Programming for School

G V V Sharma*

Contents

1	**Simultaneous Equations**	1
2	**Graphical Solution**	2
3	**C programming**	3
4	**Python programming exercises**	4
5	**C programming exercises**	5

***Abstract*—This manual introduces Python and C programming through basic geometry.**

1 Simultaneous Equations

1.1 Consider the equations

$$x_1 + x_2 = 8 \tag{1}$$
$$3x_1 - x_2 = 12 \tag{2}$$

Write (1) as a matrix equation.
Solution: (1) can be expressed as

$$\begin{pmatrix} 1 & 1 \\ 3 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 8 \\ 12 \end{pmatrix} \tag{3}$$

1.2 Let

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 3 & -1 \end{pmatrix} \tag{4}$$

Find det($\mathbf{A}$).
Solution: The *determinant* is obtained as

$$\det(\mathbf{A}) = 1 \times -1 - 3 \times 1 = -4. \tag{5}$$

1.3 Write a program for finding $\det(\mathbf{A})$.
Solution: The following program finds the determinant.

```
#Code by GVV Sharma
#March 14, 2019
#released under GNU GPL
import numpy as np

a1 = 1
a2 = 1
b1 = 3
b2 = −1
c1 = 8
c2 = 12

A = np.array(([a1,a2],[b1,b2]))
x = np.linalg.det(A)
print(x)
```

1.4 Write your own function for calclating $\det \mathbf{A}$
Solution: The following routine finds the determinant.

```
#Code by GVV Sharma
#March 14, 2019
#released under GNU GPL
import numpy as np

def det(A):
   a1 = A[0][0]
   a2 = A[0][1]
   b1 = A[1][0]
   b2 = A[1][1]
   y = a1*b2 − a2*b1
   return y


a1 = 1
a2 = 1
b1 = 3
b2 = −1

A = np.array(([a1,a2],[b1,b2]))
```

*The author is with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: gadepall@iith.ac.in.

```
x = det(A)
print(x)
```

1.5 Write a program to check if two lines intersect.
Solution: Two lines intersect if $\det(\mathbf{A}) \neq 0$. The following code checks for this condtion.

```
#Code by GVV Sharma
#March 16, 2019
#released under GNU GPL
import numpy as np



a1 = 1
a2 = 1
b1 = 3
b2 = −1

A = np.array(([a1,a2],[b1,b2]))
x = np.linalg.det(A)
if x != 0:
  print('The lines intersect')
else:
  print('The lines do not intersect')
```

1.6 Let

$$\mathbf{A}_1 = \begin{pmatrix} 8 & 1 \\ 12 & -1 \end{pmatrix} \tag{6}$$

$$\mathbf{A}_2 = \begin{pmatrix} 1 & 8 \\ 3 & 12 \end{pmatrix} \tag{7}$$

Verify that

$$x_1 = \frac{\det \mathbf{A}_1}{\det \mathbf{A}} \text{ and } x_2 = \frac{\det \mathbf{A}_2}{\det \mathbf{A}} \tag{8}$$

satisfy (1).
Solution:

$$\frac{\det \mathbf{A}_1}{\det \mathbf{A}} = \frac{-20}{-4} = 5 \tag{9}$$

Similarly,

$$\frac{\det \mathbf{A}_2}{\det \mathbf{A}} = \frac{-12}{-4} = 3 \tag{10}$$

2 Graphical Solution

2.1 Find a graphical solution for (1).
Solution: The follwoing code plots Fig. 2.1. It is obvious that the two equations in (1) represent the lines y_1 and y_2 in Fig. 2.1 and intersect at $\begin{pmatrix} 5 \\ 3 \end{pmatrix}$

```
#Code by GVV Sharma
#March 14, 2019
#released under GNU GPL
import numpy as np
import matplotlib.pyplot as plt

#if using termux
import subprocess
import shlex
#end if

x = np.linspace(−2,8,20)
y1 = 8−x
y2 = 3*x−12

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

# Major ticks every 2, minor ticks every 1
major_ticks = np.arange(−10, 10, 2)
minor_ticks = np.arange(−10, 10, 1)

ax.set_xticks(major_ticks)
ax.set_xticks(minor_ticks, minor=True)
ax.set_yticks(major_ticks)
ax.set_yticks(minor_ticks, minor=True)


# If you want different settings for the grids:
ax.grid(which='minor', alpha=0.2)
ax.grid(which='major', alpha=0.5)

#Plotting all lines
ax.plot(x,y1,label='$y_1$')
ax.plot(x,y2,label='$y_2$')


plt.xlabel('$x$')
plt.ylabel('$y$')
ax.legend(loc='best')

#if using termux
plt.savefig('../figs/draw_line.pdf')
plt.savefig('../figs/draw_line.eps')
subprocess.run(shlex.split("termux−open ../
    figs/draw_line.pdf"))
#else
#plt.show()
```

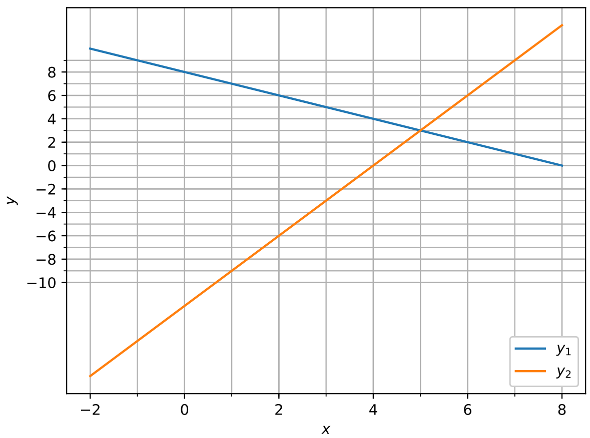

Fig. 2.1

2.2 The **np.linspace** function above generates an arithmetic sequence with first term -2, last term 8 and number of terms 20. Write your own linspace function and verify.
Solution: The code is available below.

```
#Code by GVV Sharma
#March 14, 2019
#released under GNU GPL
import numpy as np

def linspace(first,last,k):
  t= np.zeros((k,1))
  t[0]=first
  d = (last−first)/(k−1)
  for n in range(2,k):
    t[n−1] = t[0]+(n−1)*d
  return t

x = np.linspace(−2,8,20)
print(x)
```

3 C Programming

3.1 Write a C program to generate an arithmetic sequence with $t_0 = -2, t_{n-1} = 8, n = 20$ and print it to the file **ap.dat**.
Solution:

```
#include <stdio.h>

int main(void)
{
FILE *fp;
float t_0 = −2.0, t_k = 8.0, d,t_n;
int k = 20, n;

//Common difference
d = (t_k−t_0)/(k−1);
fp = fopen(”ap.dat”,”w”);
for(n = 0; n < k; n++)
{
t_n = t_0+n*d;
printf(”%f\n”,t_n);
fprintf(fp,”%f\n”,t_n);
}
fclose(fp);
return 0;
}
```

3.2 Now execute the following code.

```
#Code by GVV Sharma
#March 15, 2019
#released under GNU GPL
import numpy as np
import matplotlib.pyplot as plt

#if using termux
import subprocess
import shlex
#end if

x = np.loadtxt('ap.dat',dtype='float')
y1 = 8−x
y2 = 3*x−12

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

# Major ticks every 2, minor ticks every 1
major_ticks = np.arange(−10, 10, 2)
minor_ticks = np.arange(−10, 10, 1)

ax.set_xticks(major_ticks)
ax.set_xticks(minor_ticks, minor=True)
ax.set_yticks(major_ticks)
ax.set_yticks(minor_ticks, minor=True)


# If you want different settings for the grids:
ax.grid(which='minor', alpha=0.2)
ax.grid(which='major', alpha=0.5)

#Plotting all lines
```

```
ax.plot(x,y1,label='$y_1$')
ax.plot(x,y2,label='$y_2$')


plt.xlabel('$x$')
plt.ylabel('$y$')
ax.legend(loc='best')

#if using termux
plt.savefig('../figs/draw_line.pdf')
plt.savefig('../figs/draw_line.eps')
subprocess.run(shlex.split("termux-open ../
    figs/draw_line.pdf"))
#else
#plt.show()
```

3.3 Do all computations in Problem 2.1 using C and store the data into files. Import this data so that Python is used only for plotting.

3.4 Write a function for computing the common difference d given t_0, t_{n-1} and n.
Solution:

```c
#include <stdio.h>

float comm_diff(float,float,int);
int main(void)
{
FILE *fp;
float t_0 = -2.0, t_k = 8.0, d,t_n;
int k = 20, n;

//Common difference
d = comm_diff(t_0,t_k,k);
fp = fopen("ap.dat","w");
for(n = 0; n < k; n++)
{
t_n = t_0+n*d;
printf("%f\n",t_n);
fprintf(fp,"%f\n",t_n);
}
fclose(fp);
return 0;
}
float comm_diff(float first,float last,int n)
{
float d;
d = (last-first)/(n-1);
return d;
}
```

4 Python programming exercises

4.1 Find $\mathbf{A}^{-1}$.
Solution: The *inverse* of $\mathbf{A}$ is obtained as

$$\mathbf{A}^{-1} = \frac{1}{\det \mathbf{A}} \begin{pmatrix} -1 & -1 \\ -3 & 1 \end{pmatrix} \tag{11}$$

$$= \frac{1}{-4} \begin{pmatrix} -1 & -1 \\ -3 & 1 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 1 \\ 3 & -1 \end{pmatrix} \tag{12}$$

4.2 Write your own function for calculating $\mathbf{A}^{-1}$

4.3 Let

$$\mathbf{c} = \begin{pmatrix} 8 \\ 12 \end{pmatrix} \tag{13}$$

Find $\mathbf{A}^{-1}\mathbf{b}$
Solution: From (11) and (13),

$$\mathbf{A}^{-1}\mathbf{b} = \frac{1}{4} \begin{pmatrix} 1 & 1 \\ 3 & -1 \end{pmatrix} \begin{pmatrix} 8 \\ 12 \end{pmatrix} \tag{14}$$

$$= \frac{1}{4} \begin{pmatrix} 1 \times 8 + 1 \times 12 \\ 3 \times 8 - 1 \times 12 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 20 \\ 12 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \end{pmatrix} \tag{15}$$

4.4 Verify that (15) is a solution of (1).

4.5 Write a program to find the solution of (1).
Solution: The following program finds the solution

```python
#Code by GVV Sharma
#March 14, 2019
#released under GNU GPL
import numpy as np

a1 = 1
a2 = 1
b1 = 3
b2 = -1
c1 = 8
c2 = 12

A = np.array(([a1,a2],[b1,b2]))
c = np.array([c1,c2])
Ainv = np.linalg.inv(A)
x = np.matmul(Ainv,c)

print(x)
```

4.6 Write your own program for **np.matmul**.

5 C PROGRAMMING EXERCISES

5.1 A geometric sequence is defined as

$$t_{n-1} = t_0 r^{n-1} \tag{16}$$

Write a function for generating the nth term of a geometric sequence from t_0, r and n.

5.2 Write a function to calculate simple interest and amount.

5.3 Write a function to calculate compound interest and amount.

5.4 Write a function to calculate the circumference of a circle.

5.5 Write a function to calculate the area of a circle.

5.6 Write a program to find the sum of the first n terms of an arithmetic sequence.

5.7 Write a program to find the sum of the first n terms of a geometric sequence.

5.8 Write a program to find $\mathbf{A}^{-1}$ and print it.

Solution:

```c
#include <stdio.h>
#include <stdlib.h>

//This program shows how to use pointers as 2-D arrays

//Function declaration
double **createMat(int m,int n);
void readMat(int m,int n,double **p);
void print(int m,int n,double **p);
double detMat(double **p);
double **invMat(double **p);


//End function declaration

int main() //main function begins
{

//Defining the variables
int m,n;//integers
double **A,**A_inv,det;

m = 2;
n = 2;
printf("Enter the values of the matrix\n");
A = createMat(m,n);//creating the matrix
det = detMat(A);
readMat(m,n,A);//reading values into the matrix a
A_inv = invMat(A);
print(m,n,A_inv);//printing the matrix a

return 0;
}

double **invMat(double **p)
{
double **q, det;
det = detMat(p);
q = createMat(2,2);
q[0][0] = p[1][1]/det;
q[0][1] = -p[0][1]/det;
q[1][0] = -p[1][0]/det;
q[1][1] = p[0][0]/det;
return q;
}

double detMat(double **A)
{
double det;
det = A[0][0]*A[1][1]-A[0][1]*A[1][0];
return det;
}

//Defining the function for matrix creation
double **createMat(int m,int n)
{
 int i;
 double **a;

 //Allocate memory to the pointer
a = (double **)malloc(m * sizeof( *a));
    for (i=0; i<m; i++)
         a[i] = (double *)malloc(n * sizeof( *a[i]));

 return a;
}
//End function for matrix creation


//Defining the function for reading matrix
void readMat(int m,int n,double **p)
{
 int i,j;
 for(i=0;i<m;i++)
 {
```

```
   for(j=0;j<n;j++)
   {
    scanf("%lf",&p[i][j]);
   }
  }
}
//End function for reading matrix



//Defining the function for printing
void print(int m,int n,double **p)
{
 int i,j;

 for(i=0;i<m;i++)
 {
   for(j=0;j<n;j++)
   printf("%lf ",p[i][j]);
 printf("\n");
 }
}
```

5.9 Write a program to find $\mathbf{A}^{-1}\mathbf{c}$.