Introductory Scientific Computing with Python

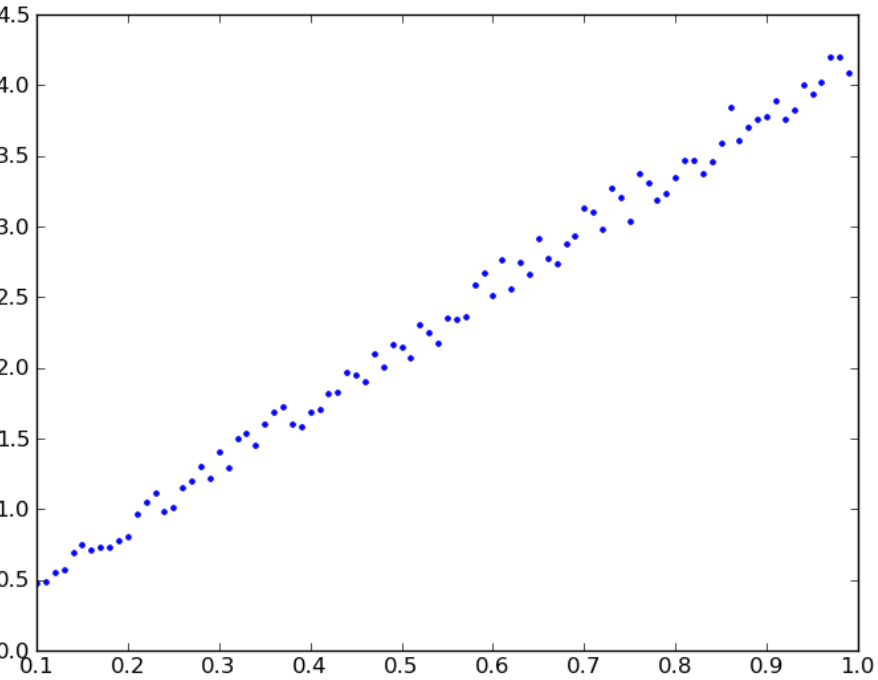
Exercises

FOSSEE

Department of Aerospace Engineering
IIT Bombay

Mumbai, India

Problem 1



Example code

```
In []: l, t = loadtxt('pendulum.txt',
                      unpack=True)
In []: plot(l, t, '.')
```

Problem Statement

Tweak above code to plot data in file **`pos.txt`**.

Problem 1 cont...

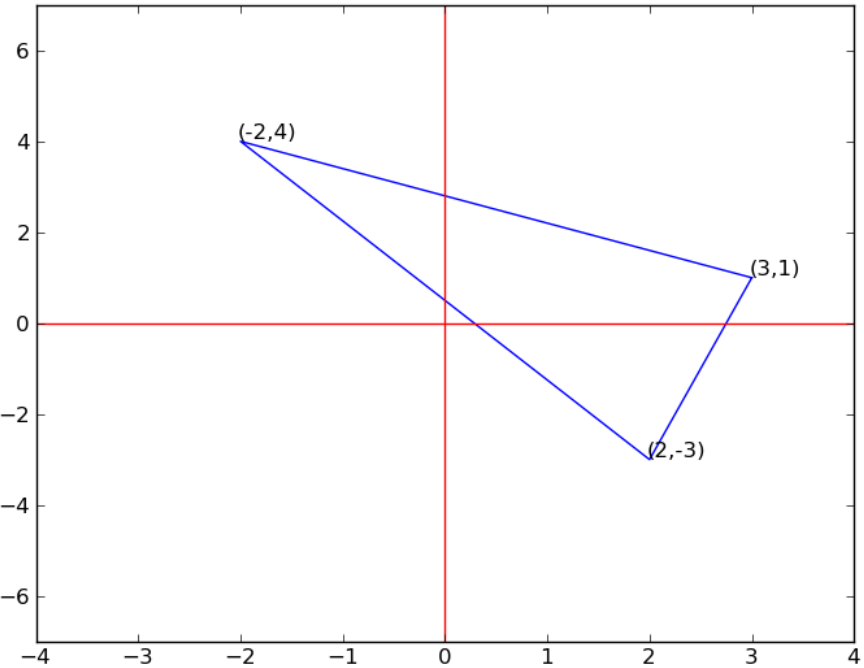
- Label both the axes.
- What kind of motion is this?
- Title the graph accordingly.
- Annotate the position where vertical velocity is zero.

Solution

```
x, y = loadtxt('pos.txt', unpack=True)
plot(x, y)
xlabel('x')
ylabel('y')
title('Projectile motion')
annotate('v = 0', xy=(5, 4.75))
# Or
annotate(r'$v_y = 0$', xy=(5, 4.75))
```

- Note the LaTeX syntax
- Note the raw strings: `r'$v_y = 0$'`

Problem 2



Plot points given x and y coordinates

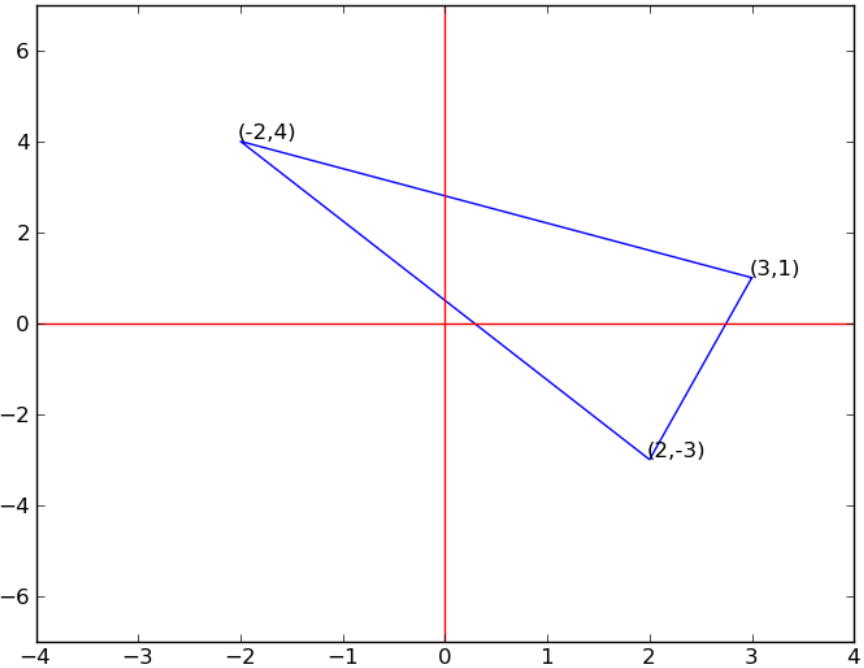
```
In []: x = [3, 2, -2, 3]
In []: y = [1, -3, 4, 1]
In []: plot(x, y)
```

Line can be plotted using arrays of coordinates.

Problem statement

Write a Program that plots a regular n-gon(Let n = 5).

Problem 2



Plot points given x and y coordinates

```
In []: x = [3, 2, -2, 3]
In []: y = [1, -3, 4, 1]
In []: plot(x, y)
```

Line can be plotted using arrays of coordinates.

Problem statement

Write a Program that plots a regular n-gon(Let n = 5).

Solution

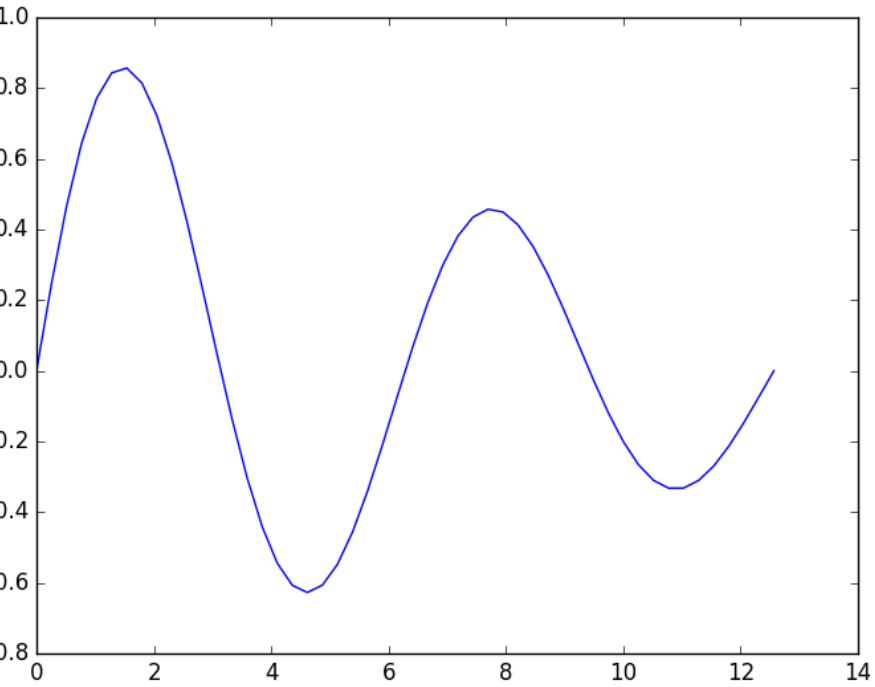
```
n = 5
t = linspace(0, 2*pi, n+1)
x = cos(t)
y = sin(t)
plot(x, y)
axis('equal')
```

Better Solution

```
def plot_ngon(n):
    t = linspace(0, 2*pi, n+1)
    x = cos(t)
    y = sin(t)
    plot(x, y)
    axis('equal')

plot_ngon(5)
clf()
plot_ngon(10)
```

Problem 3

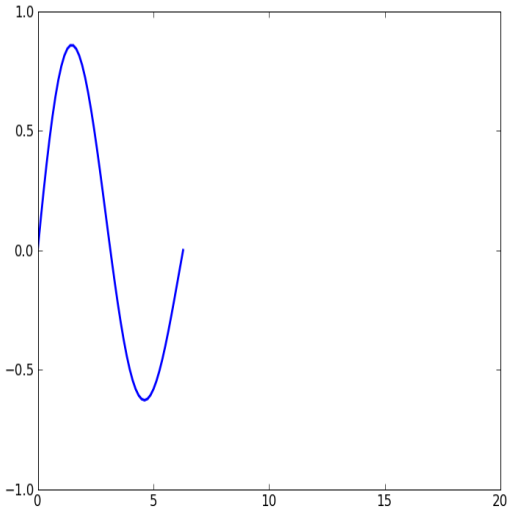


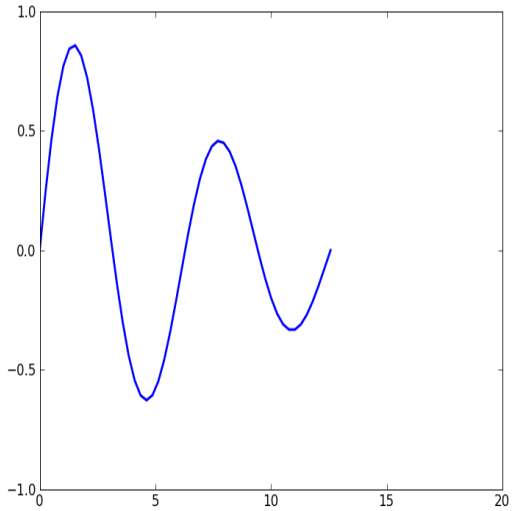
Damped Oscillation

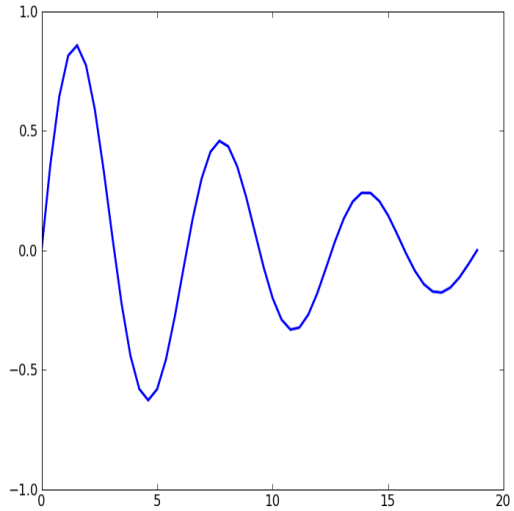
```
In []: t = linspace(0, 4*pi)
In []: plot(t, exp(-t/10)*sin(t))
```

Problem 3 cont...

Create a sequence of images (say 10) in which the damped oscillator($e^{-t/10}sin(t)$) slowly evolves over time t.







Hint

```
savefig('plot'+str(i)+'.png') #i is some int
```

Naive Solution

```
for i in range(1, 11):
    t = linspace(0, 0.5*pi*i, 50*i)
    clf()
    plot(t, exp(-t/10)*sin(t))
    savefig('plot' + str(i) + '.png')
```

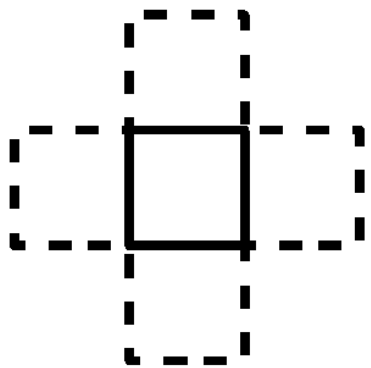
Better Solution

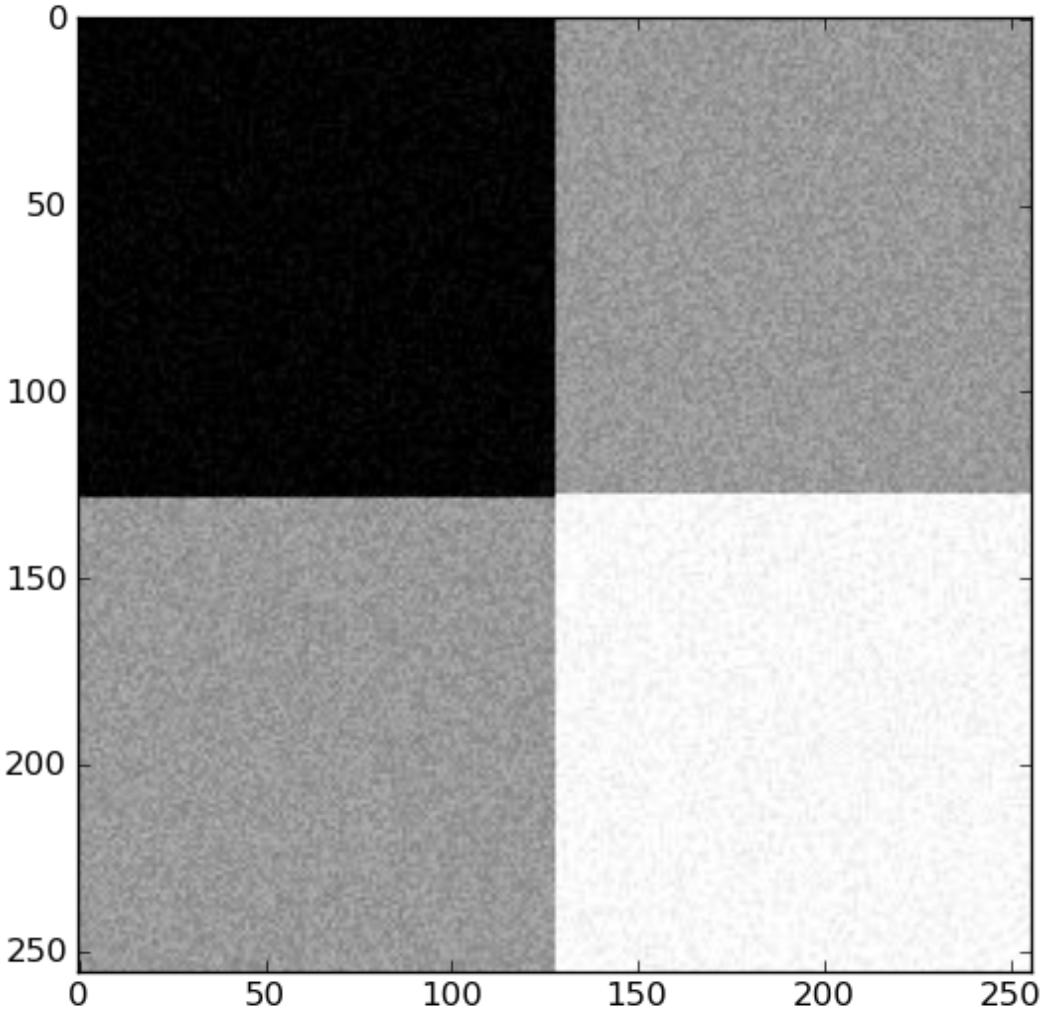
```
for i in range(1, 11):
    t = linspace(0, 0.5*pi*i, 50*i)
    clf()
    plot(t, exp(-t/10)*sin(t))
    xlim(0, 5*pi)
    ylim(-1, 1)
    savefig('plot' + str(i) + '.png')
```

Problem 4

```
In [ ]: x = imread('smoothing.gif')
In [ ]: x.shape
Out[ ]: (256, 256)
In [ ]: imshow(x,cmap=cm.gray)
In [ ]: colorbar()
```

Replace each pixel with mean of neighboring pixels

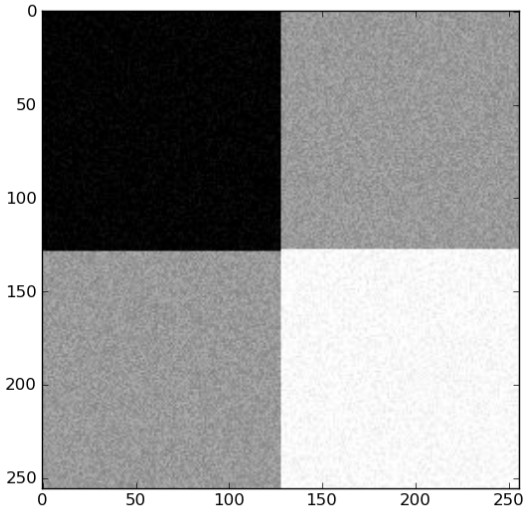


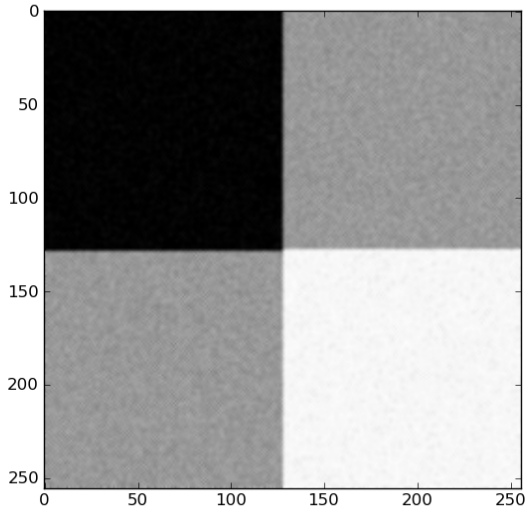

0
50
100
150
200
250
0
50
100
150
200
250

Problem 4: Approach

For `y` being resultant image:

```
y[1, 1] = x[0, 1]/4 + x[1, 0]/4
          + x[2, 1]/4 + x[1, 2]/4
```





Hint:

Use array Slicing.

Solution

```
In [ ]: y = zeros_like(x)
In [ ]: y[1:-1,1:-1] = x[:-2,1:-1]/4 +
                       x[2:,1:-1]/4 +
                       x[1:-1,2:]/4 +
                       x[1:-1,:-2]/4
In [ ]: imshow(y,cmap=cm.gray)
```

Problem 4 cont. . .

- Apply the smoothing operation repeatedly to the original image
- Subtract the smoothed image from the original to obtain the edges

Problem 5

What if you did the following in problem 4?

```
In []: y1 = zeros_like(x)
In []: y1[1:-1,1:-1] = (x[:-2,1:-1] +
                        x[2:,1:-1] +
                        x[1:-1,2:] +
                        x[1:-1,:-2])/4
```

Are the answers different?

Problem 5 cont. . .

Why? The answer lies in the following:

```
In [ ]: x.dtype
Out[ ]: dtype('uint8')

In [ ]: print(x.itemsize)
1

In [ ]: z = x/4.0

In [ ]: print(z.dtype)
float64
```

Problem 5 cont. . .

What if you did this?

```
x = imread('smoothing.gif')
y2 = zeros_like(x)
y2[1:-1,1:-1] = x[:-2,1:-1]/4. + \
                x[2:,1:-1]/4.  + \
                x[1:-1,2:]/4.  + \
                x[1:-1,:-2]/4.
```

- Will the answer be any different from `y`?
- What will the dtype of `y2` be?
- Discuss what is going on!

Problem 5 cont. . .

- Did you do the right thing to find the edges earlier in problem 4?
- If not, fix it!

Note that:

```
In []: print(x.dtype)
uint8
In []: x1 = x.astype('float64')
In []: print(x1.dtype)
float64
In []: print(x.dtype.char)
d
In []: x.dtype.<TAB> # Explore!
```

Problem 6

Edge detection looks much nicer with **`lena.png`**, try it! The caveat is that it is a 4 component RGBA image with elements in the range $[0.0, 1.0]$.

```
In [ ]: x = imread('lena.png')

In [ ]: print(x.shape)
(512, 512, 4)

In [ ]: print(x.min(), x.max())
(0.0, 1.0)
```

Repeat the edge detection with this image.