** 319–50
[27] Formaggio J *et al* 2012 *Prog. Electromagn. Res.* B **39** 1–37
[28] Saad Y and Schultz M H 1986 *SIAM J. Sci. Stat. Comput.* **7** 856–69
[29] Grengard L and Rokhlin V 1988 *The Rapid Evaluation of Potential Fields in Three Dimensions* (Berlin: Springer)
[30] Rokhlin V 1985 *J. Comput. Phys.* **60** 187–207
[31] Greengard L and Rokhlin V 1997 *Acta Numerica* **6** 229–69
[32] Ong E, Lim K, Lee K and Lee H 2003 *J. Comput. Phys.* **192** 244–61
[33] Lim K M, He X and Lim S P 2008 *Comput. Mech.* **41** 313–23
[34] Barrett J 2016 A spatially resolved study of the KATRIN main spectrometer using a novel fast multipole method *PhD Thesis* Massachusetts Institute of Technology
[35] Eupper M 1985 Eine verbesserte Integralgleichungsmethode zur numerischen Lösung dreidimensionaler Dirichletprobleme und ihre Anwendung in der Elektronenoptik *PhD Thesis* Universität Tübingen
[36] Leiber B 2014 Investigations of background due to secondary electron emission in the KATRIN-experiment *PhD Thesis* Karlsruher Institut für Technologie
[37] Stone J E, Gohara D and Shi G 2010 *Comput. Sci. Eng.* **12** 66–73
[38] Gabriel E *et al* 2004 Open MPI: goals, concept, and design of a next generation MPI implementation *Recent Advances in Parallel Virtual Machine and Message Passing Interface* (Berlin: Springer) pp 97–104
[39] Olive K A *et al* 2014 *Chin. Phys.* C **38** 9090001
[40] Dirac P A M 1938 *Proc. R. Soc. Lond.* A **167** 148–69
[41] Ford G W and O'Connell R F 1993 *Phys. Lett.* A **174** 182–4
[42] Furse D 2015 Techniques for direct neutrino mass measurement utilizing tritium beta-decay *PhD Thesis* Massachusetts Institute of Technology
[43] Fehlberg E 1969 Low order classical Runge–Kutta formulas with stepwise control *Technical report NASA* TR R-316 http://hdl.handle.net/2060/19690021375
[44] Dormand J R and Prince P J 1980 *J. Comput. Appl. Math.* **6** 19–26
[45] Hairer E, Nørsett S and Wanner G 1993 *Solving Ordinary Differential Equations I. Nonstiff Problems* 3rd edn (Berlin: Springer-Verlag) 8
[46] Prince P and Dormand J 1981 *J. Comput. Appl. Math.* **7** 67–75
[47] Verner J H 1978 *SIAM J. Numer. Anal.* **15** 772–90
[48] Tsitouras C and Papakostas S 1999 *SIAM J. Sci. Comput.* **20** 2067–88
[49] Chin S A 2008 *Phys. Rev.* E **77** 066401
[50] Nishimura H, Danjo A and Sugahara H 1985 *J. Phys. Soc. Japan* **54** 1224–7
[51] Liu J and Hagstrom S 1994 *Phys. Rev.* A **50** 3181–5

[52] Trajmar S, Register D F and Chutjian A 1983 *Phys. Rep* **97** 219–356
[53] Arrighini G, Biondi F and Guidotti C 1994 *Mol. Phys.* **41** 1501–14
[54] Chen Z and Msezane A Z 1995 *Phys. Rev.* A **51** 3745–50
[55] Rudd M E 1991 *Phys. Rev.* A **44** 1644–52
[56] Pitchford L C 2013 *J. Phys. D: Appl. Phys.* **46** 330301
[57] Pitchford L C *et al* 2013 *J. Phys. D: Appl. Phys.* **46** 334001
[58] Renschler P 2011 KESS—a new Monte Carlo simulation code for low-energy electron interactions in silicon detectors *PhD Thesis* KIT http://nbn-resolving.org/urn:nbn:de:swb:90-249597
[59] Antchva I *et al* 2009 *Comput. Phys. Commun.* **180** 2499–512
[60] Schroeder W, Lorensen B and Martin K 2006 *The Visualization Toolkit* 4th edn (New York: Kitware)
[61] Ayachit U 2015 *The ParaView Guide: A Parallel Visualization Application* (New York: Kitware)
[62] Fraenkle F, Bornschein L, Drexlin G, Glueck F, Goerhardt S, Kaefer W, Mertens S, Wandkowsky N and Wolf J 2011 *Astropart. Phys.* **35** 128–34
[63] Mertens S *et al* 2013 *Astropart. Phys.* **41** 52–62
[64] Mertens S *et al* 2012 *J. Instrum.* **7** P08025
[65] Prall M *et al* 2012 *New J. Phys.* **14** 073054
[66] Glueck F *et al* 2013 *New J. Phys.* **15** 083025
[67] Wandkowsky N, Drexlin G, Fraenkle F M, Glueck F, Groh S and Mertens S 2013 *J. Phys. G: Nucl. Part. Phys.* **40** 085102
[68] Wandkowsky N, Drexlin G, Fraenkle F M, Glueck F, Groh S and Mertens S 2013 *New J. Phys.* **15** 083040
[69] Erhard M *et al* 2014 *J. Instrum.* **9** P06022
[70] Wall B *et al* 2014 *Nucl. Instrum. Methods Phys. Res.* A **744** 73–9
[71] Asner D M *et al* 2015 *Phys. Rev. Lett.* **114** 162501
[72] Brown L S and Gabrielse G 1982 *Phys. Rev.* A **25** 2423–5
[73] Simson M *et al* 2009 *Nucl. Instrum. Methods Phys. Res.* A **611** 203–6
[74] Glück F and Hilk D 2016 arXiv:1606.03743