

Modern Computational Accelerator Physics

James Amundson Alexandru Macridin *Panagiotis Spentzouris*

Fermilab

USPAS January 2015

Outline of the class

- Overview and comments on computing
- Simulating independent particles
- Introduction to Synergia
- Simulating collective effects

Overview

Computation accelerator is a huge topic, crossing several disciplines. The three main areas of current interest are

- Electromagnetic simulations of accelerating structures
- Simulations of advanced accelerator techniques, primarily involving plasmas
- Beam dynamics simulations

We will limit ourselves to the last topic. Even so, we should consider

- Accelerator physics
- Nonlinear dynamics
- Numerical analysis in general
- Numerical techniques for ordinary differential equations (ODEs)
- Numerical techniques for partial differential equations (PDEs)
- Symplectic integration
- Parallel computing

The focus of this class

We only have one week.

- We will emphasize topics that are relevant for current research.
- We will not spend much time on topics that can easily be learned from textbooks.
- We will take a hands-on approach.
 - More time doing than listening.
 - We will give you the tools to explore various topics and some basic assignments. I hope you will take advantage of the opportunity to explore.

By the end of this class, you should

- Know something.
- Have a better idea of the existence of the many things you do not know.

You will be given a number of exercises in class.

- Most will be recorded as completed
- A few designated more open-ended assignments will be given a numerical score.

Major areas in Beam Dynamics

- Independent particle physics
 - The interaction of individual particles with external fields, *e.g.*, magnets, RF cavities, etc.
 - Usually the dominant effect in an accelerator
 - Otherwise, it wouldn't work...
 - Well-established theory of simulation
 - Easily handled by current desktop computers
- Collective effects
 - Space charge, wake fields, electron cloud, beam-beam interactions, etc.
 - Usually considered a nuisance
 - Topic of current beam dynamics simulation research
 - Calculations typically require massively parallel computing
 - Clusters and supercomputers

Comments on computing: Philosophy, General Principles and Nitty Gritty

First, the philosophical

Computer programs are not equations.

As a physicist, deriving every equation for yourself is a noble ideal. Writing every program for yourself is not the same thing.

We emphasize the use of the current state-of-the-art in numerical libraries. In particular, we will be using the GNU Scientific Library, FFTW, and LAPACK, even if indirectly at times.

A pathological example is *Numerical Recipes: The Art of Scientific Computing*.

Numerical Recipes

Numerical Recipes is often a good place to start. Numerical Recipes is almost always a bad place to stop.

Computing: General Principles

- Code change is inevitable. Therefore, old code must be read.
- *Writing* code is easy; it is *reading* code that is hard.
 - Making code easier to write is missing the point.
- It is always preferable to create a single point of maintenance.
- When stuck on a debugging problem, I find it helpful to start by assuming that it must be a compiler bug.
 - The next thing to do is to file a bug report.
 - A proper bug report must demonstrate the problem in the briefest form possible.
- You will reach a point when it is clear that either your computer is broken or you are crazy.
 - I have bad news for you.
 - Talk therapy helps.
 - Explain it so someone else.
- A book on effective programming I like: *Clean Code: A Handbook of Agile Software Craftsmanship* by Robert C. Martin.

Tips and tricks

Python:

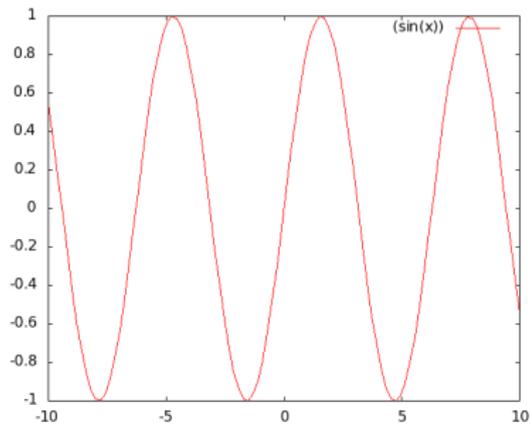
- When a Python program is in a call to a module written in C or C++, ctrl-C will often not stop the program. Use ctrl-Z instead, followed by `kill %n` where n is the number of the stopped job.
- n.b.: in Python 2.x, $9.0/10 = 0.9$ and $9/10.0 = 0.9$, but $9/10 = 0$.

Matplotlib:

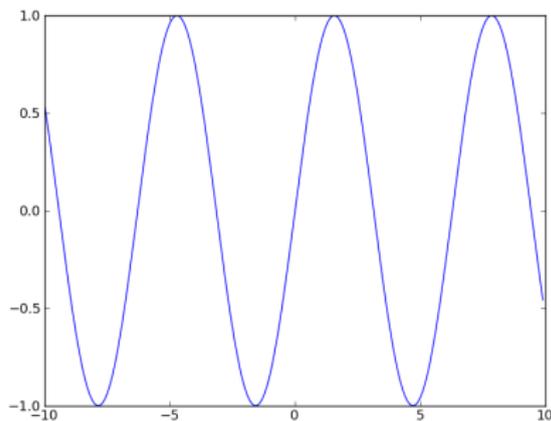
- Look at the gallery to get an idea of what is possible with Matplotlib: <http://matplotlib.sourceforge.net/gallery.html>.
- The floating exponent and offset are a frequent source of confusion.
 - Look at Gnuplot vs. Matplotlib for three examples.

$\sin(x)$

Gnuplot

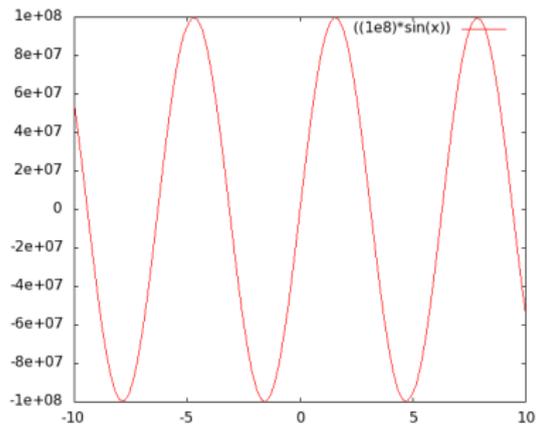


Matplotlib

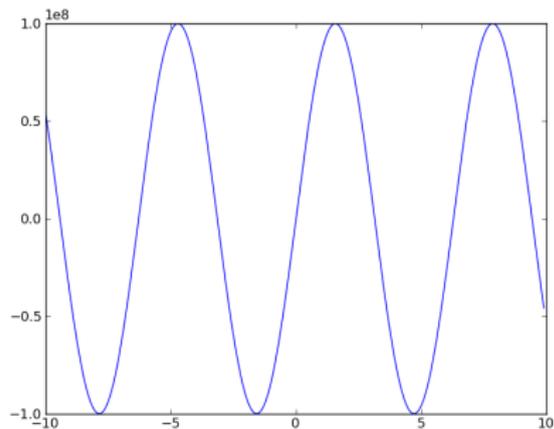


$$10^8 \sin(x)$$

Gnuplot

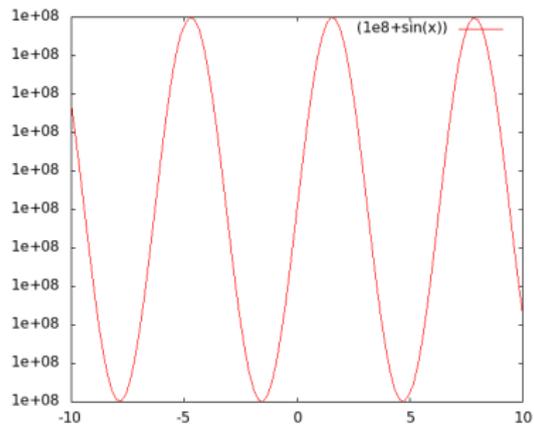


Matplotlib

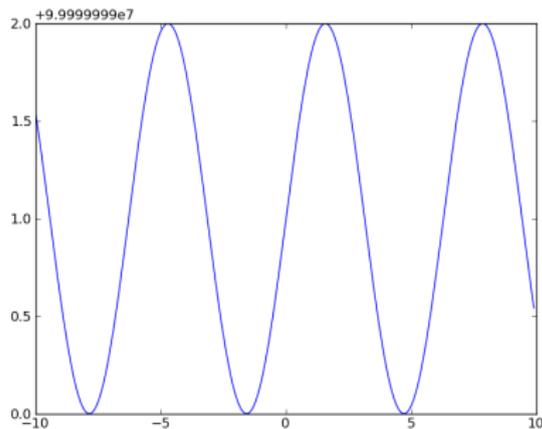


$$10^8 + \sin(x)$$

Gnuplot

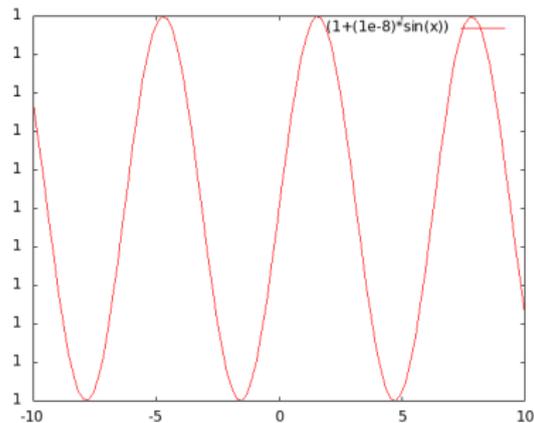


Matplotlib

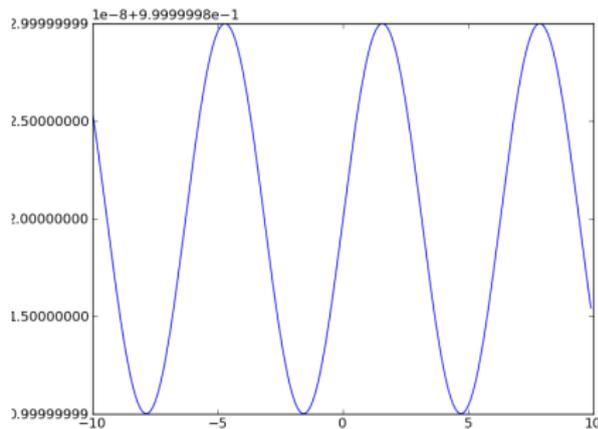


$$1 + 10^{-8} \sin(x)$$

Gnuplot



Matplotlib



Exercise 0

- 1 Install Synergia
- 2 Install pygsl
- 3 Use Matplotlib to plot $\sin(x)$