

COMPUTATION OF SYNCHROTRON RADIATION ON ARBITRARY GEOMETRIES IN 3D WITH MODERN GPU, MULTI-CORE, AND GRID COMPUTING

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

Abstract

Open Source Code for Advanced Radiation Simulation (OSCARS) is an open source project developed at Brookhaven National Laboratory for the computation of synchrotron radiation from arbitrary particle beams in arbitrary magnetic (and electric) fields on arbitrary geometries in 3D. OSCARS was designed with considerations for modern and large scale computing infrastructure. These include the ability to use graphical processing units (GPUs) for computations, multi-threaded computations, and utilities for grid (or cloud) computing. Primary applications include, but are not limited to, the computation of spectra, photon flux densities, and notably, power density distributions on arbitrary geometries in 3D which is of interest in accelerator component study and design. This modern approach and several complex geometries are highlighted and elaborated on.

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 and is designed for the most modern high performance computing infrastructures.

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+. No additional packages are required to run the core of OSCARS. The user is free to choose any visualization platform, though visualization tools do exist for OSCARS based on matplotlib [2].

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 $\epsilon_{h,v} = [0.9, 0.008]$ nm rad, and beta function in the center $\beta_{h,v} = [1.5, 0.8]$ m.

PARTICLE BEAMS

Particle beams in OSCARS are defined by particle mass, energy, current, and their position and direction. Optionally one can include the emittance ($\epsilon_{h,v}$), beta function ($\beta_{h,v}$), 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 pre-defined 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, including uniform fields in a given range, Gaussian fields, sinusoidal undulator (wiggler) fields with terminating fields, user input python function for a field, 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 equation 1 using a fourth-order Runge-Kutta algorithm.

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

The electric field in the frequency domain is given in equation 2 which is derived by taking a Fourier transform of the Liéard-Wiechert potentials and given in many texts, for example [3–5]. Equation 2 is valid for relativistic and non-relativistic particles as well as in the near-field.

* dhidas@bnl.gov

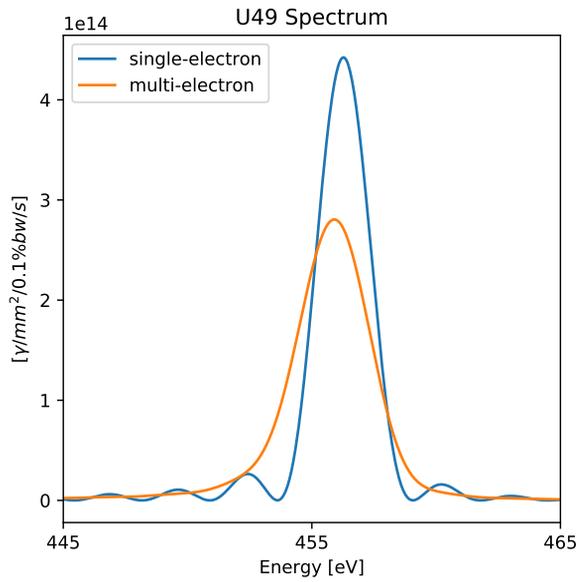


Figure 1: On-axis spectrum for the simulated U49 undulator 30 [m] downstream showing the 3rd harmonic for a filament beam (single-electron), and realistic beam using the NLSL-II design parameters for a 6.6 [m] straight section (multi-electron).

$$\vec{E}(\vec{x}, \omega) = \frac{ie\omega}{4\pi c\epsilon_0} \int_{-\infty}^{+\infty} \frac{\vec{\beta} - \hat{n}(1 + \frac{ic}{\omega D})}{D} e^{i\omega(\tau + \frac{D}{c})} d\tau \quad (2)$$

Here, \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 equation 2 once the trajectory has been calculated using equation 1. This is done either on the Central Processing Unit (CPU) or GPU depending on which mode is selected. The spectrum in figure 1 and flux densities in figure 2 use this method.

POWER DENSITY

Once the trajectory is calculated the power density can be calculated from equation 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.

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

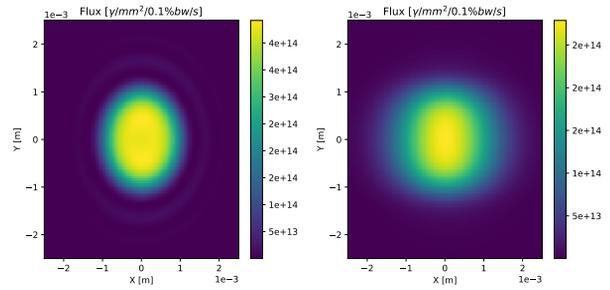


Figure 2: Flux for the simulated U49 undulator 30 [m] downstream showing the 3rd harmonic for a filament beam (left) and realistic beam (right) using the NLSL-II design parameters for a 6.6 [m] straight section.

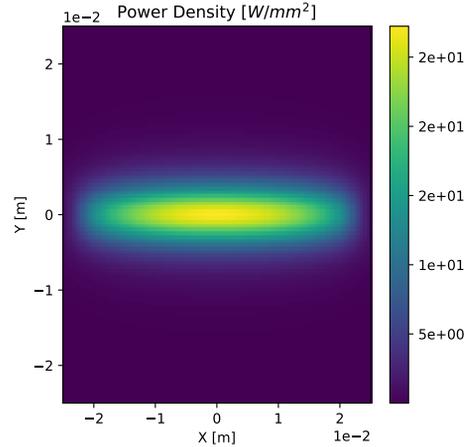


Figure 3: Power density from U49 as seen 30 [m] downstream from the source.

The power density distribution in the X-Y plane 30 [m] downstream from the example U49 is shown in figure 3. OSCARS allows for the calculation of power density distributions on arbitrary objects in 3D. Several examples of this are shown in figure 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.

GPU, MULTI-THREADING, AND GRID COMPUTING

OSCARS was designed as a multi-threaded application which is also capable of using the GPU. Native multi-threading is available by default for all calculations on all platforms. The GPU routines are currently written using CUDA [6] 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.

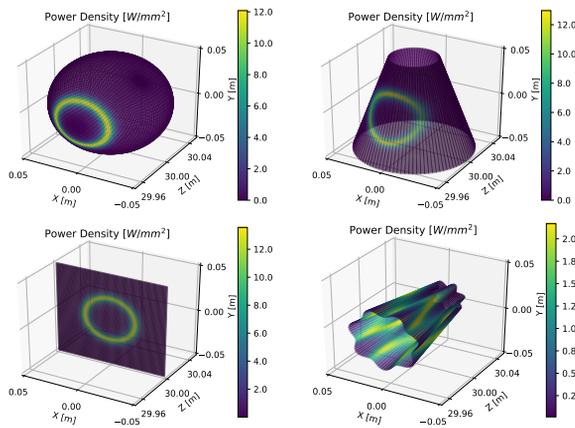


Figure 4: Power density on various 3D shapes for the simulated EPU49 undulator 30 [m] downstream.

On a modern GPU the time for a calculation similar to the single-particle flux in figure 2 for a 300x300 grid is significantly reduced going from 1 core of an Intel® Xeon® CPU 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) [7]. The multi-particle spectra and flux shown in figures 1 and 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) [8] 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.

BRIGHTNESS

OSCARS now (as of version 1.36.00) has a module (.th) which consists of specific analytic formulae for undulator and bending magnets. This includes a calculation of undulator brightness an example of which is shown for several odd harmonics of of the U49 undulator (maximum deflection parameter $K_{max} = 4.58$) in 6.6 [m] straight section of NSLSII is shown in figure 5.

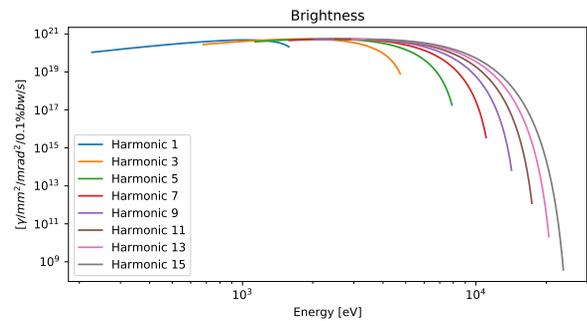


Figure 5: Brightness curves for the simulated U49 showing the first 8 odd harmonics.

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

- [1] <http://oscars.bnl.gov>
- [2] J. D. Hunter, “Matplotlib: A 2D graphics environment,” in Computing In Science and Engineering vol 9 n 3, 2007.
- [3] J. D. Jackson, “Classical Electrodynamics,” Berkeley, CA, USA, 1981.
- [4] A. Hoffman, “The Physics of Synchrotron Radiation,” Cambridge, UK, 2004.
- [5] H. Onuki, P. Elleaume, “Undulators, Wigglers and Their Applications,” London, UK, 2003.
- [6] J. Nickolls, I. Buck, M. Garland, K. Skadron, “Scalable Parallel Programming with CUDA,” ACM Queue, vol. 6 no. 2, March/April 2008, pp. 39-53.
- [7] R. Pordes et al., “The Open Science Grid,” in J. Phys. Conf. Ser. 78, 012057.doi:10.1088/1742-6596/78/1/012057, 2007. <https://www.opensciencegrid.org>
- [8] L. Dalcín, R. Paz, M. Storti, “MPI for Python,” in Journal of Parallel and Distributed Computing, vol 65 n 9, 2005.