EPJ Web of Conferences **245**, 11003 (2020) https://doi.org/10.1051/epjconf/202024511003
CHEP 2019

Meeting the challenge of JUNO simulation with Opticks: GPU optical photon acceleration via NVIDIA® OptiX™

Simon Blyth[1,*]

[1]Institute of High Energy Physics, CAS, Beijing, China.

Abstract. Opticks is an open source project that accelerates optical photon simulation by integrating NVIDIA GPU ray tracing, accessed via NVIDIA OptiX, with Geant4 toolkit based simulations. A single NVIDIA Turing architecture GPU has been measured to provide optical photon simulation speedup factors exceeding 1500 times single threaded Geant4 with a full JUNO analytic GPU geometry automatically translated from the Geant4 geometry. Optical physics processes of scattering, absorption, scintillator reemission and boundary processes are implemented within CUDA OptiX programs based on the Geant4 implementations. Wavelength-dependent material and surface properties as well as inverse cumulative distribution functions for reemission are interleaved into GPU textures providing fast interpolated property lookup or wavelength generation. Major recent developments enable Opticks to benefit from ray trace dedicated RT cores available in NVIDIA RTX series GPUs. Results of extensive validation tests are presented.

1 Introduction

Opticks[1-5] enables Geant4[6-8]-based optical photon simulations to benefit from high performance GPU ray tracing made accessible by NVIDIA® OptiX™[9-11]. The Jiangmen Underground Neutrino Observatory (JUNO)[12] under construction in southeast China features the world's largest liquid scintillator detector, with a 20 kton spherical volume of 35 m diameter. The large size and high photon yield of the scintillator, illustrated in Figure 1, makes the JUNO optical photon simulation extremely computationally challenging with regard to both processing time and memory resources. Opticks eliminates both these bottlenecks by offloading the optical photon simulation to the GPU. Recent Opticks developments allow the optical photon simulation performance to benefit from ray trace dedicated processors, called RT cores[13], available in NVIDIA Turing architecture GPUs. Monte Carlo simulation is the primary technique used to design, optimize and analyse diverse detection systems. However, sequential simulation of large numbers of optical photons has extreme computational and memory costs. Opticks enables drastically improved optical photon simulation performance that can be transformative to the design, operation and understanding of diverse optical systems. Although Opticks was developed for the simulation of the JUNO detector, it is structured to enable use with other detector geometries. Any detector simulation limited by optical photons can benefit from Opticks. Several groups from various neutrino experiments and dark matter search experiments are evaluating Opticks.

*Corresponding author and speaker on behalf of the JUNO collaboration. e-mail: simon.c.blyth@gmail.com.

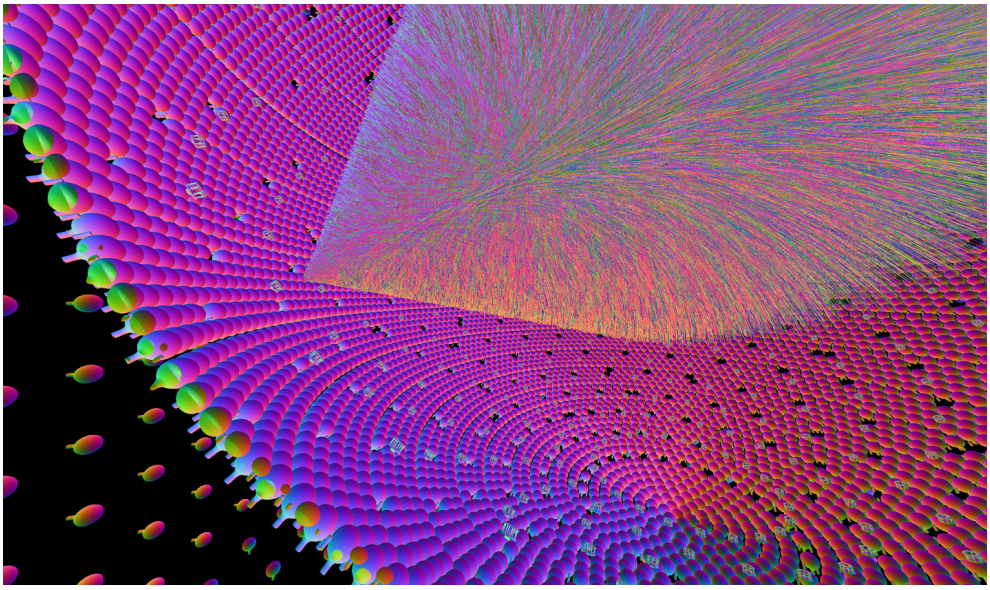

Figure 1. Cutaway OpenGL rendering of millions of simulated optical photons from a 200 GeV muon crossing the JUNO liquid scintillator. Each line corresponds to a single photon with line colors representing the polarization direction. Primary particles are simulated by Geant4, scintillation and Cerenkov "gensteps" are uploaded to the GPU and photons are generated, propagated and visualized all on the GPU. Representations of some of the many thousands of photomultiplier tubes that instrument the liquid scintillator are visible. The acrylic vessel that contains the liquid scintillator is not shown.

These proceedings give an overview of Opticks, describe its application to the JUNO detector and present performance measurements and validation tests. Further details on the implementation of automated geometry translation are available in the prior proceedings[4]. Further details on NVIDIA OptiX, the use of GPU textures and the CUDA port of Geant4 photon generation and optical physics are available in the earlier proceedings[5].

1.1 Importance of optical photon simulation to JUNO

Cosmic muon induced processes are crucial backgrounds for neutrino detectors such as JUNO[12], necessitating underground sites, water shields and muon veto systems. Minimizing the dead time and dead volume that result from applying a veto requires an understanding of the detector response to a muon. Large simulated samples of muon events are crucial in order to develop such an understanding. The number of optical photons estimated to be produced by a muon of typical energy 200 GeV crossing the JUNO scintillator is at the level of tens of millions. Profiling the Geant4-toolkit-based simulation shows that the optical photon propagation consumes more than 99% of CPU time, and imposes severe memory constraints that have forced the use of event splitting. As optical photons in neutrino detectors can be considered to be produced by only the scintillation and Cerenkov processes and yield only hits on photomultiplier tubes, it is straightforward to integrate an external optical photon simulation with a Geant4 simulation of all other particles.

1.2 GPU ray tracing

Graphics Processing Units (GPUs) evolved to meet the needs of computer graphics, optimizing throughput[14], the total amount of work completed per unit time. This contrasts with CPUs which minimize latency, the time to complete a single task. The relative simplicity of GPUs allows more chip area to be dedicated to parallel compute, resulting in greater computational throughput across all threads at the expense of slower single-thread execution. Threads blocked while waiting to access memory are tolerated by using hardware multithreading to resume other unblocked threads, allowing latencies to be hidden assuming there are sufficient parallel threads in flight. Effective use of GPUs demands problems with many thousands of mostly independent parallel tasks, each of which requires few resources, allowing more threads to be in flight simultaneously. Optical photon simulation is well matched to these requirements with abundant parallelism from the large numbers of photons and low register usage from the simplicity of optical physics.

The most computationally demanding aspect of optical photon simulation is the calculation, at each step of the propagation, of intersection positions of rays representing photons with the geometry of the system. This ray tracing limitation of optical photon simulation is shared with the synthesis of realistic images in computer graphics. Due to the many applications of ray tracing in the advertising, design, games and film industries, the computer graphics community has continuously improved ray tracing techniques. The Turing GPU architecture introduced by NVIDIA in 2018 is marketed as the world's first Ray-tracing GPU, with hardware "RT Cores" dedicated to the acceleration of ray geometry intersection. NVIDIA claims performance of more than 10 billion ray intersections per second, which is a factor 10 more than possible with earlier GPUs which perform the intersection acceleration in software.

1.3 NVIDIA® OptiX™ ray tracing engine

OptiX is a general-purpose ray tracing engine designed for NVIDIA GPUs that exposes an accessible single ray programming model. The core of OptiX is a domain-specific just-in-time compiler that constructs ray tracing pipelines, combining code for acceleration structure creation and traversal, together with user provided CUDA code for ray generation, object intersection and closest hit handling. Spatial index data structures, such as the boundary volume hierarchy (BVH), are the principal technique for accelerating ray geometry intersection. OptiX provides only the acceleration of ray geometry intersection, not the intersection itself, thus affording full flexibility to implement intersections with any form of geometry. OptiX acceleration structures supports instancing, allowing them to be shared between multiple instances of the same geometry such as the photomultiplier tubes in the JUNO geometry.

2 Hybrid simulation workflow

Implementing an efficient GPU optical photon simulation equivalent to the Geant4 simulation requires that all aspects of the Geant4 context relevant to optical photon generation and propagation are translated into an appropriate form and uploaded to the GPU. The primary aspects are the detector geometry including material/surface properties, optical physics and optical photons; the translations of these are described in the below sections.

Figure 2 provides an overview of the hybrid simulation workflow. A single class, `G4Opticks`, is used to provide a minimal interface between Geant4 user code and the Opticks package. At initialization the Geant4 top volume pointer is passed to Opticks which translates the geometry and constructs the OptiX GPU context.

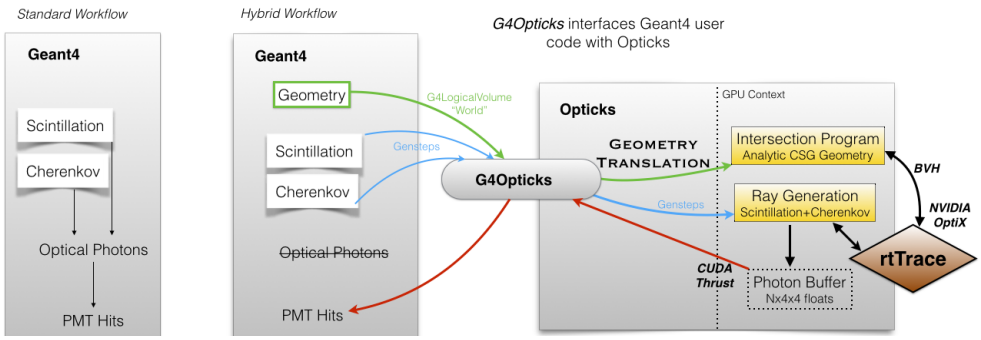

Figure 2. Comparison of the standard workflow of Geant4 optical photon simulation (left) with the hybrid Geant4 + Opticks workflow (right). A single Opticks class `G4Opticks` acts to interface Geant4 user code with the Opticks GPU propagation. Hybrid simulation requires modification of the classes representing scintillation and Cherenkov processes to collect "genstep" data structures.

The simulation is steered from the NVIDIA OptiX ray generation program which performs the photon generation followed by propagation up to a configurable maximum number of steps. For each step of the propagation, rays representing photons are intersected with the geometry. The intersected boundary provides a boundary index which allows material properties such as absorption and scattering lengths to be looked up. Converting these lengths to distances using pseudorandom numbers and the known exponential distributions allows a comparison of absorption and scattering distances with geometrical distance to boundary to assign photon histories. Details on efficient use of pseudorandom number generation with cuRAND[15] are in the earlier proceedings[5].

3 Detector geometry

Detector geometry is modelled on the GPU via intersection, bounding box and closest hit programs and buffers that these programs access. Opticks provides automated translation of Geant4 geometries into these buffers, starting by traversing the Geant4 volume tree converting materials, surfaces, solids, volumes and sensors into Opticks equivalents. For large detector geometries with many thousands of volumes, such as the JUNO geometry, this translation can take several minutes. To avoid repeating this processing, a serialization of the geometry termed the "geocache" is implemented to allow GPU ready arrays to be persisted into NumPy[16] binary files. On subsequent runs, these NumPy binary files are loaded and uploaded to the GPU, allowing OpenGL visualizations and test executables to initialize full geometries in seconds rather than minutes.

3.1 Material and surface properties

Material and surface properties as a function of wavelength are interpolated onto a common wavelength domain. The properties include refractive indices, absorption lengths, scattering lengths, reemission probabilities, as well as surface reflectivities, detection efficiencies and absorption fractions. Each volume of the geometry is assigned a boundary index uniquely identifying the combination of four indices representing outer and inner materials and outer and inner surfaces. Outer/inner surfaces handle inwards/outwards going photons, which allows the Geant4 border and skin surface functionality to be translated.

GPUs contain hardware dedicated to fast texture lookup and interpolation. This is exploited by using a single 2D `float4` texture, named the boundary texture, that contains interleaved material and surface properties as a function of wavelength for all unique boundaries. The boundary index returned from a ray traced primitive intersection, together with an orientation offset identified from the angle between the geometric normal and ray direction, enables four wavelength interpolated material or surface properties to be obtained from a single hardware optimized texture lookup.

3.2 Solid shapes

Opticks provides CUDA functions that return ray intersections for ten primitive shapes including sphere, hyperboloid and torus. These functions use implicit equations for the primitives together with the parametric ray equation, to yield a polynomial in t, the distance along the ray from its origin position. Roots of the polynomials provide intersections, and surface normals at intersects are obtained using the derivative of the implicit equation. Arbitrarily complex solids are described using constructive solid geometry (CSG) modelling, which builds shapes from the combination of primitive constituents by boolean set operations: union, intersection and difference. A binary tree data structure with primitives at the leaves of the tree and operators at the internal nodes is used to represent the solids. Any node can have an associated local transform, represented by a 4x4 transformation matrix, which is combined with other local transforms to yield global transforms in the frame of the root node of the tree. Each primitive or operator node is serialized into an array of up to 16 elements. These elements include float parameters of the primitives and integer index references into a separate global transform buffer. A complete binary tree serialization with array indices matching level order tree indices and zeros at missing nodes is used for the serialization of the CSG trees. This simple serialization allows tree navigation directly from bitwise manipulations of the serialized array index. Complete binary tree serialization is simple and effective for small trees but very inefficient for unbalanced trees, necessitating tree balancing for shapes with many constituent primitives to reduce the tree height. The prior proceedings[4] provide further details of the constructive solid geometry modelling, tree balancing and translation between Geant4 and Opticks solids.

3.3 Structural volumes

The Opticks geometry model is based upon the observation that many elements of a detector geometry are repeated demanding the use of instancing for efficient representation. Geometry instancing is a technique used in computer graphics libraries including OpenGL and NVIDIA OptiX that avoids duplication of information on the GPU by storing repeated elements only once together with 4x4 transform matrices that specify the locations and orientations of each instance. The Geant4 geometry model comprises a hierarchy of volumes with associated transforms. A geometry digest string for every structure node is formed from the transforms and shape indices of the progeny nodes descended from the original node in the geometry tree. Subsequently repeated groups of volumes and their placement transforms are identified using the geometry digests, after disqualifying repeats that are contained within other repeats. All structure nodes that pass instancing criteria regarding the number of vertices and number of repeats are assigned an instance index with the remainder forming the global non-instanced group. These groups of volumes are then used for the creation of the NVIDIA OptiX analytic geometry instances, and OpenGL mesh geometry instances. Figure 3 presents a ray traced rendering of the analytic representation of the JUNO detector geometry.

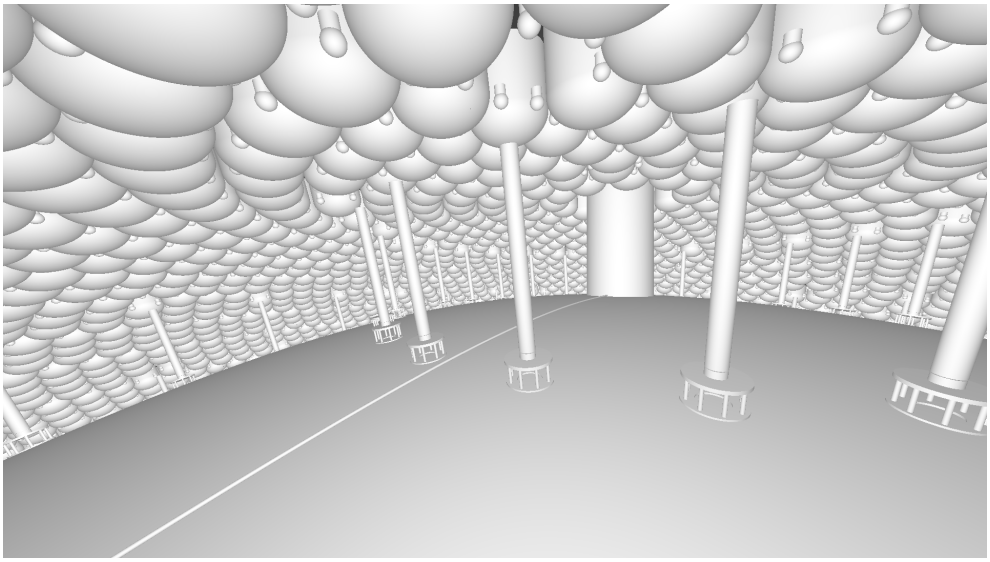

Figure 3. Ray traced rendering of the chimney region of JUNO detector showing photomultiplier tubes, acrylic sphere, supports and the calibration guide tube torus. The rendering uses exactly the same analytic geometry as the simulation. The geometry was directly converted from Geant4 into an Opticks geometry including analytic CSG and persisted into a geometry cache of NumPy[16] binary files.

4 Optical physics

Optical physics processes of scattering, absorption, scintillator reemission and boundary processes are implemented in CUDA functions based on the Geant4 implementations. The single ray programming model exposed by NVIDIA OptiX makes it mostly possible to directly port the corresponding Geant4 implementation by simply adapting to use GPU textures for property access. On the CPU, it is convenient to implement scintillator reemission using Geant4 secondary tracks. A different approach is adopted on the GPU where a fraction of absorbed photons are reborn with modified direction and wavelength within the same CUDA thread. A reemission texture that encapsulates an inverse cumulative distribution function is used to generate wavelengths that follow the desired distribution on lookup of pseudorandom numbers.

5 Optical photons and "gensteps"

Photons are brought to the GPU via NVIDIA OptiX ray generation programs, CUDA ports of photon generation loops and photon generation parameters in the form of "genstep" data structures, which play a central role in the workflow. As Geant4 has no "genstep" interface, it is necessary to modify the classes representing scintillation and Cerenkov processes. Instead of generating photon secondary tracks in a loop, the "genstep" parameters such as the process type code, the number of photons to generate and the line segment along which to generate them and all other parameters used in the generation loop are collected. Collecting and copying gensteps rather than photons avoids allocation of CPU memory for the photons, only photons that reach sensors requiring CPU memory allocation. The genstep arrays are typically several orders of magnitude smaller than the photon arrays that are generated from

them. Gensteps can be regarded as a manual collection of the "stack" just prior to the generation loop, but they are valid only for a specific version of the processes due to the need for a matched CUDA port of the generation loop.

6 Random number aligned comparison of Opticks and Geant4

Validation comparisons use a single executable that performs both the Geant4 and hybrid Opticks simulations, starting from common CPU generated input photons. Copying cuRAND random sequences from the GPU to the CPU and configuring the Geant4 random engine to use them makes it possible to align the consumption of random numbers between the two simulations, resulting in nearly perfectly matched results with every scatter, absorption and reflection happening with the same positions, times, wavelengths and polarizations. Direct comparison of the aligned simulation results allows any discrepancies to be identified without being clouded by statistics. The executable writes two events in a format which includes highly compressed positions, times, wavelengths and polarizations at up to 16 steps of the optical photon propagations.

Checks of all JUNO solids with millions of photons revealed some spurious intersects arising from fragile CSG modelling with constituent solids that had coincident faces. These were fixed by straightforward modelling changes to avoid coincidences. Shapes including the torus as a constituent were found to be prone to poor precision intersects. As the use of torus was cosmetic, the modelling was simplified to avoid its use. After fixing these geometry issues, remaining discrepancies in mis-aligned photon histories were <0.25% and deviant photons within matched histories were <0.05%. Primary sources of discrepancies are photons with grazing incidence or photons incident at constituent solid boundaries. Results in these cases are expected to depend on the arithmetic precision: Opticks uses double precision only where unavoidable, whereas Geant4 uses this everywhere.

7 Performance comparisons

Optical photon simulation performance with the full analytic JUNO geometry is measured using calibration source gensteps, positioned at the center of the scintillator volume, that uniformly emit a range of photon counts. The maximum number of optical photons that can be simulated in a single GPU launch is limited by the available VRAM. Each photon requires 64 bytes for parameters and 48 bytes for the CURAND random number generator state, corresponding to 45G for 400M photons. The measurements use a production mode with only photomultiplier hits being stored. All non-essential processing such as photon step recording are skipped. Test hardware was a single NVIDIA Quadro RTX 8000 GPU with 48G of VRAM hosted in a DELL Precision 7920T workstation with Intel Xeon Gold 5118, 2.3GHz, 62G CPU. Figure 4 shows results from a scan from 1M to 400M optical photons, where the measured Opticks times are compared with Geant4 times linearly extrapolated from a measurement at 1M photons. Comparing times with the RTX mode enabled and disabled indicates a speedup of approximately 5 times from the use of the ray trace dedicated RT cores. The single GPU speedup factor between Opticks with RTX enabled and single threaded Geant4 is measured to be 1,660.

Performance measurements with very simple analytic geometries are found to reach Giga Rays/s, more than a factor of 10 faster than performance with the full JUNO analytic geometry. This great performance sensitivity to the geometry suggests there is potential for substantial improvement by optimization of geometry modelling.

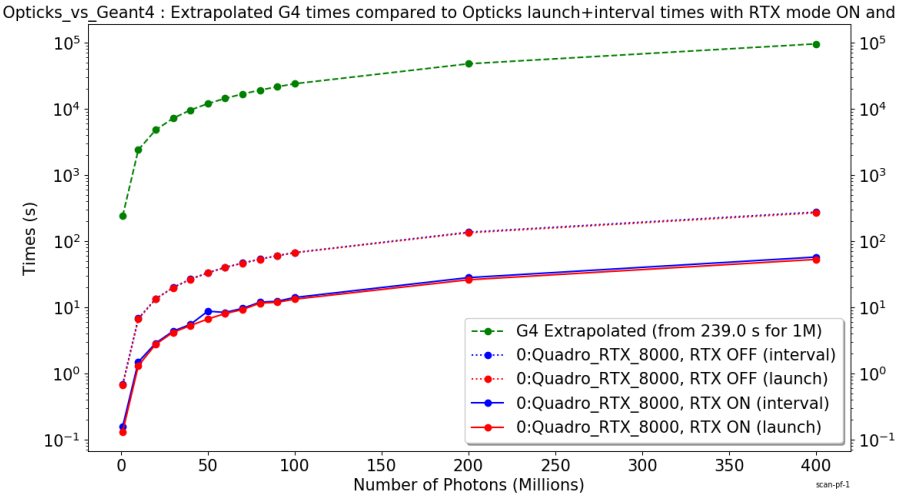

Figure 4. Full analytic JUNO geometry Opticks simulation times in seconds for 1M-400M optical photons using a single NVIDIA Quadro RTX 8000 GPU are compared with single threaded Geant4 10.4.2 simulation times extrapolated from a measurement for 1M optical photons. The solid(dotted) blue and red curves show times with RTX mode enabled(disabled). Differences between interval times which include per event upload and download overheads and launch times are not readily apparent with the logarithmic scale. The linearly extrapolated Geant4 time for 400M photons is 95,600 s (26 hours) contrasts with the Opticks time of 58s, corresponding to a speedup factor of 1,660 times with a single GPU.

8 Summary

Opticks enables Geant4-based optical photon simulations to benefit from state-of-the-art NVIDIA GPU ray tracing, made accessible via NVIDIA OptiX, that allows memory and time processing bottlenecks to be eliminated. Recent developments enable Opticks to benefit from the ray trace dedicated hardware available in Turing architecture GPUs. Opticks meets the challenge of optical photon simulation in JUNO, the world's largest scintillator detector, and can benefit any simulation limited by optical photons.

Acknowledgements

The JUNO collaboration is acknowledged for the use of detector geometries and simulation software. Dr. Tao Lin is acknowledged for his assistance with the JUNO offline software. This work is funded by Chinese Academy of Sciences President's International Fellowship Initiative, Grant No. 2018VMB0002.

References

[1] Opticks Repository, https://bitbucket.org/simoncblyth/opticks/
[2] Opticks References, https://simoncblyth.bitbucket.io
[3] Opticks Group, https://groups.io/g/opticks
[4] S. Blyth, EPJ Web Conf. **214**, 02027 (2019)
https://doi.org/10.1051/epjconf/201921402027
[5] Blyth Simon C 2017 J. Phys.: Conf. Ser. **898** 042001
https://doi.org/10.1088/1742-6596/898/4/042001
[6] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce et al., Nucl. Instrum. Methods. Phys. Res. A **506**, 250 (2003)
[7] J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Dubois, M. Asai et al., IEEE Trans Nucl Sci, **53**, 270 (2006)
[8] J. Allison, K. Amako, J. Apostolakis, P. Arce, M. Asai, T. Aso et al., Nucl. Instrum. Methods. Phys. Res. A **835**, 186 (2016)
[9] OptiX: a general purpose ray tracing engine
S. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock et al., ACM Trans. Graph.: Conf. Series **29**, 66 (2010)
[10] OptiX introduction, https://developer.nvidia.com/optix
[11] OptiX API, http://raytracing-docs.nvidia.com/optix/index.html
[12] Neutrino physics with JUNO
F. An et al., J. Phys. G. **43**, 030401 (2016)
[13] NVIDIA RTX, https://developer.nvidia.com/rtx
[14] Understanding Throughput Oriented Architectures
M. Garland, D.B. Kirk, Commun. ACM **53**(11), 58 (2010)
[15] cuRAND, http://docs.nvidia.com/cuda/curand/index.html
[16] The NumPy array: a structure for efficient numerical computation
S. Van der Walt, S. Colbert, G. Varoquaux, Comput. Sci. Eng. **13**, 22 (2011)