COMPUTATION OF SYNCHROTRON RADIATION

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

Abstract

This presentation introduces a new open-source software development for the computation of radiation from charged particles and beams in magnetic and electric fields. The computations are valid in the near-field regime for both relativistic and non-relativistic scenarios. This project is being developed, and is currently in use, at Brookhaven National Laboratory's National Synchrotron Light Source II. Primary applications include, but are not limited to, the computation of spectra, photon flux densities, and power density distributions from undulators, wigglers, and bending magnets on arbitrary shaped surfaces in 3D making possible detailed study of sensitive accelerator and beam-line equipment. Application interfaces are available in Python, Mathematica, and C. Practical use cases are demonstrated and benchmarked. Additionally, future upgrades will be elaborated on.

INTRODUCTION

Open Source Code for Advanced Radiation Simulation (OSCARS) [1] is a new open source software developed at Brookhaven National Laboratory (BNL). OSCARS is a general purpose code for the calculation of radiation of 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 great interest. This software allows for the calculation of these properties on arbitrary shaped surfaces in 3 dimensions.

At its core OSCARS is a numerical discretization of derived equations from the Liéard-Wiechert potentials, which are valid for relativistic and non-relativistic particles alike and includes the so-called near-field contributions. These calculations are based off of the particle trajectory, which in OSCARS is calculated from the well known relativistic Lorenz force given in equation 1. The particle trajectory propagation is determined in 3D using a 4^{th}-order Runge-Kutta method to solve the second order equations of motion. Other methods, such as adaptive step, are easily implemented and of potential future interest depending on use cases.

$$\frac{\mathrm{d}\vec{p}}{\mathrm{d}\tau} = q(\vec{E} + \vec{v} \times \vec{B}) \tag{1}$$

CORE CAPABILITIES

OSCARS is capable of computing charged particle trajectories, power density distributions, flux density distributions, and spectra for charged particles and charged particle beams in arbitrary magnetic field configurations. This includes multi-particle simulations of mixed-particle type beams of native or user defined particle types. OSCARS is capable of computing power density and flux density distributions on arbitrary surfaces in 3 dimensions. OSCARS also allows for the easy definition of surface planes and arbitrary objects in 3D, with simple utilities to translate and rotate them in space, making it versatile and extendible to many applications at accelerator facilities.

Utilities are built-in for modern large scale computing. A typical user will benefit from an easy python MPI [2] implementation to achieve moderate gains in speed on their desktop, workstation computer, or cluster. For **very** large-scale simulations OSCARS is also designed to be run on any modern grid/cloud computing infrastructure such as the Department of Energy and National Science Foundation supported Open Science Grid [3], where it has already been run extensively.

OSCARS is capable of reading in 1D, 2D, and 3D field configurations of various formats. In the case of 1D data, irregularly spaced points are interpolated to form a uniform grid to maximize the speed of trajectory calculations. Additionally OSCARS has utilities built-in for generating dipole, Gaussian, undulator, and wiggler fields. These fields are not discretized which allows for high precision calculations. A user may also input any functional form in the same manner by simply creating any python function which describes the field of interest.

The core code is written in C++ for speed and has a python extension. The main application programming interface is written in Python for ease of use and integrability by the larger scientific community. Currently the extension is available for python 2.7 and is forthcoming for python versions $\geq$ 3.4. No additional packages are required to run the core of OSCARS. The user is free to choose any visualization platform. As of writing this a modest package for visualization exists for OSCARS based on matplotlib [4] which is non-essential, but perhaps convenient.

SPECTRUM & FLUX DENSITY

The electric field is calculated in the frequency domain directly from equation 2, which is derived by taking the Fourier transform of the Liéard-Wiechert potentials. This derivation and different formulations can be found in many texts [5–7]. The spectrum is calculated for the idealized planar undulator U49 (λ_m = 49 [mm], $N_{periods}$ = 55, and B_v^{max} = 1 [T]) for both the case of a filament beam (single-particle) and for the NSLS-II design parameters in a 6.6 [m] straight section at its final operating current of 500 [mA]. The beam parameters used are ϵ_v = 0.008 [nm rad], ϵ_h = 0.9 [nm rad], β_v = 0.8 [m], β_h = 1.5 [m], E = 3 [GeV], and $\Delta E/E$ = 0.001. Figure 1 shows the on-axis spectrum and flux density at 30 [m] around the 3^{rd} harmonic in the spectrum. The computation of the flux density on a grid of 200×200 takes less than 4 seconds on a modest laptop (with 2.8 GHz Intel Core i7).

* dhidas@bnl.gov

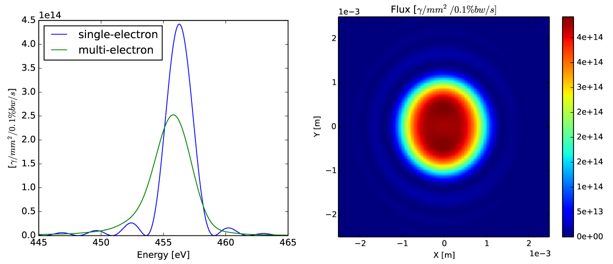

Figure 1: *Left:* On-axis spectrum for the simulated U49 undulator 30 [m] downstream showing the 3rd harmonic for a filament beam (single-particle), and realistic beam using the NSLS-II design parameters for a 6.6 [m] straight section (multi-particle). *Right:* Flux density from the same U49 as seen for an energy of 456 [eV] at 30 [m] downstream from the source.

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

POWER DENSITY

The power density is calculated as the radiated power given by equation 3. OSCARS allows for the calculation of the power density on arbitrary surfaces by simply weighting the integration by the scalar product $\hat{n}\cdot\hat{S}$ at each point in the discretization, where $\hat{S}$ is the surface unit normal. This is notably different than calculating this quantity on a flat surface and projecting it onto the desired surface (as can be done in popular CAD software), which in some cases produces an incorrect result.

Figure 2 shows the power density from a simulated elliptically polarizing undulator EPU49 (similar to U49 with an additional B_h component offset in phase by $\pi/2$ from B_v with the same magnitude). The first plot shows the power density distribution on a plane perpendicular to the direction of the beam while the second one shows the power density on a plane parallel to the beam direction but offset vertically by 4 [mm] and centered longitudinally at the center of EPU49 (which is also taken to be the lattice reference point). The latter, one can imagine, is the interior surface of the flat portion of the vacuum chamber passing through the center of an undulator.

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

GEOMETRIES AND FEATURES

OSCARS is capable of calculating these quantities on **any** arbitrary surface. One example of this is shown in Figure 3. Included in OSCARS are utilities for parametric 3D surfaces. One is not limited to these types of surfaces, however visualization of parametric surfaces is common for many data

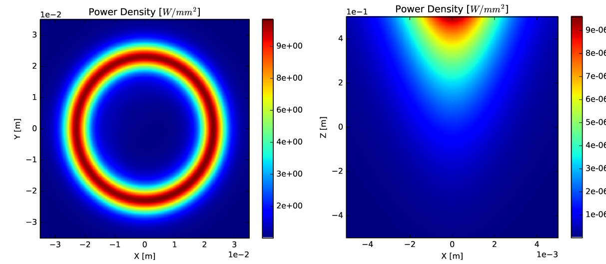

Figure 2: Power density distribution from the simulate EPU49 on a surface perpendicular to the beam (*left*) 30 [m] downstream and on a surface parallel to the beam direction displaced from the beam axis by 4 [mm] (*right*).

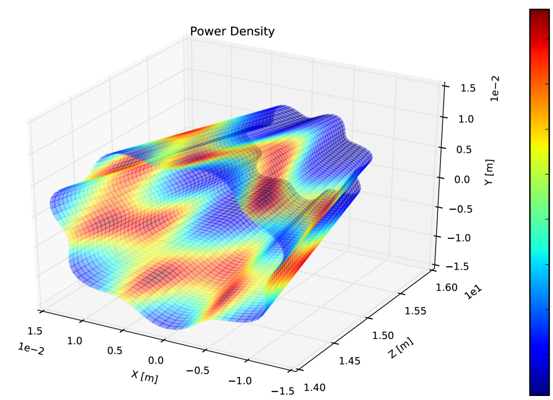

Figure 3: Power density distribution from a simulated EPU on a tapered and corrugated fictitious beam-pipe 20 [m] downstream.

visualization frameworks. The requirement for a surface is only a list of points in space and their corresponding surface normal vectors. In future releases calculation of the normal vector will likely be automated for differential geometries.

OSCARS is a generalized platform for radiation computation. There is no preferred direction for any element, giving ultimate control to the user. For example, particle beams can be defined in any direction and fields can easily be imported, rotated, and translated in space. Simple tools exist to create rectangular surfaces and orient them in any direction and position in space. All elements in OSCARS can be easily translated and rotated in space.

Any computation in OSCARS can be run in single-particle or multi-particle mode by simply adding the argument *nparticles=1000* (for 1000 particles). When multiple beams are used they are sampled according to their defined weight. The phase space for that beam is then sampled for the individual particle kinematics.

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.

ACKNOWLEDGMENTS

This work has been supported by the U.S. Department of Energy, under contract DE- AC02–98CH10886

REFERENCES

[1] `http://oscars.bnl.gov`

[2] L. Dalcín, R. Paz, and M. Storti, "MPI for Python," in *Journal of Parallel and Distributed Computing*, vol 65 n 9, 2005.

[3] 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`

[4] J. D. Hunter, "Matplotlib: A 2D graphics environment," in *Computing In Science and Engineering* vol 9 n 3, 2007.

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

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

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