PAPER • **OPEN ACCESS**

GATE 10 Monte Carlo particle transport simulation: I. Development and new features

To cite this article: David Sarrut *et al* 2026 *Phys. Med. Biol.* **71** 015042

IOP Publishing Phys. Med. Biol. 71 (2026) 015042 https://doi.org/10.1088/1361-6560/ae237b

Physics in Medicine & Biology

IPEM Institute of Physics and Engineering in Medicine

PAPER

GATE 10 Monte Carlo particle transport simulation: I. Development and new features

David Sarrut[1], Nicolas Arbor[9], Thomas Baudier[1], Julien Bert[4], Konstantinos Chatzipapas[4,15], Martina Favaretto[5], Hermann Fuchs[19], Loïc Grevillot[5], Hussein Harb[4], Gert Van Hoey[16], Maxime Jacquet[1], Sébastien Jan[6], Yihan Jia[5,19], George C Kagadis[11], Han Gyu Kang[3], Paul Klever[8,12], Olga Kochebina[6], Wojciech Krzemien[17], Lydia Maigne[7], Philipp Mohr[12], Guneet Mummaneni[2], Valentina Paneta[10], Panagiotis Papadimitroulas[10,18], Alexis Pereda[7], Axel Rannou[4], Andreas F Resch[5], Emilie Roncali[2], Maxime Toussaint[13,20], Carlotta Trigila[2], Charalampos Tsoumpas[12], Jing Zhang[4], Karl Ziemons[8] and Nils Krah[1,14,*]

[1] Université de Lyon; CREATIS; CNRS UMR5220; Inserm U1294; INSA-Lyon; Université Lyon 1, Lyon, France
[2] University of California, Davis, Davis, CA, United States of America
[3] National Institutes for Quantum Science and Technology (QST), 4-9-1 Anagawa, Inage-ku, Chiba, Japan
[4] LaTIM, INSERM UMR1101, University of Brest, Brest, France
[5] MedAustron Ion Therapy Center, Wiener Neustadt, Austria
[6] Université Paris-Saclay, Inserm, CNRS, CEA, Laboratoire d'Imagerie Biomédicale Multimodale (BioMaps), Orsay, France
[7] Université Clermont Auvergne, Laboratoire de Physique de Clermont Auvergne, CNRS, Clermont-Ferrand, France
[8] FH Aachen University of Applied Sciences, Aachen, Germany
[9] Université de Strasbourg, IPHC, CNRS, UMR7178, F-67037 Strasbourg, France
[10] Bioemission Technology Solutions, BIOEMTECH, Athens, Greece
[11] 3DMI Research Group, Department of Medical Physics, University of Patras, Rion, Greece
[12] University of Groningen, University Medical Center Groningen, Groningen, The Netherlands
[13] Laboratoire CRCI2NA, INSERM, CNRS, Nantes Université, Nantes, France
[14] Department of Research and Development, Holland Proton Therapy Centre Delft, Delft, The Netherlands
[15] Department of Radiation Science and Technology, Technical University of Delft, Delft, The Netherlands
[16] XEOS, Ghent, Belgium
[17] High Energy Physics Division, National Centre for Nuclear Research, Andrzeja Soltana 7, Otwock, Swierk PL-05-400, Poland
[18] University of Thessaly, Department of Medicine, Medical Informatics Laboratory, Larissa, Greece
[19] Department of Radiation Oncology, Medical University of Vienna, Vienna, Austria
[20] Department of Medical Imaging and Radiation Sciences, Université de Sherbrooke, Sherbrooke, Canada
[*] Author to whom any correspondence should be addressed.

E-mail: krah.nils@gmail.com

Keywords: Monte Carlo simulation, scientific software, dose calculation in radiation therapy, nuclear imaging

OPEN ACCESS

RECEIVED
28 July 2025

REVISED
30 October 2025

ACCEPTED FOR PUBLICATION
24 November 2025

PUBLISHED
14 January 2026

Abstract

We present GATE version 10, a major evolution of the open-source Monte Carlo simulation application for medical physics, built on Geant4. This release marks a transformative evolution, featuring a modern Python-based user interface, enhanced multithreading and multiprocessing capabilities, the ability to be embedded as a library within other software, and a streamlined framework for collaborative development. In this Part 1 paper, we outline GATE's position among other Monte Carlo codes, the core principles driving this evolution, and the robust development cycle employed. We also detail the new features and improvements. Part 2 will focus on the architectural innovations and technical challenges. By combining an open, collaborative framework with cutting-edge features, such a Monte Carlo platform supports a wide range of academic and industrial research, solidifying its role as a critical tool for innovation in medical physics.

1. Introduction

For about 20 years, the GATE software, developed by the OpenGATE collaboration, has been utilized by thousands of users worldwide for applications in nuclear medicine, radiotherapy, and radiology, as well as in other fields like space radiation. GATE has played a crucial role in numerous academic research projects and industrial developments, leading to hundreds of scientific publications since its inception.

Built on Geant4 (Allison *et al* 2016), GATE has been described in several reference papers throughout its history (Jan *et al* 2004, 2011, Sarrut *et al* 2014, 2021a, 2022). While fully relying on Geant4 for its Monte Carlo engine, Gate provides: (1) easy access to Geant4 functionalities, and (2) additional features (e.g. variance reduction techniques) specifically designed for medical physics. Through the years, GATE has evolved to the major version 9, and specifically to 9.4. More technical details about the version 9.x series are available in Sarrut *et al* (2022).

More recently, the collaboration has faced challenges maintaining a large codebase (around 300 000 lines of code) developed over time by numerous contributors, most of whom contributed during their PhD or postdoctoral research projects. The current GitHub repository lists more than 80 unique contributors, although this repository was only established around 2012, meaning early contributors are not fully accounted for. This diversity has fostered innovation and experimentation, but has also led to maintenance issues. Some parts of the code have been 'abandoned', while others are duplicated. Additionally, the C++ language has evolved significantly over the past 20 years, introducing more efficient and convenient constructs such as smart pointers, lambda functions, or the `auto` keyword, which improve code robustness and maintainability.

Taking into account the core principles of GATE (community-driven, open-source, and focused on medical physics), we embarked on a project to propose a new way of executing Monte Carlo simulations in medical physics, including cross-platform capability. The primary goal of GATE 10 is to provide a simple yet flexible Python-based interface through which users can set up and run Geant4 simulations. Internally, GATE 10 aims to offer developers and contributors a structured framework that simplifies the implementation and maintenance of new features. Although this radical change inevitably breaks immediate compatibility with previous versions, it preserves the core philosophy of simulation description.

In this article (Part 1), we introduce the guiding principles, goals, and technical foundations of the GATE 10 project. We also describe the new functionalities and their potential impact on the medical physics community. The companion paper (Krah *et al* 2025) provides a more detailed exploration of the architectural innovations and technical challenges faced during the development of GATE 10.

2. General considerations about the development of GATE 10

2.1. User language: Python

Building Monte Carlo simulation applications with a general-purpose toolkit like Geant4 is a challenge due to the complexity of its code. Moreover, sharing simulation setups among users is not straightforward, as it requires writing (C++) code in a specific, reusable manner. Several high-level software frameworks have been developed to address these issues and are currently used by a large community, such as GAMOS (Arce *et al* 2014), TOPAS (Faddegon *et al* 2020) or GATE 9 (Sarrut *et al* 2022). All three target medical physics applications and rely on macro script commands (Geant4 macros for GATE and GAMOS, and an ad-hoc system for TOPAS). This approach allows for a simplified description of simulations, which is essential because users of such applications should focus on physics rather than the technical details. A disadvantage of macro commands is that they are limited to parameter descriptions with few or cumbersome instructions (loops, conditional statements, etc) and are not easily extendable.

For GATE 10, we chose Python as the user language to configure and run simulations and as a user interface to the low-level Geant4 engine. The reason for this choice was that Python has become a major tool for data science over the past years, offering ease of learning and convenient access to powerful mathematical (Harris *et al* 2020) (NumPy), statistical, and artificial intelligence (AI) libraries written in C++ and optimized for high-performance computing (HPC). Instead of requiring users to learn a new syntax of commands specific to GATE, we recognized that a large proportion of users already know Python or will learn it during their career.

2.2. Previous efforts to steer Geant4 via Python

The Geant4 collaboration has previously proposed an initiative to steer Geant4 simulations via Python with the g4py or g4python[21] package, which, to our knowledge, is no longer developed or maintained. Although g4py provided a way to avoid C++ compilation of a Geant4 simulation, it still required users to have detailed knowledge of the Geant4 toolkit because it directly exposed low-level Geant4 classes and functions. Furthermore, g4py required complex installation procedures due to its dependence on the Boost library (Schling 2011). Other solutions to expose Geant4 code to Python have also been proposed, including `geant4_pybind`[22], but it is unclear how these projects will evolve. In any case, users

[21] https://github.com/koichi-murakami/g4python.
[22] https://github.com/HaarigerHarald/geant4_pybind.

must always compile Geant4 by themselves. Additionally, GATE implements some of its core functionality in C++, which needs to be exposed to Python. For these reasons, we designed a binding layer from scratch and included it into GATE 10. We explain the technical details of this new contribution in Part 2 (see companion paper (Krah *et al* 2025)).

2.3. File management and output

Simulations in GATE require a multitude of parameters and options, as well as access to external input data such as anatomical images, ROOT files, or material databases. We sought to achieve a balance between dependence on external software packages and the use of internal libraries.

2.3.1. Images

Input images are handled by the ITK toolkit (McCormick *et al* 2014), which supports reading and writing a wide variety of image formats, including common types like MHD, NIfTI, and DICOM. ITK can manage images with different types of numerical data (e.g. float, int) and dimensions (2D, 3D, 4D). The reading and writing of images are performed by Python, leveraging ITK's Python wrapping. Once the images are loaded into memory, they are transferred to the C++ side and managed by the C++ part of ITK, ensuring more efficient and integrated image processing. Relying on ITK ensures robustness and access to many file formats.

2.3.2. ROOT files

ROOT files are a standard file format for physics-related data, such as lists of particles (often referred to as 'trees' or 'phase spaces'). This format is convenient for managing large file sizes and supports compression. Historically linked to the Geant4 community and developed at CERN ROOT is familiar to many Geant4 users (Brun and Rademakers 1997). Moreover, as GATE has long used the ROOT output file format and GATE users are thus likewise familiar with it, we decided to keep it as the primary format for the list-mode data output.

2.3.3. Other data

Additionally, GATE manages other types of data, such as material files that follow the traditional GATE format (i.e. text files describing material density, elements, etc) or lists of energy spectra for conventional ICRP107 databases (Eckerman and Endo 2008). In the future, we consider integrating file formats from the Emission Tomography Standardization Initiative (ETSI[23]) once these are well established. ETSI is currently developing a new unified file format for raw data in preclinical and clinical emission tomography, including PET and SPECT imaging. This format will accommodate list-mode data, singles and sinograms/histograms, focusing on efficient data manipulation, storage and memory usage. In addition, having a common raw data file format for both simulated and measured data will make the workflow more efficient. For example, such integration will facilitate the smooth incorporation of GATE-generated output raw data into reconstruction software, enhancing the overall workflow and compatibility with industry standards.

2.4. Transition from GATE 9 to 10

The transition to GATE 10 requires users to convert their existing macro files into Python scripts and adapt to some changes in parameter names and logic. While the development aimed to preserve the feel of previous versions, we prioritized modifications that offered significant improvements.

A key example is the replacement of the `System` concept for PET or SPECT detectors used to collect detected hits via a Geant4 Sensitive Detector: now, any volume can collect hits via an internal mechanism based on Geant4 Primitive Scorers.

We anticipate that users will require some time for transitioning from GATE 9 to GATE 10. Therefore, the GATE collaboration has decided to continue maintaining the GATE 9 codebase, adapting it to new versions of Geant4 and providing minor bug fixes, but there will be no further development work on GATE 9. On the other hand, GATE 10 development will focus on consolidating the released code and on incorporating useful functionality that has been part of GATE 9 but not yet implemented in GATE 10.

We have evaluated the feasibility of automatic tools to translate existing GATE 9 input files to GATE 10 but concluded that such tools would be unreasonably difficult to implement reliably. Instead, the online documentation will contain a growing section of examples of simulations that have been successfully translated by users.

[23] https://etsinitiative.org.

2.5. Robust development cycle and cross-platform compatibility

Based on the experience gained from the open development of GATE, the challenge was to establish a robust framework to collaboratively develop new features and fix bugs while ensuring reliable validation, consistent results and timely updates aligned with Geant4 releases. The solution was to implement modern continuous integration (CI) practices and a comprehensive testing strategy. For every new feature added to the codebase, the following rules are enforced: (1) code must be submitted via the GitHub's Pull Request mechanism, (2) automated verification tests must be provided, and (3) documentation must accompany the submission.

Developing tests in a Monte Carlo context presents unique challenges. Firstly, the tests should ideally have short computation times, even though most simulations require long runtimes. Secondly, the outputs are stochastic, meaning comparisons with reference data must be performed statistically. Small variations in results are expected, which makes direct comparisons more difficult. This challenge has already been acknowledged by the Geant4 collaboration, which has developed specific physics-based tests (Arce *et al* 2021, 2025a).

Each feature to be tested is unique, so a set of global recommendations is adopted. The random engine generator seed must be fixed, and computation times should be kept 'as low as reasonably achievable', with a maximum of 1–2 minutes. When comparisons with reference data are necessary, statistical tests such as Z-score, Wasserstein distance (for comparing distributions), or sum of squared differences (for images or dose maps) are employed. Each feature must have at least one associated test before being merged into the main branch. Each test ends with a single True/False result, indicating whether the test passed successfully. This process has proven effective; as of early 2025, more than 230 tests have been created.

Thanks to GitHub's CI infrastructure, every code modification automatically triggers the test suite. If any test fails, the code is not merged, and a discussion with the contributor is initiated to understand and resolve the issue. This approach enforces a minimal set of requirements for contributors: they must provide a formal test, and the CI system simplifies the process of verifying that new contributions do not break existing functionality. While tests are triggered automatically by CI, they can also be run locally on the user's machine using the `opengate_tests` command.

On GitHub, tests are conducted across multiple Python versions (from 3.9 to 3.12 at the time of writing) and various platforms (OSX, Linux, Windows), ensuring cross-platform compatibility. To reduce computational overhead, a Monte Carlo-inspired approach (pun intended) is used, where a randomly selected subset of tests is run on each platform for each commit. Additionally, a complete test suite is run once a week across all platforms and Python versions. Note that Windows compatibility is still not fully functional, and multithreading cannot yet be used together with ROOT output.

Finally, the core libraries are built and integrated into a single 'wheel' (a Python package) that is uploaded to PyPI[24]. This allows users to install GATE 10 with a single command: `pip install opengate`. Under the hood, the process involves compiling several dependent libraries such as ITK and Geant4. Since Geant4 requires a large physical data library (approximately 2 GB), which cannot be included in the wheel due to size restrictions, a dedicated mechanism ensures that the correct data is automatically downloaded during installation if not already available. This mechanism also handles the automatic download of test data during the first execution of the `opengate_tests` command. While the standard installation enables users to create simulations in Python, it does not include the C++ source code. For developers who wish to extend GATE 10 by adding new features, the installation of a specific developer option is essential, which involves compiling ITK and Geant4 from source.

In conclusion, the development cycle is managed through a test-based framework and CI, ensuring reliable software quality and facilitating collaborative contributions. Everything is openly available on https://github.com/OpenGATE/opengate and the collaboration allows contributions from all over the world.

3. New and enhanced features

In this section, we describe features that have either significantly changed compared to the previous versions of GATE, or that have been newly developed and added to GATE 10. We group these into geometry, actors (scorers), physics, and sources.

[24] https://pypi.org/project/opengate/.

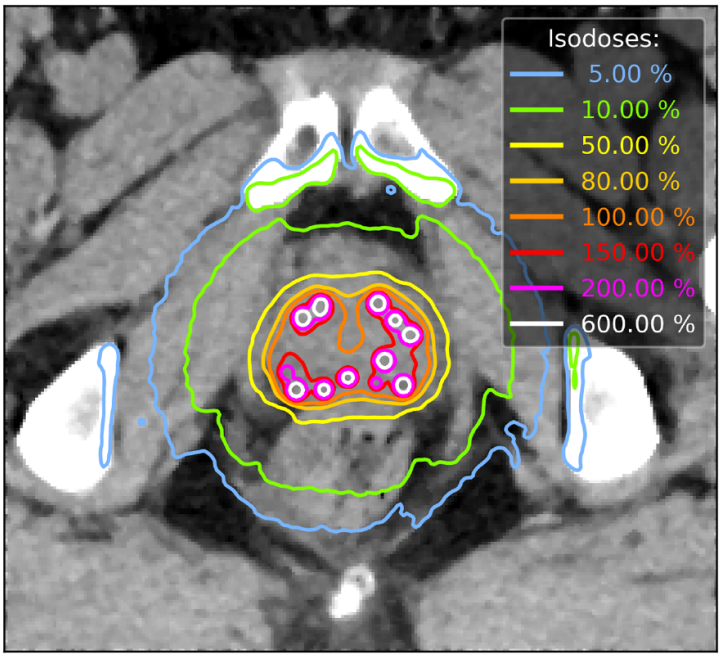

Figure 1. Example of the *parallel worlds* feature through prostate brachytherapy treatment simulation. The isodoses, overlaid on the patient's CT scan, are normalized with respect to the prescription dose.

3.1. Geometry

3.1.1. Parallel worlds

Parallel worlds in Geant4 provide a generic way to run simulations in different geometries simultaneously. A practical use case in medical physics arises when volumes overlap, such as the gantry of a linear accelerator or SPECT detectors with the patient's CT bounding box or when the user defines an analytical volume inside a voxelized CT image (e.g. brachytherapy seeds). In GATE 10, users can use the Geant4 native parallel world logic with minimal additional configuration, as Geant4 mechanics are handled internally. However, a caveat of using parallel worlds is the increased tracking time, as the Geant4 engine tracks particles in all worlds, potentially doubling the computation time.

A medical application that illustrates this feature is low-dose-rate prostate brachytherapy, where dozens of radioactive seeds (represented as analytical objects) are inserted into the prostate, as a voxelized CT volume. A dosimetry result, based on low-dose-rate clinical data, is illustrated in figure 1. In this case, the computational phantom of the patient is defined in the 'main world' while the 53 seeds defined in the treatment plan are described in a 'secondary (parallel) world'.

3.1.2. Tessellated geometry

GATE 10 introduces a tessellated geometry feature internally handled by the Geant4 `G4TessellatedSolid` class. The user can import mesh phantoms in stereolithography (STL) format and thus consider complex 3D CAD models or fine-detail objects. Figure 2 shows the GATE 10 simulation of a cardiologist's exposure to scattering radiation during an x-ray-guided intervention in the operating room. The simulations were based on different poses of a phantom provided by the ICRP 145 dataset (Kim *et al* 2020) and created using a third-party software.

3.1.3. Boolean solids

Geant4 allows users to combine solids via Boolean operations, i.e. to create intersections, unions, and subtractions of geometrically parameterized solids. Contrary to previous GATE versions, GATE 10 includes this functionality. Although the Boolean operations are performed on Geant4 solids, we decided to implement Boolean volumes, in analogy to other GATE volumes, and hide the underlying mechanisms from the user in favor of a simpler and more intuitive interface. The user needs to define and configure two volumes, e.g. 'A' and 'B', and then apply the Boolean operation via a function, e.g. `subtract_volumes(A, B)`. The returned Boolean volume is then included in the simulation representing the volume 'A-B'. Boolean solids are useful for constructing complex shapes such as complex collimators in SPECT, but at the expense of computing time as the navigation in a complex Boolean solid is proportional to the number of constituent volumes.

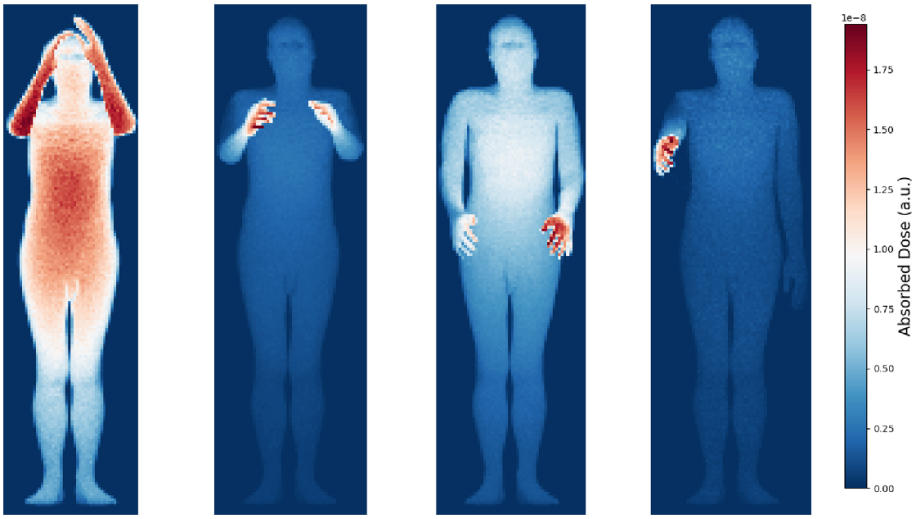

Figure 2. Example of the *tessellated geometry* feature through skin dose exposure calculation in x-ray guided intervention, for different poses and positions of the physician around the patient.

3.1.4. Voxelized geometry

In medical physics simulations, a CT image of a patient or a computational phantom is often used. It is represented as an array of voxels and requires special care to allow fast particle tracking in Geant4. From the simulation point of view, a voxel is a small box characterized by its material properties, through which particles need to be transported. In GATE, any three-dimensional input image can be used to create a voxelized geometry component (`ImageVolume`) by providing a mapping from image values to materials as an input parameter (`voxel_materials`). GATE does not make any assumptions about the nature of the input image, as long as it can be mapped to materials. Internally, this is managed by a fast Geant4 navigator based on nested parameterization using `G4PVReplica` and `G4PVParameterised` features (Sarrut and Guigues 2008). A common case in medical physics is the use a CT image in Hounsfield Units. To assist with this, GATE 10 provides a helper function, `HounsfieldUnit_to_material`, which creates an image value-to-material lookup table.

3.1.5. Voxelization of scene

GATE 10 introduces a new concept by which the user can create a voxelized representation of the simulation geometry, i.e. of the Geant4 volumes such as spheres, boxes, cylinders, etc contained in a simulation. Through a single command (`voxelize_geometry`), the user can transform the analytically described geometry into a discrete 3D grid of voxels, similar to an image. The user needs to specify the grid size (spacing) and can furthermore specify which volumes should be included in the voxelized representation, e.g. by considering only a certain subset of volumes such as an imaging sensor. The function `voxelize_geometry()` returns an ITK image, which can be saved to disk, and a dictionary that maps voxel labels to the materials of the geometry scene. `voxelize_geometry()` performs the conversion by iteratively querying the Geant4 engine for the material at each voxel's center to determine the voxel label.

3.2. Actors and scorers

3.2.1. Phase space actor

The phase space actor in GATE records detailed information about particles entering or exiting a specified volume during a simulation or particles undergoing interactions in the volume for the first time. In GATE 10, the user can select from more than 50 attributes, including key parameters related to particle history, such as parent particle information, volume of generation, position, direction, energy, time, and track length. The phase space actor generates output in the ROOT file format, which can be easily analyzed using Python libraries such as uproot (Pivarski *et al* 2020).

3.2.2. Digitizer actors

Introduced in an earlier version of GATE (Jan *et al* 2004, Kochebina *et al* 2024), the Digitizer (technically an actor) is a key component of the GATE workflow for PET, SPECT, and other imaging applications. With the help of analytical and semi-analytical models, digitizers can simulate the signal processing chain including signal readout and electronic distortions and reproduce effects on energy, position, and

time resolutions as well as transport and detection efficiencies. GATE 10 currently provides ten digitizers that can be arranged by a user in a processing chain. ROOT output can be saved at each step of this digitizer chain.

GATE 10 also includes a Coincidence Sorter for PET or Compton Camera applications. It matches pairs of single events (singles) that occur within a user-defined coincidence time window. The user can choose from six available policies to resolve pile-up and impose rejection rules based on geometry considerations. Currently, and contrary to the previous version of GATE, the Coincidence Sorter operates as a post-processing step in Python, i.e. after the simulation has terminated, and users can re-run the coincidence sorter on simulation output stored in disk.

3.2.3. Enhanced linear energy transfer (LET) actor

In GATE 10, both track- and dose-averaged LET (unrestricted) can be calculated, employing by default method C from Cortés-Giraldo and Carabe (2015). Currently, stopping powers are obtained using `G4EmCalculator`, but future updates of the software aim to support user-provided custom stopping power tables. The output of this scorer is a 3D matrix of LET values.

3.2.4. Relative biological effectiveness (RBE) actor

The `RBEActor` in GATE 10 can calculate RBEs and RBE-weighted dose based on the local effect model I (`LEM1lda`) (Krämer and Scholz 2006) and a modified microdosimetric kinetic model (`mMKM`) (Inaniwa *et al* 2010). The actor supports multithreading and multiprocessing in GATE 10. In addition to specifying model parameters via a lookup table, users can customize cell radiosensitivity parameters and model-specific parameters, allowing flexibility in radiobiological modeling.

3.2.5. Track length estimator (TLE) (hybrid) actor

The TLE (Baldacci *et al* 2014) implements a variance reduction technique (VRT) designed to rapidly compute the deposited energy and dose in voxelized geometries for low-energy gamma rays, typically below 1–2 MeV, where electron delocalization is minimal. This method is particularly effective in scenarios where precise energy deposition calculations are needed without the computational overhead of detailed particle tracking. In GATE 10, we have optimized the simulation process of the TLE actor by implementing a hybrid mode that automatically switches between tracking gamma dose with and without the TLE method, based on the gamma ray's energy. The TLE method has previously been validated and has demonstrated significant speed improvements (up to 100 times faster) depending on the configuration (Noblet *et al* 2016). The enhanced versions seTLE (Split Exponential) (Smekens *et al* 2014) and neutron (Elazhar *et al* 2018) are not yet implemented in GATE 10.

3.2.6. Free flight actor

GATE 10 introduces a new (approximate) VRT, particularly beneficial for Monte Carlo simulations of SPECT. The method is based on the concepts of Free Flight, a biasing technique from Geant4, and Angular Acceptance rejection in a two-step approach, addressing both primary and scattered events. Primary gamma photons are tracked in straight lines towards the detector, considering angular acceptance and weighted contributions. Scattered events are simulated using a combination of split Compton and Rayleigh interactions, which are then free-flighted towards the detector. The Free Flight actor demonstrated a speedup of approximately 50 times compared to standard analog Monte Carlo simulations, with even greater improvements in high-count regions. A significant advantage of this method is that it can be applied to various geometrical elements in the simulation scene without requiring a detector response function, such as an angular response function (ARF). This makes it especially suitable for the development of SPECT collimator designs.

3.3. Sources and physics

3.3.1. Physics lists and physics options

GATE 10 provides a transparent interface to the extensive library of physics lists available in Geant4. Users can easily enable various validated physics models via simple commands, without any modification to the underlying Geant4 physics engines or databases. This approach ensures consistency with the Geant4 toolkit and allows users to leverage its continuous improvements. Available physics lists include, but are not limited to: Electromagnetic Physics (Standard, Low Energy e.g. Livermore, Penelope), Hadronic Physics (FTFP_BERT, QGSP_BIC, QBBC, etc), Radioactive Decay.

Beyond selecting these lists, specific processes and models can be enabled. For example, the quantum entanglement of annihilation photons, introduced in Geant4 11.0, can be activated using a dedicated

command (`g4_em_parameters.SetQuantumEntanglement(True)`). The standard two-gamma annihilation process is naturally included in these lists; however, triple-gamma annihilation is not yet available. Atomic de-excitation processes such as fluorescence, the Auger cascade, and particle-induced x-ray emission can be activated with the Python interface to the Geant4 `G4EmParameters`.

Positronium lifetime imaging is a relatively recent multiphoton imaging technique that extends conventional PET tomography by incorporating additional information related to positronium, i.e. a metastable bound state of an electron and a positron (Shibuya *et al* 2020, Moskal *et al* 2021, 2024, Qi and Huang 2022, Huang and Qi 2024, Steinberger *et al* 2024, Tashima and Yamaya 2024, Mercolli *et al* 2025). Accurate simulation of positronium decays demands the inclusion of various decay channels and careful modeling of the lifetime component mixtures within phantoms. Starting from version 9.3, GATE supports positronium decay simulations by modeling part of the positronium-related physics, including three decay models and a prompt gamma emission model. Ongoing work aims to port this functionality to GATE 10 and extend it to facilitate modeling complex phantoms.

3.3.2. Annihilation photon acollinearity (APA)

The spatial blurring in positron emission tomography (PET) images caused by the APA becomes more pronounced as the effective diameter of the scanner increases. This effect is known to follow a Gaussian distribution. Toussaint *et al* showed (2024) that APA was incorrectly modeled in various Monte Carlo software, which resulted in an underestimation of its effect on spatial resolution. Their corrections and conclusions were included in GATE 10.

In GATE, the user can define a source for PET systems in three different ways: as an ion source (e.g. of ^{18}F or ^{68}Ga), as a positron source (with the corresponding energy spectra), or as a back-to-back gamma source (neglecting positron travel). By default, annihilation photon pairs from positron-electron annihilation will be collinear. However, it is known that, for example, for water between 20 °C and 30 °C, the deviation of APA follows a 2D Gaussian distribution with a FWHM of 0.5° (Colombino *et al* 1965). To enable this behavior in an ion or positron source, the user must set the mean energy per ion pair to 0.5 eV in the materials where APA is desired. The implementation of APA for back-to-back sources is based on the simplifying assumption that its deviation follows a 2D Gaussian distribution. The user activates APA by a Boolean flag and defines the FWHM of the 2D Gaussian distribution (with 0.5° being the default).

The deviation of APA in a human subject follows a double Gaussian distribution with a combined FWHM of 0.55° (Shibuya *et al* 2007). Although the double Gaussian distribution is currently not available in GATE, setting the mean energy per ion pair of the material to 6.0 eV results in a 2D Gaussian with a FWHM of 0.55°.

3.3.3. Scanned pencil beam sources

Like previous GATE versions (Grevillot *et al* 2011), GATE 10 enables users to simulate the delivery of scanned ion pencil beam treatment plans, using the `IonPencilBeamSource` and `TreatmentPlanPencilBeamSource`. In this treatment modality, the target is irradiated by scanning the beam across the transverse plane with steering magnets, while adjusting the beam energy to reach different depths within the target. While retaining all the features existing in GATE 9, the new version of the source introduces the capability to directly read treatment plan files in their native DICOM format, thereby eliminating the need for prior conversion of the irradiated spot information into a text file. Additionally, it offers greater flexibility by allowing users to configure the source without a treatment plan, using custom spot data instead.

3.3.4. Voxelized sources

A three-dimensional activity distribution can be simulated with the help of A `VoxelSource` simulates three-dimensional activity distribution based on an input image in which each voxel contains the activity. For example, a SPECT image can be used as activity map and the voxelized CT scan of the patient as attenuation map (assigned to the `attached_to` parameter). A useful helper function, `get_translation_between_images_center()`, aligns the voxelized activity with the CT image by computing the correct translation (`get_translation_between_images_center()`). A label image, e.g. as the result of a segmentation, can be converted to an activity map by the help of the `voxelized_source` helper function and a label-to-activity lookup table provided by the user.

3.3.5. Time-activity curve (TAC) sources

The TAC feature can now be enabled for a particle source. The TAC is represented as a discrete histogram parameterized by two vectors: time points and corresponding activities. When enabled, the source's

activity at the current the simulation time is updated using linear interpolation of the TAC. The activity is set to zero at simulation times outside the range of the TAC vector. This feature is useful for simulating dynamic processes in which the activity of the source changes over time due to radioactive decay and tracer kinetics.

3.3.6. Photon from ion decay (PHID) source

A novel virtual source model named PHID was designed to simulate photons emitted during the complex decay chains of alpha-emitters, making it useful for SPECT acquisition simulations (Sarrut *et al* 2024). The model extracts photon emission lines for both isomeric transition and atomic relaxation processes from the Geant4 database for a given alpha-emitter. The Bateman equations are used to calculate photon abundances and activities throughout the decay chain, considering decay rates and initial radionuclide abundances over a specified time range. PHID generates photons with accurate energy and temporal distributions without simulating the entire decay chain, resulting in a speed-up of the simulation. For example, in a simulation of 1 MBq of ^{225}Ac in water, PHID demonstrated a 30× speedup compared to analog Monte Carlo simulations. Additionally, compared to a simplified source model using only the two main photon emission lines, PHID simulated twice as many particles and detected 60% more counts in the resulting images.

3.3.7. Generative adversarial network (GAN) sources

Particle tracking within a voxelized patient may be computationally costly because each voxel intersection triggers a new particle step. Recently, the development of a new approach that combines low-statistics Monte Carlo simulations with AI was proposed. The outgoing gamma photons from a patient's body are recorded at the body surface and used to train a deep neural network based on a GAN (Goodfellow *et al* 2014). Once trained, the GAN's generator replicates the phase space distributions of these particles, effectively serving as a low-cost particle source. With conditional GANs (cGANs), a single low-statistics simulation is sufficient to train an activity-parameterized family of GANs, enabling the generation of particles for any desired activity distribution. These GAN-based sources have been integrated into GATE 10, facilitated by the Python interface. To enable these methods, PyTorch (Paszke *et al* 2019), a widely used deep learning framework, must be installed.

Coupled with ARFs, the GAN source has shown speed-ups of more than 100 times in image generation (Sarrut *et al* 2021b, Saporta *et al* 2022). However, the approach requires specific cGAN training for each CT image and customized ARF models for different detector and energy window configurations. GAN sources can also be useful for modeling linear accelerator (linac) phase-space data (Sarrut *et al* 2019, 2021c).

3.3.8. Optical simulation using GANs: optiGAN

Optical photon transport simulations are essential for developing and optimizing radiation detectors in medical imaging and high-energy physics. While detailed optical Monte Carlo methods remain the gold standard for modeling photon interactions, their high computational requirements pose a challenge, especially when considering large system simulations. To address this, a GAN model, optiGAN, was developed (Trigila *et al* 2023), optimized (Srikanth *et al* 2024), and integrated into GATE 10 (Mummaneni *et al* 2025). OptiGAN reduces the optical photon transport simulations time to half while preserving modeling accuracy with more than 92% similarity between traditional and GAN simulations in GATE 10 (Trigila *et al* 2023, Mummaneni *et al* 2025).

An optical simulation using optiGAN provides very similar results as the full optical simulation but without the need for optical photon tracking. Users must define both the geometry and source. Currently, GATE 10 includes a specific model checkpoint from offline training on a predefined dataset (a $3 \times 3 \times 10$ mm^3 bismuth germanate crystal). This configuration is readily available for testing, with examples provided on GATE's GitHub. However, since the implementation is open-source (Lab 2025), users will be able to generate their own datasets to train AI models, and integrate them into GATE 10.

3.4. Automatic serialization into JSON file

GATE 10 includes an automated feature which writes the configuration of the entire simulation into a JSON file, i.e. a structured, human-readable file. JSON is a widely adopted data format supported by a multitude of libraries. The user can activate this feature by setting `sim.store_json_archive = True`. The manager-based architecture, together with functionality in the base class `GateObject` (see the second part of this paper (Krah *et al* 2025)), automatically collects all relevant user input parameters of all components of a simulation and serializes them into JSON format. The generated JSON file can be reloaded into GATE 10, and the simulation will be recreated based on its content. This feature is

particularly useful when a user runs multiple simulations from a single Python script, e.g. with different geometry configurations, and wishes to maintain access to parameters of each simulation. It is also helpful as part of an archiving mechanism when GATE 10 is used for automated routine dose calculation tasks, such that any simulation can be re-run later, although it was generated dynamically by a script. Finally, a structured representation of the simulation in JSON format could help authors who want to follow guidelines from TG-268 report (Sechopoulos *et al* 2018) to share simulation details.

3.5. Dynamic parameterization, timing

A Geant4 simulation, by itself, is only time-aware with respect to tracking of primary particles. However, many simulations in medical physics and imaging require time-dependent modeling on a macroscopic time scale. For example, the source rotates around the patient, the patient breathes, the activity distribution changes due to kinetics, etc. To accurately reflect such dynamics, simulations must be discretized in time. GATE uses Geant4's concept of *runs*: the simulation is initialized once at the beginning, and certain aspects of the simulation are changed after all primary particles which belong to one run are tracked. Previous GATE versions implemented Geant4's concept of runs in a task-specific fashion. In GATE 10, we adopted a more generic approach through 'dynamic parameterisation'. Users can now define a sequence of run timing intervals for a GATE simulation, i.e. pairs of moments in time when a run is considered to start and stop, respectively. On the other hand, the user can provide a list of parameters to a component of a simulation instead of only providing a single parameter. For example, if a simulation is split into 8 run intervals, the user might provide a list of eight CT images to dynamically parameterize the patient geometry (e.g. to account for breathing). Additionally, the user might provide a list of eight rotation matrices to rotate the particle source around the patient. The generic 'dynamic parameterisation' approach provides a unified user interface, yet it is flexible and can be easily extended to other compatible parameters in the future.

4. Discussion

In this first article, we have presented GATE version 10, outlining its new major features and the principles guiding its development. GATE has transitioned from a standalone application into a versatile software platform that can be used not only for conventional simulations but also as a library integrated into other software packages. This latter capability is particularly important for embedding GATE into specialized software, such as treatment planning systems or image reconstruction frameworks.

We outlined the impact of such highly collaborative scientific open-source software as a significant contributor in advancing the use of physics in medicine, biology and beyond. This impact can be evidenced by the publication of thousands of articles and citations of the four main GATE publications. Furthermore, GATE has been adopted by several major companies for the design and optimization of imaging or radiation-based prototypes and commercial systems. GATE is also widely used as a valuable resource in teaching several medical physics courses by providing a range of well-prepared computational laboratory materials.

We also highlighted the collaborative and community aspects, both interests and challenges, of the GATE initiative. The collaboration is open, governed only by a contribution-based model, allowing any group to integrate as soon as they propose a relevant and useful contribution. However, over time, some groups may stop contributing to the further development of GATE, necessitating a clearer definition of our vision and how resources should be maintained.

Due to its significantly lower learning curve compared to the underlying Geant4 toolkit, we think that GATE with Python can be a key educational tool. It allows students to focus on the principles of physics rather than the complex programming details. As an example, students from the French DQPRM qualification ('Diplôme de Qualification en Physique Radiologique et Médicale') follow Monte Carlo courses using GATE. We believe that this new version will further support and enhance the education of medical physicists in Monte Carlo simulation techniques.

The main focus of our work was to develop a user-friendly and maintainable Python-based Monte Carlo simulation software that builds on top of the development of previous GATE versions. At the same time, we carefully ensured that the hybrid C++/Python architecture would not come at the expense of increased computation time. In essence, this is ensured by implementing computation intensive, frequently called components, e.g. the scoring algorithms in actors, in C++ and rely on Python for less critical parts. Once the user starts a simulation, the Geant4 engine is initialized and started with the exact same computation time as a conventional Geant4 (or Gate9) simulation. Exceptions may occur with specific components that require back-and-forth communication between Python and C++, e.g. when using neural networks with PyTorch. In such cases, the Geant4 engine is briefly stopped once

a batch of data is ready to be sent-to/retrieved-from the neural network. This slight pause is infrequent and almost unnoticeable when the batch size is large.

We monitored time performance during development to ensure that computation times remained equal or inferior to GATE 9 simulations. A full benchmark against GATE 9, or even other Monte Carlo codes, is beyond the scope of this paper, but points to interesting work in the future.

It is important to note that GATE (9 and 10) does not change the underlying Geant4 physics. If a Fano test passes in Geant4, it will also pass in GATE. There are more than 240 tests, which are now part of GATE 10's CI framework (see section 2.5). These are mostly functional tests, but the outcome of the test is also based on the consistency of the simulation results (among the different minor versions of GATE 10). Physics-based tests against reference data would be more devoted to the Geant4 engine side.

Despite its advancements, GATE 10 remains an ongoing development with several limitations. First, there is a deliberate break in backward compatibility, which requires users to manually transcribe legacy macros into Python scripts. Technical limitations also exist; full functionality on the Windows platform is not yet achieved, and multithreading is currently incompatible with ROOT output. Furthermore, not all of Geant4's fine-tuning capabilities are exposed through the Python interface, potentially requiring advanced users to implement custom C++ code for specific needs. Some data processing modules are also still being finalized; for instance, coincidence sorting currently operates as an offline script, and digitizers for time-related effects like dead time and pile-up are in development.

Looking ahead, the new architecture opens up several avenues for future development. One area is the extension of GATE's capabilities into radiobiology, with better integration/link with the Geant4-DNA toolkit (Incerti *et al* 2010, Arce *et al* 2025a) for track structure simulations at the cellular and subcellular level. We also plan to expand the portfolio of variance reduction techniques; for example, the free-flight method could be adapted to estimate out-of-field dose distributions in radiation therapy. Finally, while the Python interface already facilitates the use of neural networks, future work will focus on a tighter integration of AI-based models.

5. Conclusion

Looking beyond GATE 10's specific features, it is crucial to acknowledge the broader context in which such scientific software exists and evolves. The current development model was as follows: GATE is developed feature by feature by researchers for researchers, typically during their PhD or postdoc periods, supervised by their advisors. However, only a few full-time researchers or engineers can devote time to coding. Funding for such human power is crucial, but has, until now, been associated with research grants as a side product of conventional research publications. This makes it challenging to sustain a continuous development, maintenance, and support of such a large and complex program challenging and slow, as specific funds for these service tasks have been a rare exception.

As we look into the future, the interest of the OpenGATE collaboration is to continue to innovate and adapt GATE according to the significant advances of computational and software methods to meet the evolving needs of medical physics and beyond. The challenges ahead will require sustained large-scale collaboration, a continuous stream of funding, and a commitment to maintaining and enhancing this valuable toolkit. We hope that GATE will continue to serve as a cornerstone in the advancement of biomedical imaging and radiation therapy, fostering a community of researchers and developers dedicated to pushing the boundaries of scientific discovery and technological innovation.

In the companion paper (Krah *et al* 2025), we detail technical challenges and selected architectural innovations.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/OpenGATE/opengate.

Acknowledgments

The authors acknowledge funding support from the following Grants: ANR-21-CE45-0026, INCa-INSERM-DGOS-12563, ANR-11-LABX-0063, ANR-11-IDEX-0007, Erasmus+ program. W K acknowledges the support by the Foundation for Polish Science through the FIRST TEAM FENG.02.02-IP.05-0152/23 programme co-financed by the European Union under the European Funds for Smart Economy 2021-2027 (FENG). This research was funded in part by the Austrian Science Fund (FWF) 10.55776/PIN5705024. L G, M F, Y J, and A F R appreciate the support by the 'Niederösterreichischen

Wirtschafts- und Tourismusfonds' and European union in form of the EFRE funds (WST3-F-5033232/002-2020) and FW774A0101 (PAIR project) of the Austrian Science Fund (FWF). ANR-21-RHU5-0005 and SIRIC LYriCAN Grant INCa-DGOS-INSERM-ITMO cancer_18003.

Funding

K C was supported by the Dutch Ministry of Economic Affairs and KGG under grant number 1300033255. M T acknowledges the support by the French INCA-DGOS-INSERM/ITMO Cancer 18011 (SIRIC ILIAD) and by the Discovery grant RGPIN-2025-06139 of the Natural Science and Engineering Research Council of Canada (NSERC).

ORCID iDs

David Sarrut 0000-0002-4854-4141
Nicolas Arbor 0000-0001-6457-1767
Julien Bert 0000-0002-2756-942X
Konstantinos Chatzipapas 0000-0003-4006-9304
Martina Favaretto 0009-0000-7899-3952
Hermann Fuchs 0000-0001-9003-5730
Loïc Grevillot 0000-0003-3433-9582
Hussein Harb 0009-0004-9615-6831
Maxime Jacquet 0000-0002-5665-704X
Yihan Jia 0000-0001-8411-0036
George C Kagadis 0000-0002-6983-2863
Han Gyu Kang 0000-0001-5338-4819
Olga Kochebina 0000-0003-4119-4465
Lydia Maigne 0000-0002-0414-8462
Philipp Mohr 0000-0001-9391-3287
Guneet Mummaneni 0009-0006-5745-7249
Valentina Paneta 0000-0002-9485-8993
Panagiotis Papadimitroulas 0000-0002-5981-6149
Alexis Pereda 0000-0002-1434-8230
Axel Rannou 0009-0002-9963-0020
Andreas F Resch 0000-0002-6728-455X
Emilie Roncali 0000-0002-2439-1064
Maxime Toussaint 0000-0003-3402-8186
Carlotta Trigila 0000-0002-0414-9369
Charalampos Tsoumpas 0000-0002-4971-2477
Jing Zhang 0000-0001-5471-3418
Karl Ziemons 0000-0001-5744-7290
Nils Krah 0000-0002-1376-6633

References

Allison J *et al* 2016 Recent developments in Geant4 *Nucl. Instrum. Methods Phys. Res. A* **835** 186–225
Arce P *et al* 2014 Gamos: a framework to do Geant4 simulations in different physics fields with an user-friendly interface *NIMA* **735** 304–13
Arce P *et al* 2021 Report on G4-Med, a Geant4 benchmarking system for medical physics applications developed by the Geant4 Medical Simulation Benchmarking Group *Med. Phys.* **48** 19–56
Arce P *et al* 2025a Results of a Geant4 benchmarking study for bio-medical applications, performed with the G4-Med system *Med. Phys.* **52** 2707–61
Baldacci F *et al* 2014 A track length estimator method for dose calculations in low-energy x-ray irradiations: implementation, properties and performance *Z. Med. Phys.* **25** 36–47
Brun R and Rademakers F 1997 ROOT — an object oriented data analysis framework *Nucl. Instrum. Methods Phys. Res. A* **389** 81–86
Colombino P, Fiscella B and Trossi L 1965 Study of positronium in water and ice from 22 to -144 °C by annihilation quanta measurements *Il Nuovo Cimento* **38** 707–23
Cortés-Giraldo M A and Carabe A 2015 A critical study of different Monte Carlo scoring methods of dose average linear-energy-transfer maps calculated in voxelized geometries irradiated with clinical proton beams *Phys. Med. Biol.* **60** 2645–69
Eckerman K and Endo A 2008 ICRP Publication 107. Nuclear decay data for dosimetric calculations *Ann. ICRP* **38** 7–96
Elazhar H, Deschler T, Létang J M, Nourreddine A and Arbor N 2018 Neutron track length estimator for GATE Monte Carlo dose calculation in radiotherapy *Phys. Med. Biol.* **63** 125018
Faddegon B, Ramos-Méndez J, Schuemann J, McNamara A, Shin J, Perl J and Paganetti H 2020 The TOPAS tool for particle simulation, a Monte Carlo simulation tool for physics, biology and clinical research *Phys. Medica* **72** 114–21

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 Generative adversarial nets *Advances in Neural Information Processing Systems* vol 2 pp 2672–80
Grevillot L, Bertrand D, Dessy F, Freud N and Sarrut D 2011 A Monte Carlo pencil beam scanning model for proton treatment plan simulation using gate/geant4 *Phys. Med. Biol.* **56** 5203
Harris C R *et al* 2020 Array programming with NumPy *Nature* **585** 357–62
Huang B and Qi J 2024 High-resolution positronium lifetime tomography by the method of moments *Phys. Med. Biol.* **69** 24NT01
Inaniwa T, Furukawa T, Kase Y, Matsufuji N, Toshito T, Matsumoto Y, Furusawa Y and Noda K 2010 Treatment planning for a scanned carbon beam with a modified microdosimetric kinetic model *Phys. Med. Biol.* **55** 6721–37
Incerti S *et al* 2010 The Geant4-DNA project *Int. J. Modeling, Simulat. Sci. Comput.* **01** 157–78
Jan S *et al* 2004 GATE: a simulation toolkit for PET and SPECT *Phys. Med. Biol.* **49** 4543–61
Jan S *et al* 2011 GATE V6: a major enhancement of the GATE simulation platform enabling modelling of CT and radiotherapy *Phys. Med. Biol.* **56** 881–901
Kim C H *et al* 2020 Icrp publication 145: adult mesh-type reference computational phantoms *Ann. ICRP* **49** 13–201
Kochebina O, Bonifacio D A B, Konstantinou G, Paillet A, Pommranz C M, Razdevšek G, Sharyy V, Yvon D and Jan S 2024 New gate digitizer unit for versions post v9.3 *Front. Phys.* **12** 1294916
Krah N *et al* 2025 GATE 10 Monte Carlo particle transport simulation – Part II: architecture and innovations *Phys. Med. Biol.* (https://doi.org/10.1088/1361-6560/ae237c)
Krämer M and Scholz M 2006 Rapid calculation of biological effects in ion radiotherapy *Phys. Med. Biol.* **51** 1959–70
Lab R. 2025 Roncali lab github repository (available at:https://github.com/roncalilab)
McCormick M, Liu X, Ibanez L, Jomier J and Marion C 2014 ITK: Enabling reproducible research and open science *Front. Neuroinform.* **8** 13
Mercolli L *et al* 2025 *In vivo* intralesional positronium lifetime imaging of thyroid cancer using clinically routine i-124 pet/ct *medRxiv* (https://doi.org/10.1101/2025.05.28.25328504)
Moskal P *et al* 2021 Positronium imaging with the novel multiphoton PET scanner *Sci. Adv.* **7** eabh4394
Moskal P *et al* 2024 Positronium image of the human brain *in vivo Sci. Adv.* **10** eadp2840
Mummaneni G, Trigila C, Krah N, Sarrut D and Roncali E 2025 Deep learning-based optical photon transport with optigan: An alternative to photon tracking in gate 10 *Phys. Med. Biol.* submitted
Noblet C, Chiavassa S, Smekens F, Sarrut D, Passal V, Suhard J, Lisbona A, Paris F and Delpon G 2016 Validation of fast Monte Carlo dose calculation in small animal radiotherapy with EBT3 radiochromic films *Phys. Med. Biol.* **61** 3521–35
Paszke A *et al* 2019 PyTorch: an imperative style, high-performance deep learning library *NEURIPS 2019* p 12
Pivarski J *et al* 2020 scikit-hep/uproot: 3.12.0 *Zenodo* (https://doi.org/10.5281/zenodo.3952728)
Qi J and Huang B 2022 Positronium lifetime image reconstruction for tof pet *IEEE Trans. Med. Imaging* **41** 2848–55
Saporta A, Etxebeste A, Krah N, Létang J M and Sarrut D 2022 Conditional GAN for Monte Carlo SPECT simulation *Int. Conf. on Monte Carlo Techniques for Medical Applications (MCMA 2022)* (*Antwerpen, Belgium*)
Sarrut D *et al* 2014 A review of the use and potential of the GATE Monte Carlo simulation code for radiation therapy and dosimetry applications: GATE for dosimetry *Med. Phys.* **41** 064301
Sarrut D *et al* 2021a Advanced Monte Carlo simulations of emission tomography imaging systems with GATE *Phys. Med. Biol.* **66** 10TR03
Sarrut D *et al* 2022 The OpenGATE ecosystem for Monte Carlo simulation in medical physics *Phys. Med. Biol.* **67** 184001
Sarrut D, Etxebeste A, Krah N and Létang J-M 2021b Modeling complex particles phase space with GAN for Monte Carlo SPECT simulations: a proof of concept *Phys. Med. Biol.* **66** 055014
Sarrut D, Etxebeste A and Létang J M 2024 A photon source model for alpha-emitter radionuclides *Phys. Med. Biol.* **69** 095009
Sarrut D, Etxebeste A, Muñoz E, Krah N and Létang J M 2021c Artificial intelligence for Monte Carlo simulation in medical physics *Front. Phys.* **9** 601
Sarrut D and Guigues L 2008 Region-oriented CT image representation for reducing computing time of Monte Carlo simulations: Voxelized geometry with GEANT4 *Med. Phys.* **35** 1452–63
Sarrut D, Krah N and Létang J M 2019 Generative adversarial networks (GAN) for compact beam source modelling in Monte Carlo simulations *Phys. Med. Biol.* **64** 215004
Schling B 2011 *The Boost C++ Libraries* (XML Press)
Sechopoulos I, Rogers D W O, Bazalova-Carter M, Bolch W E, Heath E C, McNitt-Gray M F, Sempau J and Williamson J F 2018 RECORDS: improved reporting of montE CarlO RaDiation transport studies: report of the AAPM Research Committee Task Group 268 *Med. Phys.* **45** e1–e5
Shibuya K, Saito H, Nishikido F, Takahashi M and Yamaya T 2020 Oxygen sensing ability of positronium atom for tumor hypoxia imaging *Commun. Phys.* **3** 1–8
Shibuya K, Yoshida E, Nishikido F, Suzuki T, Tsuda T, Inadama N, Yamaya T and Murayama H 2007 Annihilation photon acollinearity in PET: volunteer and phantom FDG studies *Phys. Med. Biol.* **52** 5249
Smekens F, Létang J M, Noblet C, Chiavassa S, Delpon G, Freud N, Rit S and Sarrut D 2014 Split exponential track length estimator for Monte-Carlo simulations of small-animal radiation therapy *Phys. Med. Biol.* **59** 7703–15
Srikanth A, Trigila C and Roncali E 2024 GPU optimization techniques to accelerate optiGAN-a particle simulation GAN *Mach. Learn.: Sci. Technol.* **5** 027001
Steinberger W M *et al* 2024 Positronium lifetime validation measurements using a long-axial field-of-view positron emission tomography scanner *EJNMMI Phys.* **11** 76
Tashima H and Yamaya T 2024 Three-gamma imaging in nuclear medicine: a review *IEEE Trans. Radiat. Plasma Med. Sci.* **8** 853–66
Toussaint M, Loignon-Houle F, Auger E, Lapointe G, Dussault J-P and Lecomte R 2024 On the implementation of acollinearity in PET Monte Carlo simulations *Phys. Med. Biol.* **69** 18NT01
Trigila C, Srikanth A and Roncali E 2023 A generative adversarial network to speed up optical Monte Carlo simulations *Mach. Learn.: Sci. Technol.* **4** 025005