PH 3120
Computational Physics Laboratory

This practical course focuses on providing the student with hands-on learning in computing through relevant laboratory work. The course involves exercises in numerical techniques with applications in theoretical and experimental physics.

Intended Learning Outcomes

At the end of this course, students are expected to be able to;

- appreciate the use of computational methods to solve problems in physics,
- visualize experimental data and understand the underlying systems by forming theoretical models,
- represent experimental and theoretical data in suitable graphical and mathematical forms,
- solve differential equations numerically for analysing linear and non-linear physical systems,
- perform computer-based simulations of complex systems and analyse their behaviour,
- analyse signals in frequency and time domain, and use frequency domain analysis to remove noise,
- present results of numerical calculations in the form of written reports.

Teaching Learning Assessment

- Short lectures/discussions
- Weekly laboratory exercises (reports)
- Weekly laboratory assessments
- Semester end exam

- Continuous assessment (completion of exercises, written reports, weekly assessment, viva) - 60%
- End of semester practical test - 40%

Rules and Guidelines

- Lab time – Wednesday, 8.00 am – 3.00 pm (6 hours)
- Students should report to the lab on time, stay through the lab until the end
- Assignments/reports should be submitted via LMS every week
- You can discuss but do not copy
- No excuses unless a medical is provided

Introduction to Python and Jupyter Notebook

Python is a high-level language that can be used for Visualization, Programming, algorithm development, Numerical computation: linear algebra, optimization, control, statistics, signal and image processing, etc.

The **Jupyter Notebook** is a web-based interactive computing platform.

Installing Python and Jupyter Notebook

Variables, Expressions and Statements

- Variables and basic operations

```
a = 15
b = 6
print(a+b)
print(a-b)
print(a/b)
print(a*b)
print(a**b)
Print(a%b)
```

- Strings

```
print(“Hello World”)
```

Variables, Expressions and Statements

- Types of variables

```
x = 22
print(x)
type(x)
```

```
22
```

```
int
```

- User inputs

```
a = float(input('Enter the value of a '))
print('The value you entered is ',a)
```

```
Enter the value of a 10
The value you entered is  10.0
```

e.g. Write a code that asks the user to input the components of a 3-dimensional vector to calculate the magnitude of the vector.

Conditional Statements

```
a = 10
if a > 5:
    print('a is greater than 5')
elif a == 5:
    print('a is equal to 5')
else:
    print('a is less than 5')
```

```
a is greater than 5
```

Indentation is important !

e.g. Write a code that asks the user to enter the temperature of water and output the state (ice, liquid, vapour) of water.

User defined functions

- Used for reusable pieces of codes

```
def area(r):
    pi = 3.14
    area = pi*r**2
    return area

a = area(5)
print(a)
```

```
78.5
```

e.g., Write a function to calculate the surface area of a cylinder with radius *r* and height *h*.

Loops and Iterations

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

```
names = ['John', 'Jane','Jack']
for names in names:
    print('Good morning ',names)
```

```
Good morning  John
Good morning  Jane
Good morning  Jack
```

```
i = 0
while i < 5:
    print(i)
    i = i+1
```

```
0
1
2
3
4
```

Write a program to print first 5 fibbachi numbers

Write a program that asks the user to enter the password and let the user in if the password is correct if not, ask it again

Lists

- Lists are used to store multiple items in a single variable.

```
lst = [1,'apple',3.4,'cherry',True]
print(type(lst))
print(len(lst))
print(lst[3])
if "apple" in lst:
    print("Yes, apple is in the list")

lst.append(45)
print(lst)

lst[1] = 'banana'
print(lst)

lst.insert(1,'apple')
print(lst)

lst1 = [1,2,3]
lst2 = [4,5,6]
lst1.extend(lst2)
print(lst1)
```

```
<class 'list'>
5
cherry
Yes, apple is in the list
[1, 'apple', 3.4, 'cherry', True, 45]
[1, 'banana', 3.4, 'cherry', True, 45]
[1, 'apple', 'banana', 3.4, 'cherry', True, 45]
[1, 2, 3, 4, 5, 6]
```

Numpy arrays

- Faster than lists
- Not as flexible as lists

```
import numpy as np

arr0 = np.array(25)
print(arr0)

arr1 = np.array([1,2,3,4,5])
print(arr1)
print(arr1[0])

arr2 = np.array([[1,2,3],[4,5,6]])
print(arr2)
print(arr2[0,0])
```

```
25
[1 2 3 4 5]
1
[[1 2 3]
 [4 5 6]]
1
```

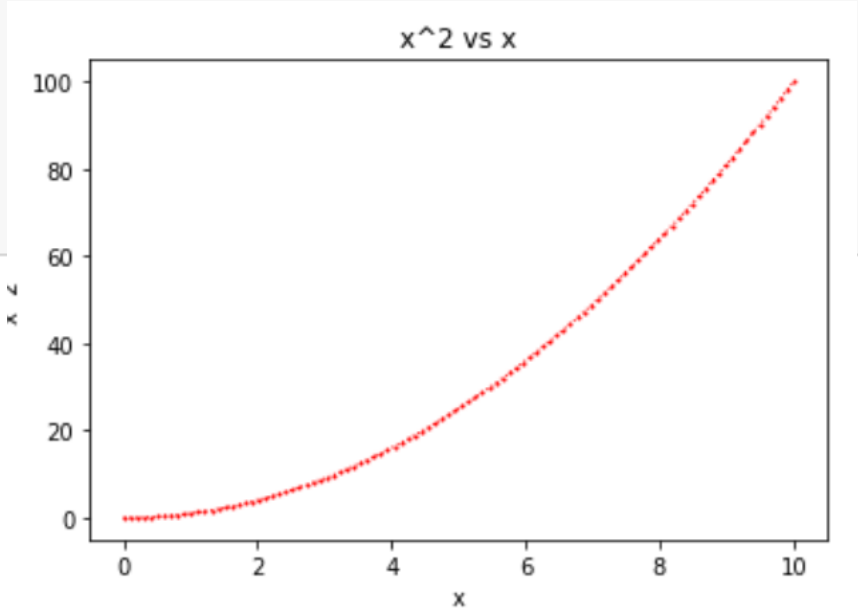
Creating arrays

```
import numpy as np

x = np.linspace(0,1,10)
print(values)
y = x**2
print(y)
```

```
[0.         0.11111111 0.22222222 0.33333333 0.44444444 0.55555556
 0.66666667 0.77777778 0.88888889 1.        ]
[0.         0.01234568 0.04938272 0.11111111 0.19753086 0.30864198
 0.44444444 0.60493827 0.79012346 1.        ]
```

Plotting

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0,10,100)
y = x**2
plt.plot(x,y,'o',linestyle = 'dotted',color = 'r',linewidth = '1',markersize = '1')
plt.title("x^2 vs x")
plt.xlabel("x")
plt.ylabel("x^2")
plt.savefig('plot.png')
plt.show()
```



Reading from a file

file = open('myfile.txt')

```
import numpy as np
data = np.loadtxt('D:\\Teaching\\PH 3120\\Week 1\\distance.dat')
```

```
type(data)
```

```
numpy.ndarray
```

```
data.shape
```

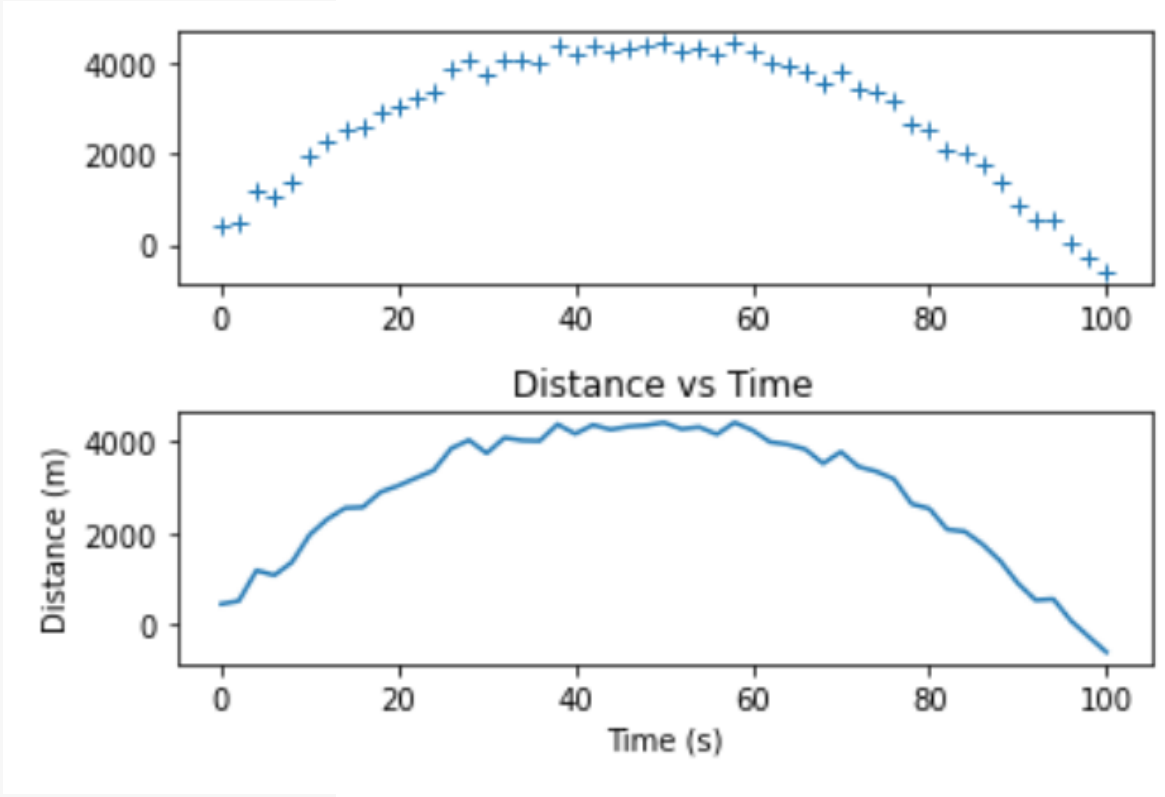
```
(51, 3)
```

```
for i in range(5):
    print(data[i,:])
```

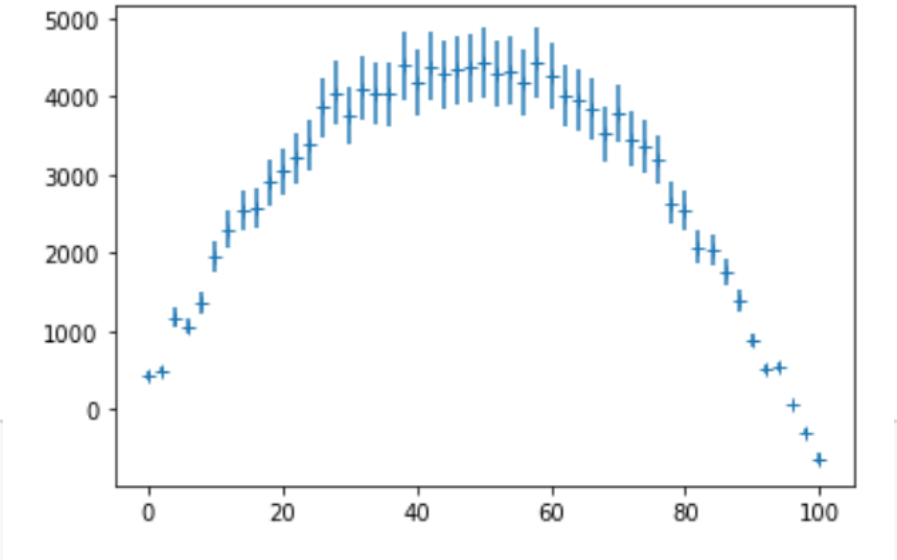
```
[  0.    429.857  43.   ]
[  2.    497.775  50.   ]
[   4.    1171.703  117.   ]
[   6.    1061.673  106.   ]
[   8.    1350.768  135.   ]
```

Plotting data from file

```
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt('D:\\Teaching\\PH 3120\\Week 1\\distance.dat')
x = data[:,0]
y = data[:,1]
err = data[:,2]

fig = plt.figure()
fig.subplots_adjust(hspace=0.5, wspace=5.0)

plt.subplot(211)
plt.plot(x,y,'+')

plt.subplot(212)
plt.plot(x,y)

plt.title("Distance vs Time")
plt.xlabel("Time (s)")
plt.ylabel("Distance (m)")

plt.show()
```



Plot error bars




```
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt('D:\\Teaching\\PH 3120\\Week 1\\distance.dat')
x = data[:,0]
y = data[:,1]
err = data[:,2]

plt.errorbar(x,y,marker = '+',yerr = err,ls = 'none')
```

Writing to a file

```
import numpy as np
import matplotlib.pyplot as plt
arr = np.empty([50,2])
x = np.linspace(0,10,50)
y = 3.4*x + 0.2
arr[:,0] = x
arr[:,1] = y
plt.plot(x,y,'*')
plt.show()
np.savetxt('mydata.dat', arr)
```