ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAG21

NOVEL, FAST, OPEN-SOURCE CODE FOR SYNCHROTRON RADIATION COMPUTATION ON ARBITRARY 3D GEOMETRIES

Dean Andrew Hidas*, Brookhaven National Laboratory, Upton, NY, USA

Abstract

Open Source Code for Advanced Radiation Simulation (OSCARS) is an open-source project (https://oscars.bnl.gov) developed at Brookhaven National Laboratory for the computation of synchrotron radiation from arbitrary charged particle beams in arbitrary and time-dependent magnetic and electric fields on arbitrary geometries in 3D. Computational speed is significantly increased with the use of built-in multi-GPU and multi-threaded techniques which are suitable for both small scale and large scale computing infrastructures. OSCARS is capable of computing spectra, flux, and power densities on simple surfaces as well as on objects imported from common CAD software. It is additionally applicable in the regime of high-field acceleration. The methodology behind OSCARS calculations will be discussed along with practical examples and applications to modern accelerators and light sources.

INTRODUCTION

OSCARS [1] is an open source software developed at Brookhaven National Laboratory (BNL). OSCARS is a general purpose code for the calculation of radiation from charged particles in motion. Primary uses are for synchrotron and accelerator facilities where photon density distributions and heat loads on accelerator and beam-line equipment is of interest. This software allows for the calculation of these properties on arbitrary shaped surfaces in 3 dimensions. Recently added features include the ability to import surface models directly in the common Computer Aided Design (CAD) STL format and the implementation of of time dependent magnetic and electric source fields.

The core code is written in modern C++ for speed and has a python extension which serves as the main application programming interface (API). The API was written in python for ease of use and integrability by the larger scientific community. Currently OSCARS is available for Python 2.7 and Python 3+, for Linux, OS X, and Windows operating systems.

Throughout this paper the 2 example undulators referred to (U49 and EPU49) have a period of 49 mm, 55 periods plus terminating fields, and a peak magnetic field of 1 Tesla. U49 is a planar undulator with the only non-zero component of the magnetic field $\vec{B}$ being a sinusoidally varying B_v (vertical component). EPU49 is an elliptically polarizing undulator where B_h (horizontal component) and B_v are sinusoidally varying with the same peak magnitude, but phase shifted by $\pi/2$ relative to each other while B_l (longitudinal component) remains zero. The beam parameters used in these simulations are that of the NSLS-II 6.6 m straight sections with a beam energy $E = 3$ GeV, energy spread $\Delta E/E = 0.001$, emittance $\epsilon_{\mathrm{h,v}} = [0.9, 0.008]$ nm rad, and beta function in the center $\beta_{\mathrm{h,v}} = [1.5, 0.8]$ m.

* dhidas@bnl.gov

PARTICLE BEAMS

Particle beams in OSCARS are defined by particle mass, energy, current, and their position and direction. Optionally one can include the emittance ($\epsilon_{\mathrm{h,v}}$), beta function ($\beta_{\mathrm{h,v}}$), and energy spread ($\Delta E/E$). In multi-particle simulations the energy, initial position, and initial momentum are sampled accordingly assuming a Gaussian distribution of the position (σ) and momentum (σ'). OSCARS also allows for the definition of multiple beams which are sampled randomly according to their relative weights given. Any beam direction is valid in OSCARS. For convenience several predefined beams exist in OSCARS, for example in this case "NSLSII-ShortStraight".

MAGNETIC AND ELECTRIC FIELDS

Several field types (applicable to both magnetic and electric fields) are available in OSCARS. These source fields may be time dependent, static, or any combination therein. Simple parameters exist for configuring any static field as a time-dependent sinusoidal resonant field with a phase offset. Some common built-in fields include uniform fields in a given range, Gaussian fields, sinusoidal undulator (wiggler) fields with terminating fields, user input python functions, and 1D, 2D, and 3D discrete field data. In the case of the 1D discrete field data, the field points do not need to be uniformly distributed in space as OSCARS will regularize it by interpolation for internal use and fast internal field lookup. OSCARS also has an interpolating tool (using a cubic spline method) in the case that you have field data measured at several parameter points (such as undulator gap, but not restricted to this) but wish to use the field at an intermediate point. Several input data formats are implemented for convenience.

CALCULATION OF ELECTRIC FIELD

Particle trajectories are calculated in 3D according to the relativistic Lorentz equation given in Eq. (1) using a fourth-order Runge-Kutta (RK4) algorithm or an optional adaptive step RK4 method. Care is taken in the RK4 method implementation to avoid $\beta \geq 1$ by an iterative step-halving method until the criteria $\beta < 1$ is satisfied in for the trajectory propagation of Eq. (1):

$$\frac{\mathrm{d}\vec{p}}{\mathrm{d}t} = q(\vec{E} + c\vec{\beta} \times \vec{B}). \quad (1)$$

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAG21

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

The source electric and magnetic fields in Eq. (1) need not be static fields, *e.g.* $\vec{E} = \vec{E}(t)$ and $\vec{B} = \vec{B}(t)$. The electric field in the frequency domain is given in Eq. (2) which is derived by taking a Fourier transform of the Liéard-Wiechert potentials and given in many texts, for example [2–4]. Equation (2) is valid for relativistic and non-relativistic particles as well as in the near-field:

$$\vec{E}(\vec{x},\omega)/C = \frac{1}{\gamma^2}\int_{-\infty}^{+\infty}\frac{\hat{n}-\vec{\beta}}{D^2(1-\hat{n}\cdot\vec{\beta})^2}e^{i\omega(\tau+D/c)}\mathrm{d}\tau \quad (2)$$
$$+\int_{-\infty}^{+\infty}\frac{\hat{n}\times(\hat{n}-\vec{\beta})\times\dot{\vec{\beta}}}{R(1-\hat{n}\cdot\vec{\beta})^2}e^{i\omega(\tau+D/c)}\mathrm{d}\tau.$$

Here, $C = \frac{2I}{\hbar q\mu_0 c}$, $\vec{p}$ is the particle momentum, τ the laboratory time, q particle charge, $\vec{E}$ electric field, c speed of light in vacuum, $\vec{\beta}$ the particle velocity divided by c, $\vec{B}$ the magnetic field, and $\vec{x}$ the observation point. ω is the photon angular frequency of interest, ϵ_0 the permittivity of free space, $\hat{n}$ a unit vector in the direction from the particle to the observation point, and D the distance between the particle and the observation point.

For the calculation of spectra and flux OSCARS performs a numerical integration of Eq. (2) once the trajectory has been calculated using Eq. (1). This is done either on the Central Processing Unit (CPU) or GPU depending on which mode is selected. The spectrum in Fig. 1 and flux densities in Fig. 2 use this method. An interpolating method is used to calculate trajectory points on a regularized grid from the results of the RK4 propagation. Each successive integration step (for a flux, spectrum, or power density calculation) uses $2N+1$ trajectory integration points until the desired relative precision is reached (typically 1%, but the user is free to specify any alternative).

POWER DENSITY

Once the trajectory is calculated, the power density can be calculated from Eq. (3):

$$P(\vec{x}) = \frac{qI}{16\pi^2\epsilon_0 c}\int_{-\infty}^{\infty}\frac{\vec{n}\times((\vec{n}-\vec{\beta})\times\frac{\vec{a}}{c})}{(1-\vec{\beta}\cdot\vec{n})^5}\frac{1}{D^2}(\hat{n}\cdot\hat{S})\,\mathrm{d}\tau \quad (3)$$

where I is the beam current, $\vec{a}$ the particle acceleration and $\hat{S}$ a unit normal vector for the surface point at position $\vec{x}$. Here again, the numerical integration is performed either on the CPU or the GPU if desired.

The power density distribution in the x-y plane 30 m downstream from the example U49 is shown in Fig. 3. OSCARS allows for the calculation of power density distributions on arbitrary objects in 3D. Several examples of this are shown in Fig. 4. These were calculated using the GPU option available in OSCARS, but can as well easily be computed on a single-core, in multi-threaded mode, or using grid/cloud computing.

Power density distributions can also be calculated on 3D surfaces designed in common CAD software. OSCARS accepts the STL (stereo lithography) file format. Ray-tracing

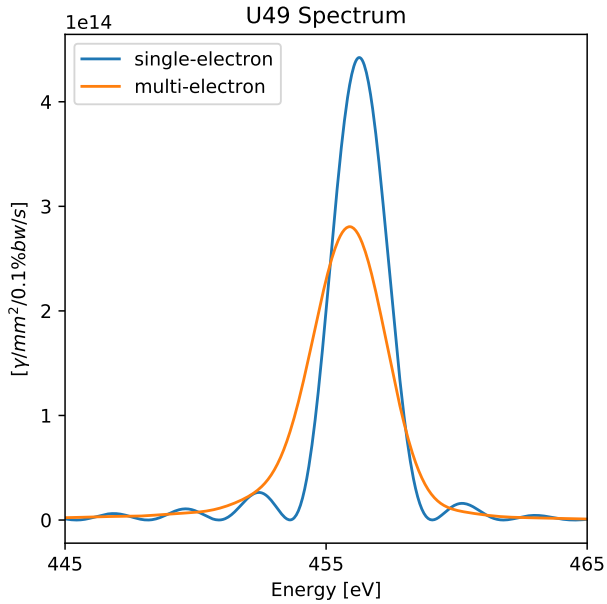

Figure 1: On-axis spectrum for the simulated U49 undulator 30 m downstream showing the 3rd harmonic for a filament beam (single-electron), and realistic beam using the NSLS-II design parameters for a 6.6 m straight section (multi-electron).

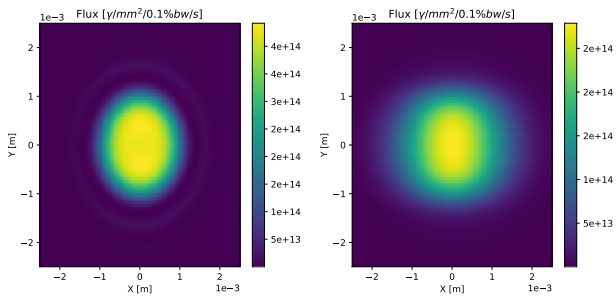

Figure 2: Flux for the simulated U49 undulator 30 m downstream showing the 3rd harmonic for a filament beam (left) and realistic beam (right) using the NSLS-II design parameters for a 6.6 m straight section.

is performed in the far-field approximation in such a way that the radiation will be blocked at the first surface it encounters (*i.e. shadow effect*). This will lend itself well to the investigation of very complex objects within reach of high intensity beams from synchrotron radiation. A simple example of this effect is shown in Fig. 5 where two spheres of different sizes are shown with the U49 source 30 meters upstream.

As an example of OSCARS time dependent source field calculations Fig. 6 shows the β_z response of a point-charge bunch in a strong longitudinal electric field. The power density 1 m downstream of this is shown in Fig. 7. Although of limited interest at most accelerators these quantities (spectra, flux, power) can be calculated for extremely high field gradients with time dependence.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAG21

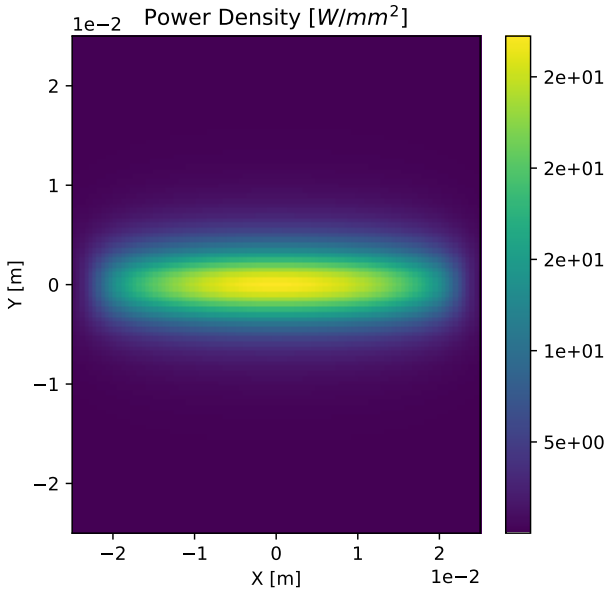

Figure 3: Power density from U49 as seen 30 m downstream from the source.

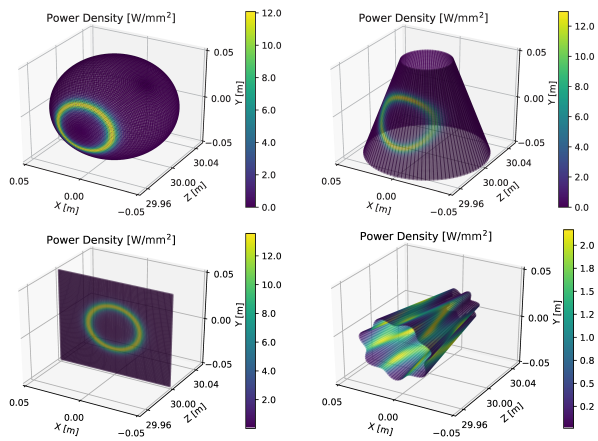

Figure 4: Power density on various 3D shapes for the simulated EPU49 undulator 30 m downstream.

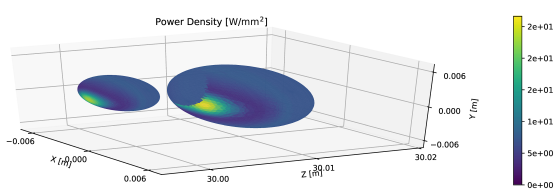

Figure 5: Power density on 2 spheres produced with CAD software and imported into OSCARS in the STL file format.

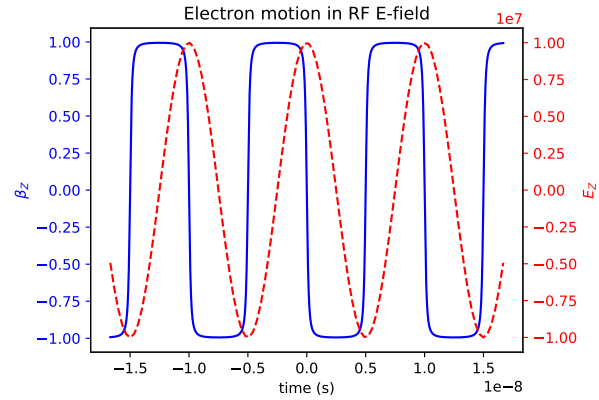

Figure 6: High gradient acceleration of a point-charge bunch.

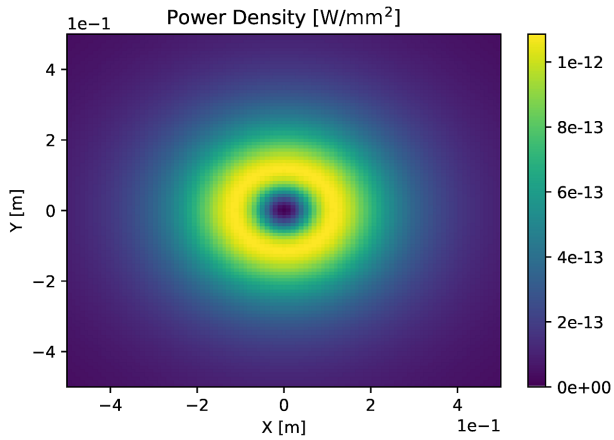

Figure 7: Power density distribution from the motion described in Fig. 6.

GPU, MULTI-THREADING, AND GRID COMPUTING

OSCARS was designed as a multi-threaded application which is also capable of using multiple GPUs simultaneously making use of direct GPU-GPU cascading memory transfers. Native multi-threading is available by default for all calculations on all platforms. The GPU routines are currently written using CUDA [5] for compatible NVidia GPUs and are also available for all calculations. For both multi-threaded and GPU calculations one typically has a set of points in 1D (like a spectrum), 2D (for instance flux maps or power density distributions), or 3D (arbitrary shapes and surfaces in 3D). Given a trajectory, these points are then distributed to the specified number of threads in the case of multi-threading, or sent to the GPU in large thread-blocks until complete. When trajectory interpolations are done to reach the desired level of calculation accuracy they are done using mutex locking when CPU calculations are enabled to avoid thread memory collisions, but are done on-the-fly by each GPU thread as needed.

On a modern GPU the time for a calculation similar to the single-particle flux in Fig. 2 for a 300 × 300 grid is significantly reduced going from 1 core of an Intel® Xeon® CPU

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAG21

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

E5-2630 v3 @ 2.40GHz to using the GPU option in OSCARS on a modest 1344 core Quadro® K4200 GPU (greater than a factor of 15). On a modern high performance GPU one should easily achieve over a factor of 100 in performance improvement.

For extremely large-scale simulations OSCARS was designed to be run on modern grid/cloud computing infrastructures such as the Open Science Grid (OSG) [6]. The multi-particle spectra and flux shown in Fig. 1 and Fig. 2 respectively were performed on OSG and contain 100,000 particles sampled from the beam distributions. Tools are also available for the commonly used Message-Passing Interface (MPI) [7] which may be beneficial when one has a cluster which utilizes MPI, however on a typical workstation the multi-threaded utility will outperform MPI since there is very little overhead in the straightforward implementation used as compared to MPI.

CONCLUSION

A new and modern simulation code for advanced radiation simulation has been developed which is fast, powerful, flexible, and open source. Notably, this new simulation is capable of calculating power densities on arbitrary geometries in 3D and utilizes native multi-threading and GPU computing infrastructure for significantly increased performance for large-scale simulations.

ACKNOWLEDGMENTS

This work has been supported by the U.S. Department of Energy, under contract DE-SC0012704.

REFERENCES

[1] https://oscars.bnl.gov

[2] J. D. Jackson, "Classical Electrodynamics," Berkeley, CA, USA, 1981,

[3] A. Hoffman, "The Physics of Synchrotron Radiation," Cambridge, UK, 2004,

[4] H. Onuki and P. Elleaume, "Undulators, Wigglers and Their Applications," London, UK, 2003

[5] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable Parallel Programming with CUDA," *ACM Queue*, vol. 6, no. 2, pp. 39–53, March/April 2008.

[6] R. Pordes *et al.*, "The Open Science Grid," *J. Phys. Conf. Ser.*, vol. 78, pp. 012057, 2007. doi:10.1088/1742-6596/78/1/012057

[7] L. Dalcín, R. Paz, and M. Storti, "MPI for Python," *Journal of Parallel and Distributed Computing*, vol. 65, no. 9, 2005.