P346/745 : Physics Computer Lab

Credit : **6** (Int-MSc Sem-V)

Credit : **4** (Int-MSc-PhD Sem-III)

Classes : **LH-1** every Wednesday & Thursday @ 2:30 pm

Programming language : **Python**, **C/C++** only

- *Programming language(s) will not be taught in this course*
- *Must bring laptop in class for code writing*
- *Allowed platform : Jupyter (Colab). Having Github account is recommended, particularly for* **C/C++** *users*
- All assignments, DIY, Midsem and Endsem codes and reports should be uploaded in or linked to Google classroom

Marks distribution : Assignments 40, Midsem 15, DIY 15, Endsem 30

Instructor : Subhasish Basak (SPS-507, sbasak@niser.ac.in)
TA : Sandeep Pradhan (SPS-312, sandeep.pradhan@niser.ac.in)

P346/745 : Physics Computer Lab

Assignments	• One assignment for each topics discussed, including one warm-up assignment (compulsory but marks will be considered only for those who fail to secure pass marks otherwise). • Must be submitted / uploaded within class hours or due date, else 20% marks will be deducted. • If suspected of copying, assignment(s) will be awarded **zero** to all parties involved without giving a chance to defend. • Wi-fi must be turned off or kept in Airplane mode during class hours
Midsem	• Lab courses do not have Midsem but there will be one on the first class after Midsem recess.
DIY	• Problems to be offered before Midsem recess and submitted (codes and report) before Endsem
Endsem	• Full syllabus and mandatorily use of own library functions (not system-built)

P346/745 : Physics Computer Lab

Grading : You may find it controversial

2022 :	51 out of 55 students got $\geq 75/100$
	AA (6) in 95 – 100, **AB** (12) in 88 – 94,
	BB (11) in 84 – 87, **BC** (11) in 76 – 83 etc.
2021 :	48 out of 53 students got $\geq 75/100$
	AA (6) in 96 – 100, **AB** (7) in 91 – 95,
	BB (15) in 86 – 90, **BC** (14) in 81 – 85 etc.
2020 :	47 out of 49 students got $\geq 75/100$
	AA (7), **AB** (14), **BB** (15), **BC** (7) etc.

The statistics in 2025 may change becasue all assignments have to done in class except for DIY. But if not then grading is not expected to be very different. In such case, the same grading scheme will be followed this year too.

Not more than 10% of class strength will be awarded **AA**

But grading scheme may be different (absolute ?) for Int. MSc-PhD because of different credit requirement.

P346/745 : Physics Computer Lab

Syllabus :

- Random number generation
- Gauss-Jordan elimination – solving linear algebraic equations, inverse, determinant
- LU decomposition – solving linear algebraic equations, inverse
- Jacobi or Gauss-Seidel – iterative solution of the above
- Roots finding – Regula falsi, Newton-Raphson, Laguerre
- Integration – Midpoint, Trapezoidal, Simpson, Monte Carlo
- Ordinary differential equation – Euler, Predictor-Corrector, Runge-Kutta (both initial and boundary value problems)
- Least square fitting
- *if time permits* : PDE & Eigen value problem
- *if time permits* : ML regression and classification

P346/745 : Physics Computer Lab

Programming practices :

1. Each code should have a comment header explaining the purpose of the program, name and roll number.
2. Input should NOT be interactive or *hardwired*. All input must be read from external file(s), whether it is matrix dimension(s) or precision or characters/strings for file name or any other sorts of numbers.
3. All the output should be given in a separate file(s) and must be exactly the same that the code will generate. Without output, the problem will be considered incomplete. Additional comments (if any) must be appended after the output and clearly separated from the actual output itself.
4. Codes should contain decent amount of comments explaining the computing steps (not too many nor too few), which will help others to follow the code.

P346/745 : Physics Computer Lab

Programming practices contd. ...

5. Initialization must be done in loop when needed such as reading a matrix or a series of coefficients etc.
6. Each routines or functions like your own random number generator or LU decomposition or RK4 MUST be in a single library file (which obviously will grow in size as you put more and more functions in it) and NOT in the main front code.
7. The main body of the code or the front code will only involve I/O, allocation of memories, opening of files etc. and calls to routines or functions.
8. No system-built routines, like Numpy, Scipy etc., for the major algorithms can be used. For instance, `random` or `inverse` function cannot be called. But, of course, you can use basic stuff like `sqrt()`, `sin()` or `log()` etc.
9. Whenever in doubt, ask me!

P346/745 : Physics Computer Lab

An example program in C

```c
// Code for mean and s.d. of data to be read from a file
// Name : Imagine Dragon, Roll No : 007

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc,cahr *argv[]){

  int Ndata,idx,i,j;
  float *data,av=0,sd=0;
  FILE *fpin;

  // echo usage
  if(argc != 3){
    printf("Usage␣:␣./avsd␣<no_of_data_points>␣<data_filename>\n")
    exit(1);}

  // initialize from command line arguments
  Ndata=atoi(argv[1]);
  fpin=fopen(argv[2],"r");
```

P346/745 : Physics Computer Lab

C example program contd. . . .

```
  // create data space and read
  data=(float *)malloc(Ndata*sizeof(float));
  for(i=0;i<Ndata;i++) fscanf(fpin,"\%d␣\%f",\&idx,\&data[i]);
  fclose(fpin);

  // calculate average & sd
  for(i=0;i<Ndata;i++) av+=data[i]/(float)Ndata;
  for(i=0;i<Ndata;i++)
     sd+=(data[i]-av)*(data[i]-av)/(float)Ndata;
  sd=(float)sqrt((double)sd);

  // print result
  printf("average␣=␣%.4f,␣s.d.␣=␣%.4f\n",av,sd);

  return 0;

} // end of main
```

```
###########################################################
average = 2.4671, s.d. = 0.0023
###########################################################
```

P346/745 : Physics Computer Lab

An example program in Python

```python
# Code for mean and s.d. of data to be read from a file
# Name : Imagine Dragon, Roll No : 007

import sys
import math

def main():
  # check command line arguments
  if len(sys.argv)!=3:
    print("Usage:␣python3␣avsd.py␣<no_of_data_points>␣<data_filena
    sys.exit(1)
  Ndata = int(sys.argv[1])
  filename = sys.argv[2]
  data = []

  # Read data from file
  try:
    with open(filename,'r') as f:
         for _ in range(Ndata):
             line=f.readline()
             if not line:
                break
```

Python example program contd. . . .

```
            idx_str, value_str=line.strip().split()
            value=float(value_str)
            data.append(value)

    # calculate average and sd
    average=sum(data)/Ndata
    variance=sum((x-average)**2 for x in data)/Ndata
    sd=math.sqrt(variance)

    # print result
    print(f"average = {average:.4f}, s.d. = {sd:.4f}")

if __name__ == "__main__":
    main()
```

```
###############################################################
average = 2.4671, s.d. = 0.0023
###############################################################
```

P346/745 : Physics Computer Lab

Git Basics

- **Git**
  - Tool to track and manage code changes
  - Supports version control and collaboration
- **Staging Area**
  - Temporary area to prepare changes
  - Use `git add <filename>` to stage
- **Local Repository**
  - Git folder on your system
  - Use `git commit -m "message"` to save
- **Remote Repository**
  - Log in to your GitHub account
  - Create a new repository (click **New**)
  - Copy the SSH/HTTPS URL
  - Link to local: `git remote add origin <URL>`
  - Upload: `git push -u origin main`

P346/745 : Physics Computer Lab

An Example of git workflow

Listing 1: Creating Local Git Repo

```
# Step 1: Create and move to project folder
mkdir Computational_physics_Examples
cd Computational_physics_Examples

# Step 2: Initialize Git
git init
# Output: Initialized empty Git repository...

# Step 3: Add some content
echo "an example of git workflow" >> chapter1.txt

# Step 4: Check Git status
git status
# Output: chapter1.txt is untracked

# Step 5: Track and commit changes
git add chapter1.txt
git commit -m "create chapter1"
# Output: 1 file changed, 1 insertion(+) ...
```

P346/745 : Physics Computer Lab

Git workflow example contd. ...

Listing 2: Creating Local Git Repo

```
# check commit history
git log
# output : commit 9f432d4bdc117e40d5b7ca7785c32d4ec23e6c7d (HEAD —
#Author: Sandeep Pradhan ..
# Step 6: Add another file
echo "creating␣chapter2" >> chapter2.txt
git add chapter2.txt
git commit —m "add␣chapter2"
# checkout the first commit , have all the changes at the first co
git checkout 9f432d4bdc117e40d5b7ca7785c32d4ec23e6c7d
```

P346/745 : Physics Computer Lab

Git workflow example contd. . . .

Listing 3: Push to GitHub and Pull Updates

```
# Step 7: Add remote GitHub repository, if not added
git remote add origin git@github.com:sandeep-phyML/
                                computational_physics_lab.git
git remote -v
# Output:
# origin   git@github.com:... (fetch)
# origin   git@github.com:... (push)

# Step 8: Push code to GitHub
git push -u origin main
# Output:
#Enumerating objects: 6, done.
#Counting objects: 100% (6/6), done.
#Delta compression using up to 8 threads
#Compressing objects: 100% (3/3), done.
## check it in the github repository
```

P346/745 : Physics Computer Lab

Git workflow example contd. ...

Listing 4: Push to GitHub and Pull Updates

```
# Step 9: Create new branch and pull updates
git switch -c new_branch
git pull origin main

# Output:
#From github.com:sandeep-phyML/computational_physics_lab
 * branch            main       -> FETCH_HEAD
#Updating 9f432d4..95528d8

i
```