Educational Administration: Theory and Practice
2023, 29(3), 450-454
ISSN: 2148-2403
https://kuey.net/

Research Article

A Review On Computational Physics And Computational Science Through Python

Jyothi Budida[1*], Dr.R. Padma[2], Dr.Kamala Srinivasan[3]

[1*]Assistant Professor ,Aditya College Of Engineering ,Surampalem ,A.P,India
Jyothi_Bse@Acoe.Edu.In
[2]Department Of Physics, Institute Of Aeronautical Engineering (IARE), Hyderabad, India
Padmavenkatadhri@Gmail.Com
[3]Academic Consultant, S.V.University, Tirupati, A.P, India.
Kamalasrinivasan@Rediffmail.Com

Citation: Jyothi Budida et al. (2023), A Review On Computational Physics And Computational Science Through Python, Educational Administration: Theory and Practice, 29(3), 450-454
Doi: 10.53555/kuey.v29i3.3032

ARTICLE INFO

ABSTRACT

In this article computational physics through python can be discussed. Computational physics is literally physics that uses computations to draw inferences and arrive at conclusions. They are both theoretical disciplines that require a lot of abstract thinking. They are also both very mathematical. A lot of scientific packages are being developed by Python, NumPy, SciPy. In order to use them effectively one should know Python. Some scientific packages writen with C, C++ or FORTRAN work with input files written compulsorily with Python. In computational physics, with Numpy and also Scipy (numeric and scientific library for Python)In theoretical physics worked for 15 years in research, then switched to the computer software industry. In this article, we will discuss about basic steps of programming, background of python& we will solve the Laplace equation using numerical approach rather than analytical/calculus approach and results can be presented.

1. Introduction

In computational physics, we "always" use programming to solve the problem, because computer program can calculate large and complex calculation "quickly". Computational physics can be represented as this diagram. Python is easy to learn, simple to use, and enormously powerful. It has facilities and features for performing tasks of many kinds. You can do art or engineering in Python, surf the web or calculate your taxes, write words or write music, make a movie or make the next billion-dollar Internet start-up. We will not attempt to discuss about all of Python's features, however, but restrict ourselves to those that are most useful for doing physics calculations. We will discuss about the core structure of the language first, how to put together the instructions that make up a program, but we will also discuss about some of the powerful features that can make the life of a computational physicist easier, such as features for doing calculations with vectors and matrices, and features for making graphs and computer graphics. Some other features of Python that are more specialized, but still occasionally useful for physicists, will not be covered here. A good place to start when looking for information about Python is the official Python website at www.python.org. A Python program consists of a list of instructions, resembling a mixture of English words and mathematics and collectively referred to as code There are several different development environments available for use with Python, but the most commonly used is the one called IDLE.
There are so many programming languages that are used today to solve many numerical problems, Matlab for example. But here, we will use Python, the "easy to learn" programming language, and of course, it's free. It also has powerful numerical, scientific, and data visualization library such as Numpy, Scipy, and Matplotlib. Python also provides parallel execution and we can run it in computer clusters.

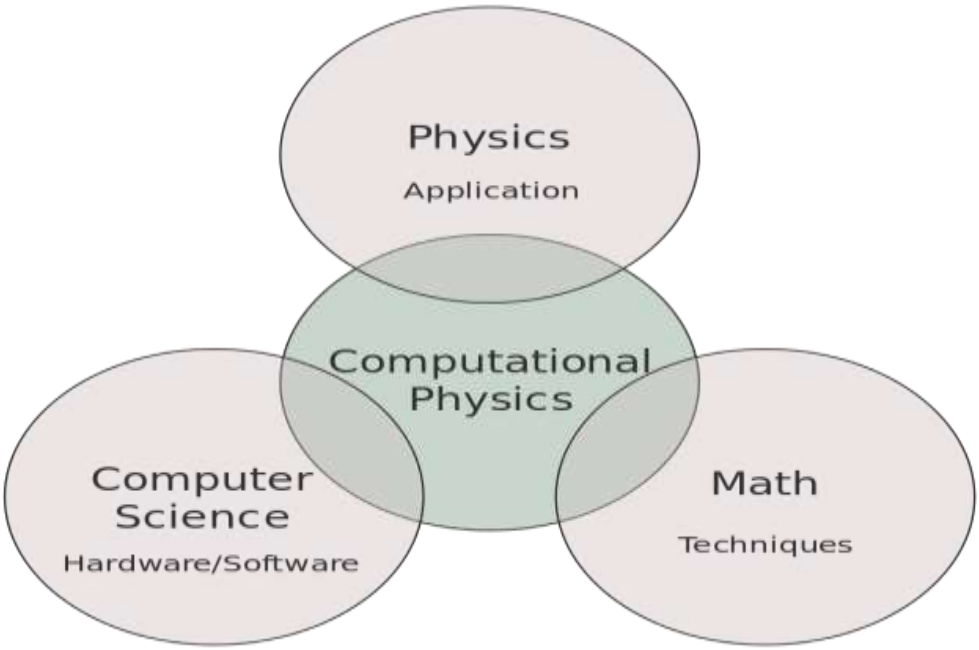

Fig1: Relation of Computational Physics

2 BASIC PROGRAMMING

A program is a list of instructions, or statements, which under normal circumstances the computer carries out, or executes, in the order they appear in the program. Individual statements do things like performing arithmetic, asking for input from the user of the program, or printing out results. The following sections introduce the various types of statements in the Python language one by one.
The Python programming language is an excellent choice for learning, teaching, or doing computational physics. It is a well-designed, modern programming language that is simultaneously easy to learn and very powerful.
Computational physics is the study of scientific problems using computational methods; it combines computer science, physics and applied mathematics to develop scientific solutions to complex problems. Computational physics complements the areas of theory and experimentation in traditional scientific investigation.
Python has become a staple in data science, allowing data analysts and other professionals to use the language to conduct complex statistical calculations, create data visualizations, build machine learning algorithms, manipulate and analyse data, and complete other data-related tasks.
Example applications include the heat capacity of solids, thermal radiation, electrostatics calculations, and image processing.

3. Importance of Computational Physics relates to Python

Introduction Computational physics has been an important research tool in physics for over 40 years. It has enabled physicists to understand complex problems more completely compared to using theoretical and experimental methods alone. Introduction Computational physics has been an important research tool in physics for over 40 years. It has enabled physicists to understand complex problems more completely compared to using theoretical and experimental methods alone.

- Create a code generator.
- Build a countdown calculator.
- Write a sorting method.
- Build an interactive quiz.
- Tic-Tac-Toe by Text.
- Make a temperature/measurement converter.
- Build a counter app.
- Build a number-guessing game.

Using Math function 1.polar.py

```
r = float (input ("Enter r: "))
d = float (input ("Enter theta in degrees: "))
theta = d*pi/180
x = r*cos(theta)
```

```
y = r*sin(theta) print("x =",x," y =",y)
A BALL DROPPED FROM A TOWER
h = float (input("Enter the height of the tower: "))
t = float (input ("Enter the time interval: "))
s = 9.81*t**2/2
print ("The height of the ball is",h-s,"meters")
```

Let us use this program to calculate the height of a ball dropped from a 100 m high tower after 1 second and after 5 seconds. Running the program twice in succession we find the following:

```
Enter the height of the tower: 100
Enter the time interval: 1
The height of the ball is 95.095 meters
Enter the height of the tower: 100
Enter the time interval: 5
The height of the ball is -22.625 meters
```

4. Laplace transform-Heat conduction function using mathematical code

Laplace equation is a simple second-order partial differential equation. It is also a simplest example of elliptic partial differential equation. This equation is very important in science, especially in physics, because it describes behaviour of electric and gravitation potential, and also heat conduction. In thermodynamics (heat conduction), we call Laplace equation as steady-state heat equation or heat conduction equation.

In this article, we will solve the Laplace equation using numerical approach rather than analytical/calculus approach. When we say numerical approach, we refer to discretization. Discretization is a process to "transform" the continuous form of differential equation into a discrete form of differential equation; it also means that with discretization, we can transform the calculus problem into matrix algebra problem, which is favored by programming.

Here, we want to solve a simple heat conduction problem using finite difference method. We will use Python Programming Language, Numpy (numerical library for Python), and Matplotlib (library for plotting and visualizing data using Python) as the tools. We'll also see that we can write less code and do more with Python.

5. Methods and Results

Preparation

To produce the result below, I use this environment:

- **OS**: Linux Ubuntu 14.04 LTS
- **Python**: Python 2.7
- **Numpy**: Numpy 1.10.4
- **Matplotlib**: Matplotlib 1.5.1

Using the code

This is the Laplace equation in 2-D cartesian coordinates (for heat equation):

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Where T is temperature, x is x-dimension, and y is y-dimension. x and y are functions of position in Cartesian coordinates.

In our case, the final discrete equation is shown below.

$$T_{i,j} = \frac{1}{4}(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1})$$

Now, we are ready to solve the equation above. To solve this, we use "guess value" of interior grid (green nodes), here we set it to 30 degree Celsius (or we can set it 35 or other value), because we don't know the value inside the grid (of course, those are the values that we want to know). Then, we will iterate the equation until the difference between value before iteration and the value until iteration is "small enough", we call it convergence. In the process of iterating, the temperature value in the interior grid will adjust itself, it's "self-correcting", so when we set a guess value closer to its actual solution, the faster we get the "actual" solution.

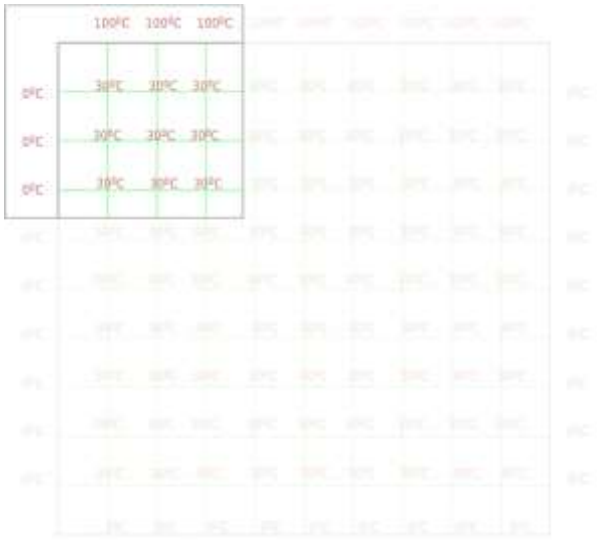

Fig2: Settings of Grids-Temperature contour

np.meshgrid() creates the mesh grid for us (we use this to plot the solution), the first parameter is for the x-dimension, and the second parameter is for the y-dimension. We use np.arange() to arrange a 1-D array with element value that starts from some value to some value, in our case, it's from 0 to lenX and from 0 to lenY. Then we set the region: we define 2-D array, define the size and fill the array with guess value, then we set the boundary condition, look at the syntax of filling the array element for boundary condition above here

```
Configure the contour
plt.title("Contour of Temperature")
plt.contourf(X, Y, T, colorinterpolation, cmap=colourMap)

# Set Colorbar|
plt.colorbar()

# Show the result in the plot window
plt.show()

print("")
```

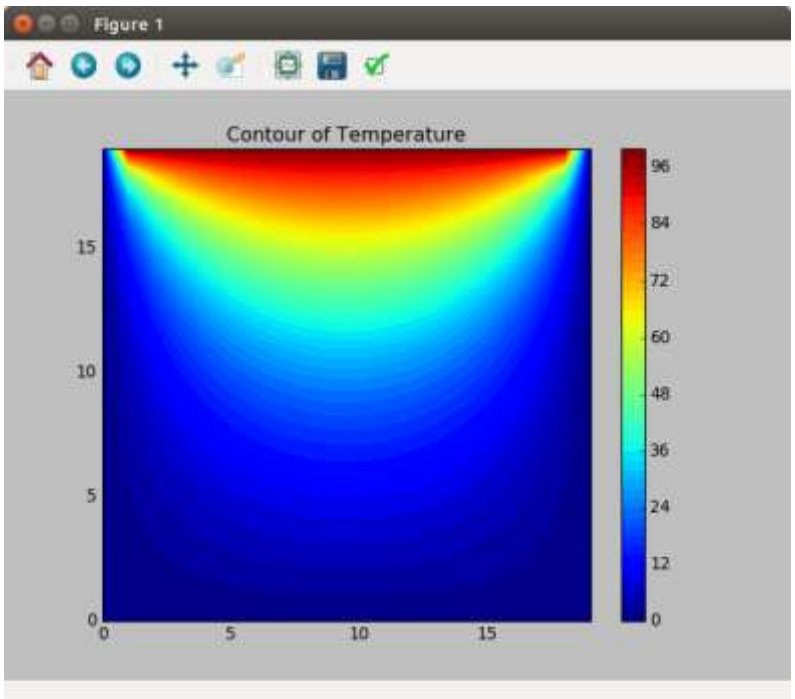

Fig3: Contour of temperature

6. Conclusion

In this article we discussed about how physics can be related with python in basic level, which leads to change in research to solve problems. A lot of scientific packages are being developed by Python, NumPy, SciPy. In order to use them effectively one should know Python. Some scientific packages writen with C, C++ or FORTRAN work with input files written compulsorily with Python. In computational physics, with Numpy and also Scipy (numeric and scientific library for Python), we can solve many complex problems because it provides matrix solver (eigenvalue and eigenvector solver), linear algebra operation, as well as signal processing, Fourier transform, statistics, optimization, etc.In addition to its use in computational physics, Python is also used in machine learning, even Google's TensorFlow uses Python.

References

1. Python: Visual QuickStart Guide, Chris Fehily, Peachpit Press, 2001, 0-201- 74884-3.
2. http://matplotlib.sourceforge.net/index.html
3. Random Walk and Percolation, Patrick Dickstein.
4. http://code.google.com/p/qutip/
5. http://www.physics.usyd.edu.au/~wheat/dpend_html/
6. Raise of Python , 2023 September http://dx.doi.org/10.13140/RG.2.2.27388.92809
7. Python current trend applications-an overview, October 2019,IJAERD Volume 6, Issue 10, October -2019.
8. K. R. Srinath, “Python – The Fastest Growing Programming Language,” International Research Journal of
9. Engineering and Technology (IRJET), vol. 4, Issue 12 pp. 354–357, December 2017.
10. Kalyani Adawadkar, “Python Programming-Applications and Future,” International Journal of Advance Engineering and Research Development(IJAERD), Special Issue SIEICON-2017, pp. 1–4, April -2017.