Introduction to Computational Physics

Aldo H Romero

Department of Physics and Astronomy,
West Virginia University,
Morgantown, WV 26505-6315, USA

August 22, 2020

Abstract

Herewith, we will make a small intro about what is the class about and we will describe what I do expect from your reports. Specially the final report.

Introduction

The main purpuse of this class is to give you a background on computational science applied to physics. Basically, we will do science via numerical experiments. I hope that at the end of this semester, you have a clear way of doing "algorithmic thinking". An idea you have used before but probably you were not aware of.

Computing means that we are able to solve scientific problems using computers. It goes from **symbolic computing**, as you do in your math classes, to **numerical computing**, where we develop numerical methods that the computer can solve efficiently.

Some of the following info is based on the nice **Morten Hjorth-Jensen** class notes.

what we expect then in terms of the our goals are:

- derivation, verification, and implementation of algorithms
- understanding what can go wrong with algorithms
- overview of important, known algorithms
- understanding how algorithms are used to solve complicated problems
- reproducible science and ethics
- algorithmic thinking for gaining deeper insights about scientific problems

As you learn how to deal with everyone of these goals, you will have to make million of mistakes, until you get a better understanding. To provide a path on your thiking, the reports you will be submitting are giving you the chance to put in writing this process.

why do we need these computational methods?

- Theory+experiment+simulation is almost the norm in research and industry
- To be able to model complex systems with no simple answers. Solve real problems.
- Emphasis on insight and understanding of fundamental principles and laws in the Sciences.
- Be able to visualize, present, discuss, interpret and come with a critical analysis of the results, and develop a sound ethical attitude to own and other's work.
- Enhance reasoning about the scientific method

Practicalities

This semester we will use different utilities to program, to check, to plot and to hand it homeworks. Let me go, one by one

- Running Python. We can use
  - Google colab.
  - My own computer (if you have not done so, use anaconda to install IPython)
  - Spruce. This is for "larger computations" and it will not be interactive.
- Homeworks will be accessed through GitHub. Every week you will get a link that refers to that week and you can work the solution and uploaded to your site. I will explain how to do it a bit later.

You **must** have a *github account* where we can monitor your progress and give you appropriate feedback.

Furthermore, when setting up your git repository for a given numerical project, you should create a folder where selected benchmarks are placed. These benchmarks could represent a calculation with specific input parameters. This makes your work reproducible, and allows us to see that your programs reproduce selected benchmarks. Furthermore, developing a habit of producing benchmarks, allows you to keep track of the results produced by different versions of your codes.

If you have not used version control before now, it is thus time to do so. **Proper version control is central to a good ethical scientific conduct**. We do require that you use some kind of version control software when working on the projects. We recommend strongly github. Lectures will be available in this GitHub site:

General advices on successful numerical projects

1. Structure the code in terms of functions
2. Make module
3. How to read input data flexibly from the command line
4. How to create graphical/web user interfaces
5. How to write **unit tests** (test functions)
6. How to refactor code in terms of classes (instead of functions only), in our case you think of a system and a solver class
7. How to conduct and automate large-scale numerical experiments
8. How to write scientific reports in various formats (LaTeX, HTML). Important for large projects.

If the previous page works, then this follows

1. New code is added in a modular fashion to a library (modules)
2. Programs are run through convenient user interfaces
3. It takes one quick command to let all your code undergo heavy testing
4. Tedious manual work with running programs is automated,
5. Your scientific investigations are reproducible, scientific reports with top quality typesetting are produced both for paper and electronic devices.

The report: how to write a good scientific/technical report

mostly for the final project useful for homeworks at smaller scale

- An abstract where you give the main summary of your work
- An introduction where you explain the aims and rationale for the physics case and what you have done. At the end of the introduction you should give a brief summary of the structure of the report
- Theoretical models and technicalities. This is the methods section...
- Results and discussion
- Conclusions and perspectives
- Appendix with extra material
- Bibliography

Keep always a good log of what you do.

The report, the abstract

The abstract gives the reader a quick overview of what has been done and the most important results. Here is a typical example taken from a scientific article

We study the collective motion of a suspension of rodlike microswimmers in a two-dimensional film of viscoelastic fluids. We find that the fluid elasticity has a small effect on a suspension of pullers, while it significantly affects the pushers. The attraction and orientational ordering of the pushers are enhanced in viscoelastic fluids. The induced polymer stresses break down the large-scale flow structures and suppress velocity fluctuations. In addition, the energy spectra and induced mixing in the suspension of pushers are greatly modified by fluid elasticity.

The report, the introduction

You don't need to answer all questions in a chronological order. When you write the introduction you could focus on the following aspects

- Motivate the reader, the first part of the introduction gives always a motivation and tries to give the overarching ideas
- What I have done
- The structure of the report, how it is organized etc

The report, discussion of methods

- Describe the methods and algorithms
- You need to explain how you implemented the methods and also say something about the structure of your algorithm and present some parts of your code
- You should plug in some calculations to demonstrate your code, such as selected runs used to validate and verify your results. The latter is extremely important!! A reader needs to understand that your code reproduces selected benchmarks and reproduces previous results, either numerical and/or well-known closed form expressions.

The report, results part

- Present your results
- Give a critical discussion of your work and place it in the correct context.
- Relate your work to other calculations/studies
- An eventual reader should be able to reproduce your calculations if she/he wants to do so. All input variables should be properly explained.
- Make sure that figures and tables should contain enough information in their captions, axis labels etc so that an eventual reader can gain a first impression of your work by studying figures and tables only.

The report, conclusions and perspectives

- State your main findings and interpretations
- Try as far as possible to present perspectives for future work
- Try to discuss the pros and cons of the methods and possible improvements

The report, appendices

- Additional calculations used to validate the codes
- Selected calculations, these can be listed with few comments
- Listing of the code if you feel this is necessary

You can consider moving parts of the material from the methods section to the appendix. You can also place additional material on your webpage.

The report, references

- Give always references to material you base your work on, either scientific articles/reports or books.
- Refer to articles as: name(s) of author(s), journal, volume (boldfaced), page and year in parenthesis.
- Refer to books as: name(s) of author(s), title of book, publisher, place and year, eventual page numbers

How can I use Python and **matplotlib** to make figures for my report

Writing scripts in for example Python to produce high-quality figures allows you in a fast and efficient way to produce scientific results that can be included in a report. Furthermore, many operations can easily be automated, avoding thereby tedious repetitions of commands, as well as possible errors. Here we present a simple Python program which solves parts of project 2 for one quantum mechanical particle in a harmonic oscillator potential. The code plots the radial distribution of the three lowest-lying states, in addition to displaying the lowest three eigenvalues. It is easy to modify the trapping potential and run numerical experiments and test different boundary conditions. The plot is obtained using matplotlib, a Python plotting library which produces publication quality figures in a variety of formats and interactive environments across platforms.

```
#Program which solves the one-particle Schrodinger
#equation for a potential specified in function
#potential(). This example is for 3D harmonic oscillator

from  matplotlib import pyplot as plt
import numpy as np

#-----------------------------------------------------
#Function for initialization of parameters
def initialize():
    RMin = 0.0
    RMax = 10.0
    lOrbital = 0
    Dim = 400
    return RMin, RMax, lOrbital, Dim

# Here we set up the harmonic oscillator potential
def potential(r):
    return r*r
```

```
#Get the boundary, orbital momentum and number
#of integration points
RMin, RMax, lOrbital, Dim = initialize()

#Initialize constants
Step    = RMax/(Dim+1)
DiagConst = 2.0 / (Step*Step)
NondiagConst =  -1.0 / (Step*Step)
OrbitalFactor = lOrbital * (lOrbital + 1.0)

#Calculate array of potential values
v = np.zeros(Dim)
r = np.linspace(RMin,RMax,Dim)
for i in xrange(Dim):
    r[i] = RMin + (i+1) * Step;
    v[i] = potential(r[i]) + OrbitalFactor/(r[i]*r[i]);
```

```
#Setting up a tridiagonal matrix and finding
#eigenvectors and eigenvalues
Hamiltonian = np.zeros((Dim,Dim))
Hamiltonian[0,0] = DiagConst + v[0];
Hamiltonian[0,1] = NondiagConst;
for i in xrange(1,Dim-1):
    Hamiltonian[i,i-1]  = NondiagConst;
    Hamiltonian[i,i]    = DiagConst + v[i];
    Hamiltonian[i,i+1]  = NondiagConst;
Hamiltonian[Dim-1,Dim-2] = NondiagConst;
Hamiltonian[Dim-1,Dim-1] = DiagConst + v[Dim-1];
# diagonalize and obtain eigenvalues, not sorted
EigValues, EigVectors = np.linalg.eig(Hamiltonian)
# sort eigenvectors and eigenvalues
permute = EigValues.argsort()
EigValues = EigValues[permute]
EigVectors = EigVectors[:,permute]
```

```
# plot the results for the three lowest lying eigenstates
for i in xrange(3):
    print EigValues[i]
FirstEigvector = EigVectors[:,0]
SecondEigvector = EigVectors[:,1]
ThirdEigvector = EigVectors[:,2]
plt.plot(r, FirstEigvector**2 ,'b-',r, SecondEigvector**2,'
   'g-',r, ThirdEigvector**2 ,'r-')
plt.axis([0,4.6,0.0, 0.025])
plt.xlabel(r'$r$')
plt.ylabel(r'Radial probability $r^2|R(r)|^2$')
plt.title(r'Radial probability distributions for'+
       ' three lowest-lying states')
plt.savefig('eigenvector.pdf')
plt.show()
```