Jupyter for Accelerator Physics

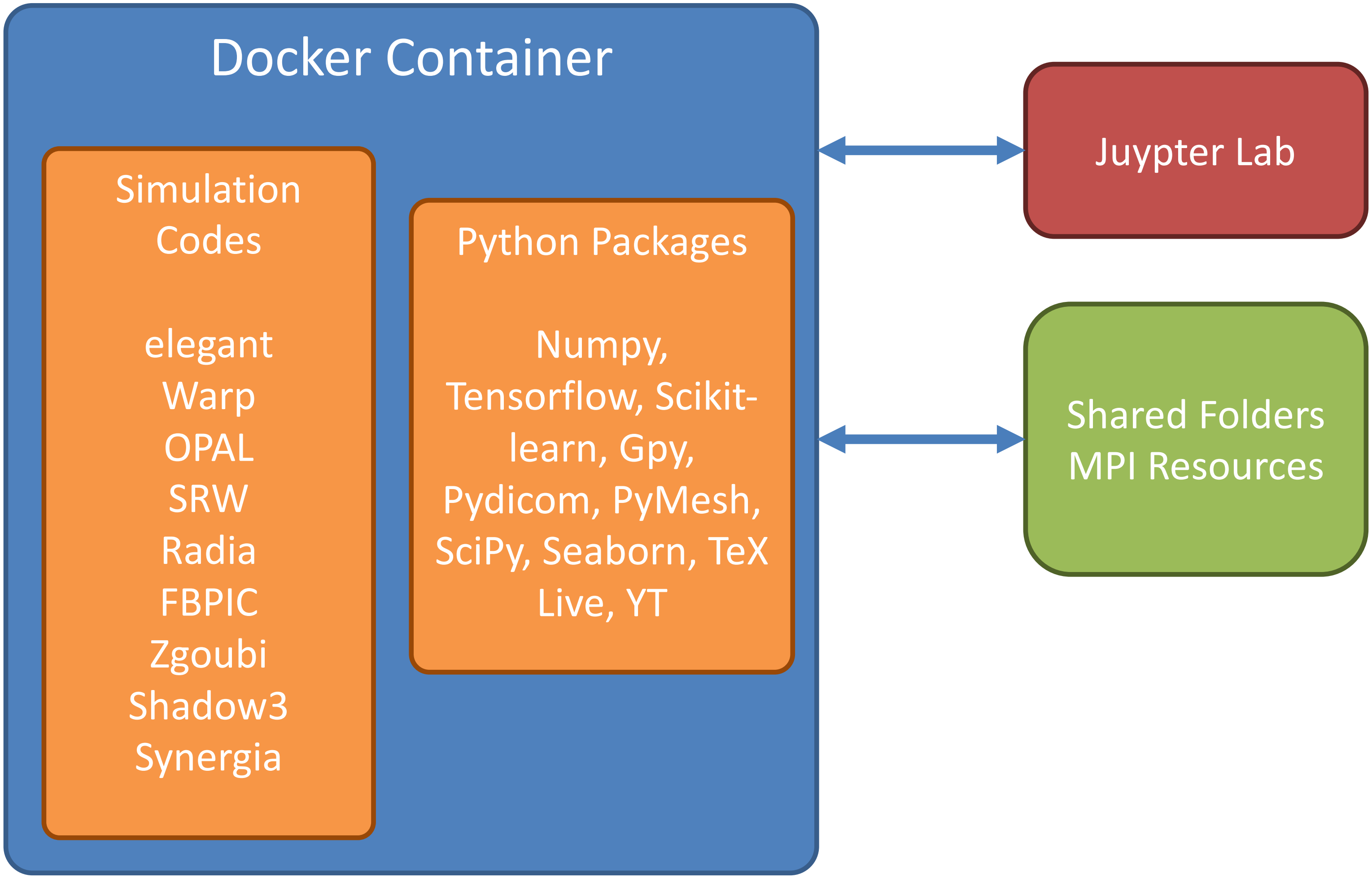
Jonathan Edelen, Robert Nagler, Paul Moeller, David Bruhwiler, Nathan Cook, Chris Hall, and Stephen Webb

radiasoft

Jupyter Tutorial @ ICALEPCS 2019

5 October 2019 – Brooklyn, NY

Jupyter/Hub @ RadiaSoft



RadiaSoft Jupyter/Hub Enviornment

- *17 staff users and 60 public users (in last 4 weeks)*
- *8TB used*
- *Pools: 3 public nodes, 5 internal nodes*
- *MPI: 12 nodes (pool nodes for workshops)*
- *Nginx proxy*
- *Dev, Alpha, Beta, Prod configurations*

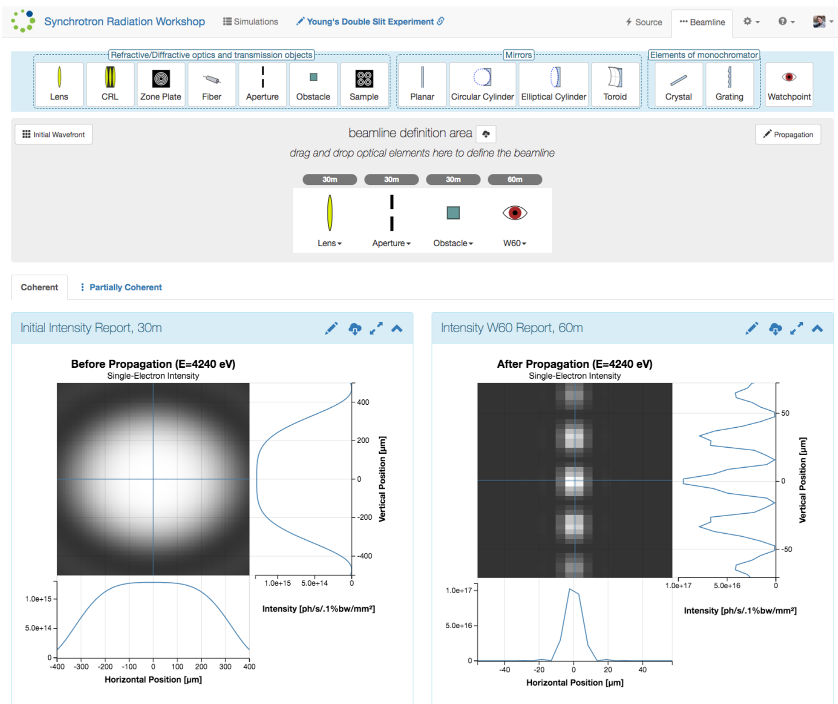
User Customizations

- *`github.com/radiasoft/jupyter.radiasoft.org`*
  - *Executes `radia-run.sh` inside container before Jupyter starts*
  - *Copies template notebooks and other files*
  - *Used for patches in between releases*
  - *Runs git config user.name and credential.helper*
  - *If user has jupyter.radiasoft.org repo, it runs after global repo*
- *`~/jupyter/bashrc` runs after container's bashrc*
- *`~/jupyter/bin` in path lets users persist commands*

Jupyter/Hub Use Case: Teaching and Workshop Tutorials

- *Fermilab scientist learned Synergia via example notebooks running on jupyter.radiasoft.org*
- *UCLA undergrad learned FBPIC via example notebook in order to complete work study*
- *Grad student at UCLA learned Warp through example RadiaSoft notebooks*
- *Jan 2018 session of US Particle Accelerator School used jupyter.radiasoft.org to teach Synergia to 20 students*
- *ICFA ML Workshop in CH used jupyter.radiasoft.org to teach ML for accelerator physics to 60 participants*

Jupyter/Hub Use Case:
Comparing Multiple Simulation Codes



README.md

SHADOW 3.0 SOURCE DISTRIBUTION

Contents:

1. What is SHADOW
2. Download
3. Source files
4. Building SHADOW
5. Other info
6. Contact

1 What is SHADOW

SHADOW is an open source ray tracing code for modeling optical systems.

Targeted to synchrotron radiation beamlines, it has unique features for designing X-ray optical systems.

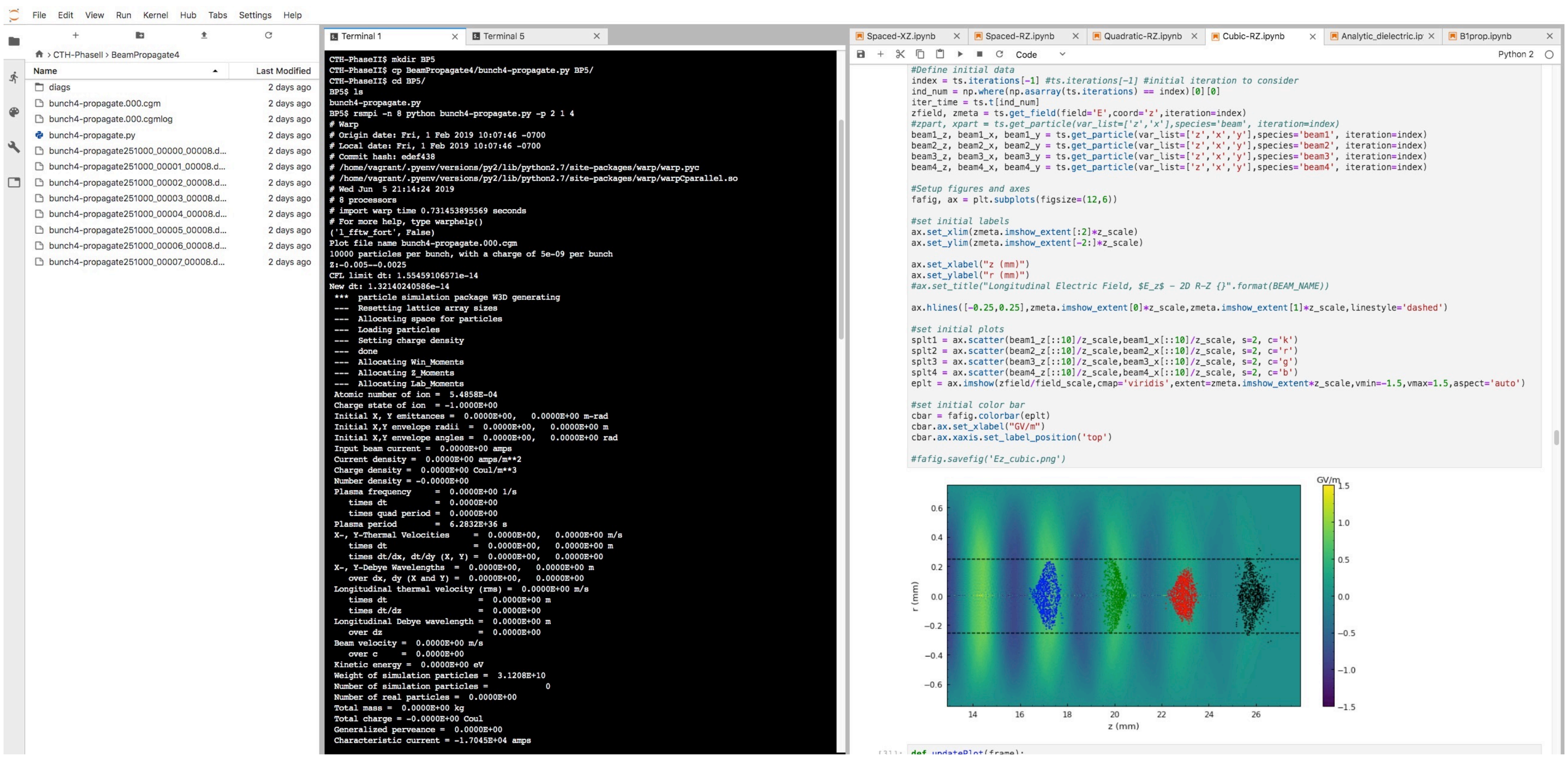
For more info, please read this paper (open access):

SHADOW3: a new version of the synchrotron X-ray optics modelling package M. Sanchez del Rio, N. Canestrari, F. Jiang and F. Cerrina Journal of Synchrotron Radiation Volume 18, Part 5 (September 2011) http://dx.doi.org/10.1107/S0909049511026306

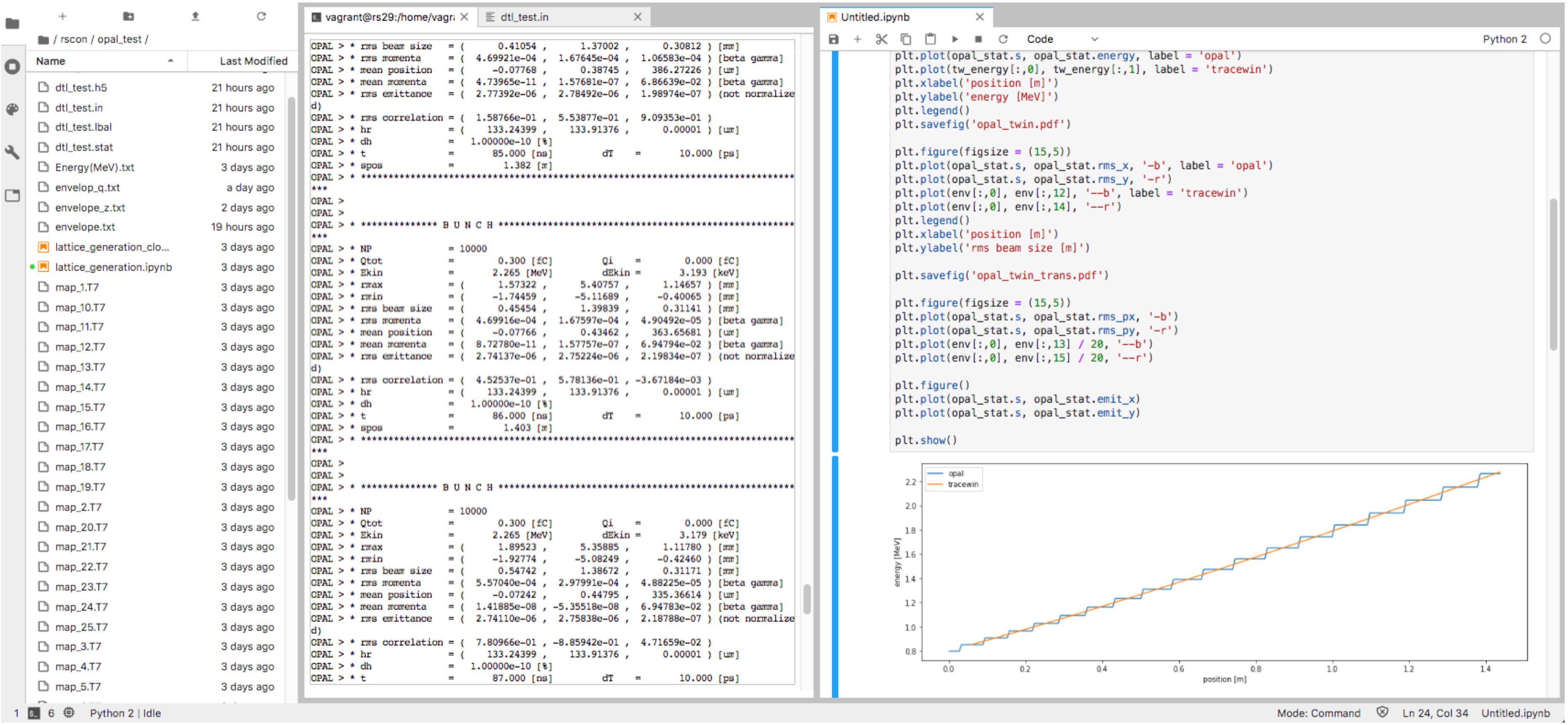
If you are (or want to be) a SHADOW user, it is recommended that you use a user interface. We strongly recommend OASYS (https://www.elettra.eu/oasys.html).

Jupyter/Hub Use Case: In-situ Analysis

"This is my most common working arrangement, as I am consistently running simulations in one panel while running analysis in a notebook in another."



Jupyter/Hub Use Case: In-situ Analysis



Jupyter/Hub Use Case:
Machine Learning for Accelerators

Sirepo Machine Learning Tutorial: Predicting the e- beam Longitudinal Phase Space from Elegant Simulations of the LCLS

NAPAC'19 Sirepo Users Workshop (September 3, 2019)

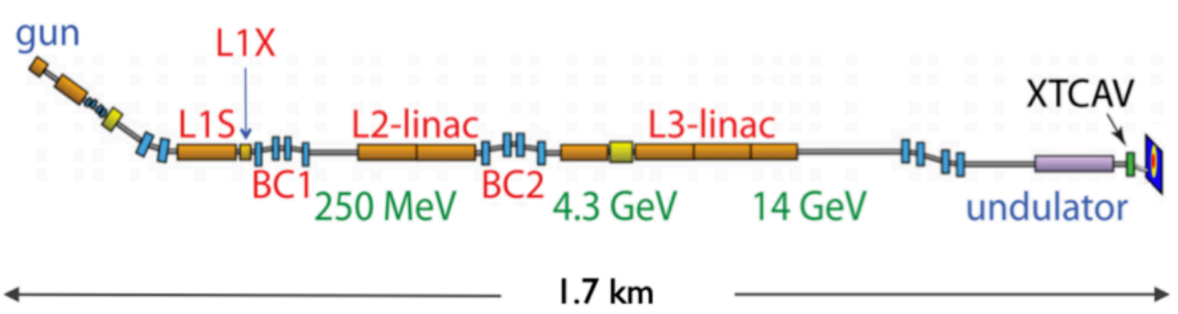
Developed by Auralee Edelen (SLAC) and Christopher Hall (RadiaSoft)
Elegant simulation of LCLS adapted from work by Yuantao Ding (SLAC)

Introduction

The aim of this tutorial is to show one example of how machine learning can be used to predict beam parameter output in an accelerator system.

Although we give some very basic tips along the way, this is not meant to be a pedagogical introduction to machine learning (for that and associate hands-on notebooks with toy accelerator problems see the ICFA ML tutorials: https://indico.psi.ch/event/6698/sessions/3632/#20190226).

Here, we run an elegant simulation of the LCLS with phase and amplitude jitter in L1 and produce the resultant e- beam phase space at the end of BC1 or BC2.



Jupyter/Hub Use Case:
Machine Learning for Accelerators

Create Training Data - Jitter in Linac Phases and Amplitudes

(or skip if you want to just load the pre-made output file)

The training data simulates the impact of jitter in the linac phases and amplitudes. The data is generated by imposing random errors, with a Gaussian distribution, on phase and amplitude in the L1 linac section. The first run is a fiducializing step that uses the exact linac settings.

- `data_points` : Number of times to run.
- `phase_error` : The width of the Gaussian distribution of error in the L1 phase settings. This is an absolute error in units of Degrees.
- `amplitude_error` : The wdith of the Guassian distribution of error in the L1 amplitude (Voltage) setting. This is a fractional error based on the default Voltage setting.

You can choose some range of setting errors and number of points to sample, but use the nominal values first.

More data is not always better -- we need to avoid oversampling the space, which would make our training set look identical to our validation and testing sets.

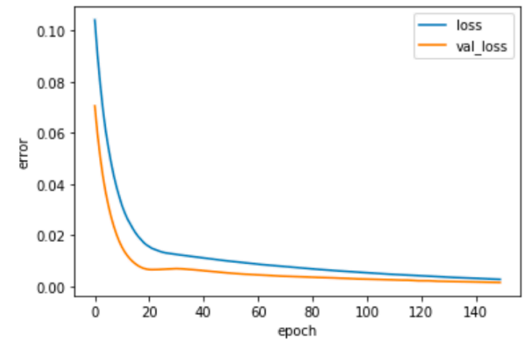
]:

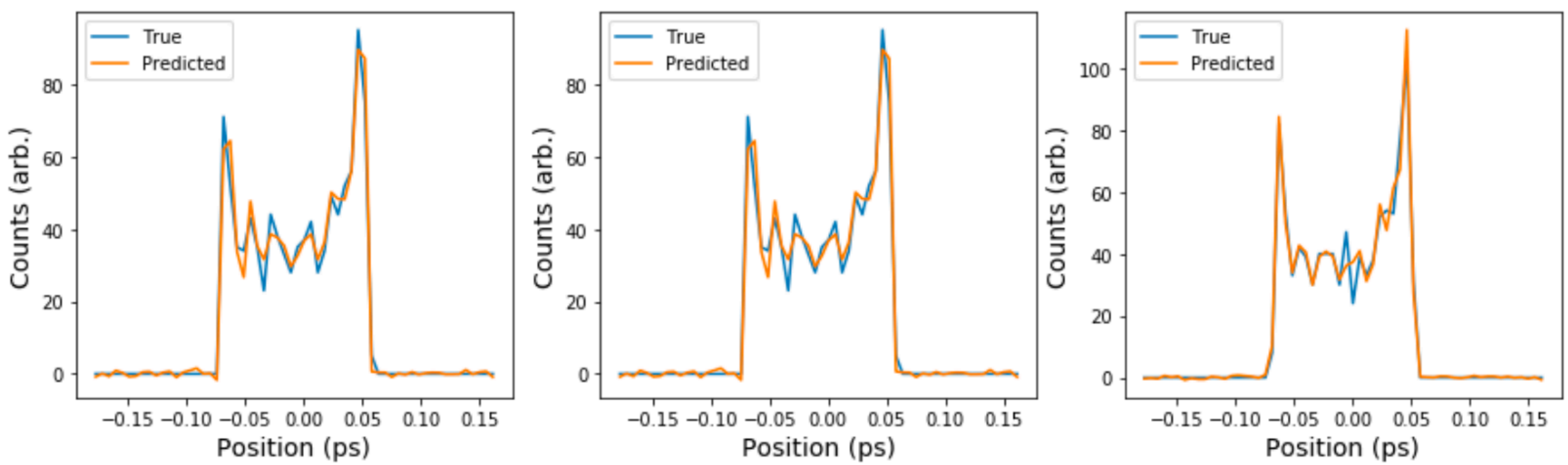
```
make_data = True

if make_data == True:
    # Scan settings
    data_points = 1000              # nominal: 1000
    phase_error_setting = 0.03      # nominal: 0.03
    amplitude_error_setting = 0.02  # nominal: 0.02
    lattice_name = 'BC2LINE'        # nominal: BC2LINE, may be BC1LINE or BC2LINE

    # Run scan in elegant
    t1=time.time()
    error_scan(max_iter=data_points,
               phase_error=phase_error_setting,
               amplitude_error=amplitude_error_setting,
               lattice=lattice_name)
    print(str('simulations took ' + str(np.round(((time.time()-t1)/60),2))+ ' mins'))
```

Jupyter/Hub Use Case: Machine Learning for Accelerators

```
[1]: #import required packages
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.constants import physical_constants
from SDDS import readSDDS
m_e = physical_constants["electron mass energy equivalent in MeV"][0]
from utilities3 import error_scan, lps_plot
from utilities import read_settings
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler,MinMaxScaler
import keras
from keras.models import Sequential
from keras.layers import Dense, GaussianNoise
from mlhelpers4 import PlotLosses, plot_a_bunch_of_beams,make_dataset
plot_losses=PlotLosses()
import time
```

```
# fit model
hist = model.fit(x = x_train, y = y_train,
                 validation_data = (x_val,y_val),
                 batch_size = 100,
                 shuffle = 'true',
                 epochs = 150,
                 verbose = 'false',
                 callbacks=[plot_losses])
```





Jupyter/Hub Use Case:
Simulating Dielectric Wakefield Accelerators

Slab Dielectric Wakefield Accelerator Demonstration

Nathan Cook & Stephen Webb
RadiaSoft Sirepo User Workshop
swebb@radiasoft.net

This notebook documents the analysis of the GPT phase space provided by Gwanghui Ha. Afterwards, the notebook will be expanded to set up a Warp simulation using that phase space to describe each drive bunch.

This version removes the dielectric and looks at the bunch evolution from the center of the channel. Unique species labels for each bunch are included (e.g. beam1/2/3/4).

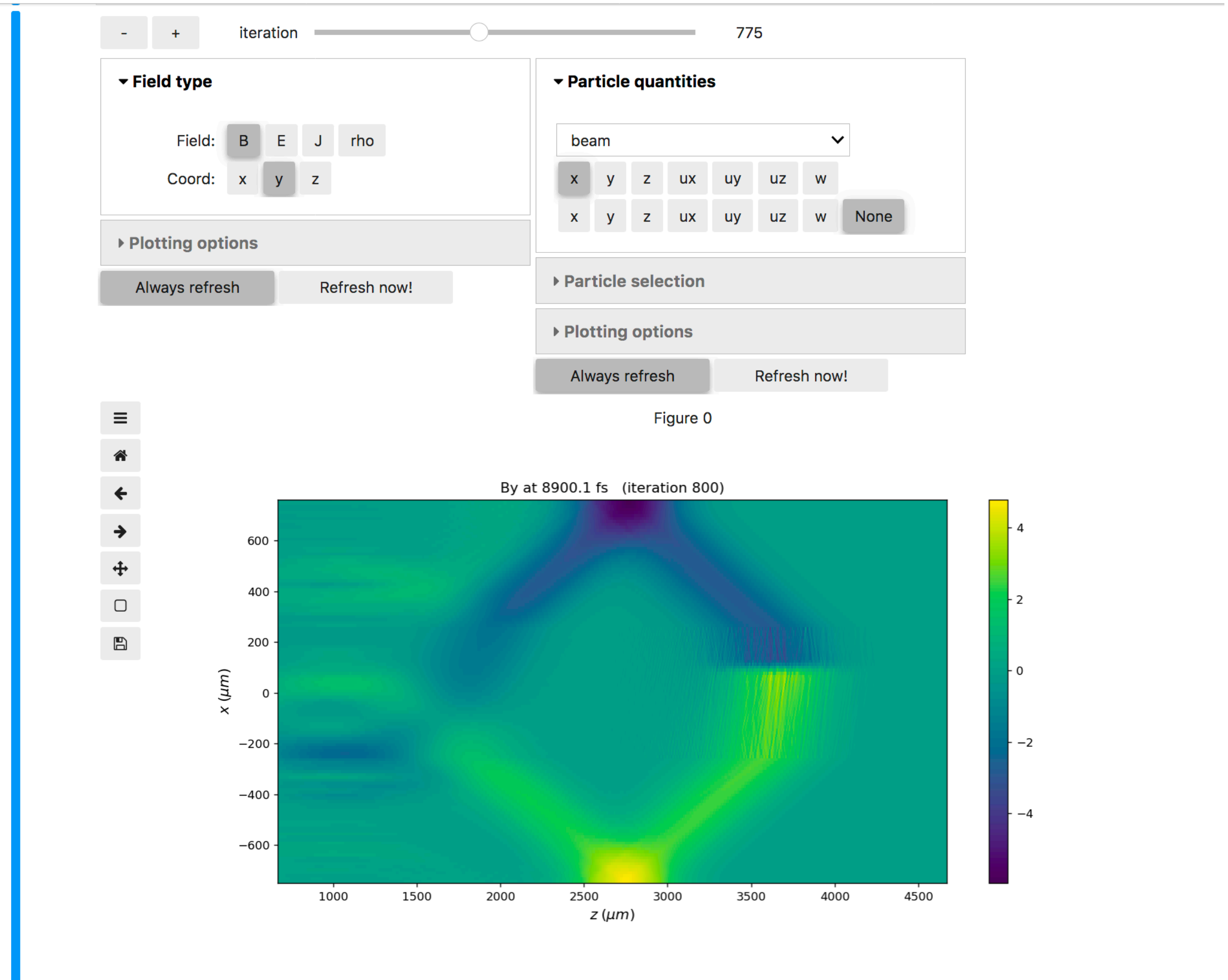
This version takes the X-Z geometry with original bunch spacing and changes the deposition order to third order.

Jupyter/Hub Use Case: Simulating Dielectric Wakefield Accelerators

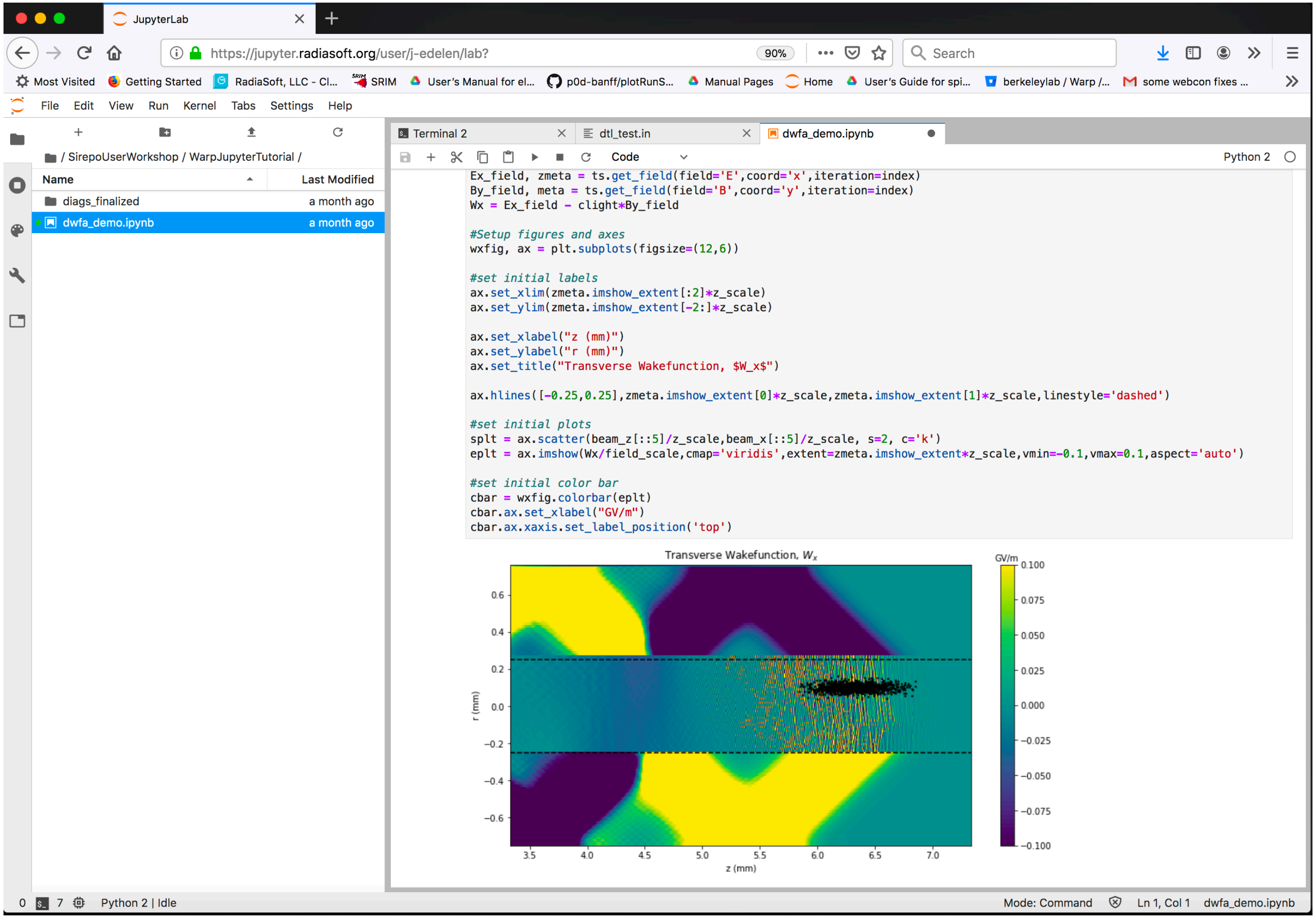
```
%matplotlib inline

# Basic imports
import sys
del sys.argv[1:] #  Necessry to run 'from warp import *' in IPython notebook without conflict.

# Import warp-specific packages
from warp import *
from warp.init_tools import *
from opmd_viewer import OpenPMDTimeSeries


# Import rswarp packages
import rswarp
from rswarp.utilities.file_utils import cleanupPrevious
from rswarp.utilities.file_utils import readparticles
from rswarp.utilities.file_utils import loadparticlefiles

# Import plotting and analysis packages
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

# Constants imports
from scipy.constants import e, m_e, c, k, elementary_charge
from scipy.constants import c as clight
kb_eV = 8.6173324e-5 #Bolztmann constant in eV/K
kb_J = k #Boltzmann constant in J/K
m = m_e # electron mass
qe = elementary_charge # electron charge
```

```
# Warp
# Origin date: Fri, 1 Feb 2019 10:07:46 -0700
# Local date: Fri, 1 Feb 2019 10:07:46 -0700
# Commit hash: edef438
# /home/vagrant/.pyenv/versions/py2/lib/python2.7/site-packages/warp/warp.py
# /home/vagrant/.pyenv/versions/py2/lib/python2.7/site-packages/warp/warpC.so
# Fri Oct  4 19:18:33 2019
# import warp time 0.634296178818 seconds
# For more help, type warphelp()
('l_fftw_fort', False)
```

Jupyter/Hub Use Case: Simulating Dielectric Wakefield Accelerators



Jupyter/Hub Use Case: Simulating Dielectric Wakefield Accelerators



Takeaways

- *Wish list*
  - *Storage limits (quotas)*
  - *User/group file sharing*
  - *Real-time collaboration/debugging (like CoCalc)*
  - *Better user notifications (server restarts, long operations, no more servers)*
  - *Hub admin page spawner-specific output*

- *Highlights*
  - *Users love Jupyter*
  - *Users want all codes pre-installed*
  - *Users will consume all available resources*
  - *Jupyter/Hub is easily customizable*