3D Algebra Presentation

Mansi Arpit Nanavati
EE19BTECH11036

September 13, 2019

Outline

1. Problem
2. Solution
3. C code
   - Functions used
   - Code
   - Result
4. Plot

Problem

Given

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^T \tag{2.1}$$

$$\mathbf{B} = \begin{pmatrix} 0 & 3 & 4 \end{pmatrix}^T \tag{2.2}$$

and **P** such that

$$\mathbf{BP} \parallel \mathbf{OA} \tag{2.3}$$

$$\mathbf{P}^T\mathbf{A} = 0 \tag{2.4}$$

where **O** is the origin, find

$$(\mathbf{B} - \mathbf{P}) \times \mathbf{P} \tag{2.5}$$

Solution

Since $\mathbf{BP} \parallel \mathbf{OA}$,

$$\mathbf{BP} = k\mathbf{OA} \tag{3.1}$$

where k is any arbitrary constant.

$$\implies \mathbf{P} = \mathbf{B} + k\mathbf{A} \tag{3.2}$$

$$\mathbf{P}^T\mathbf{A} = 0 \tag{3.3}$$

$$\implies (\mathbf{B} + k\mathbf{A})^T\mathbf{A} = 0 \tag{3.4}$$

$$\implies k = -3/2 \tag{3.5}$$

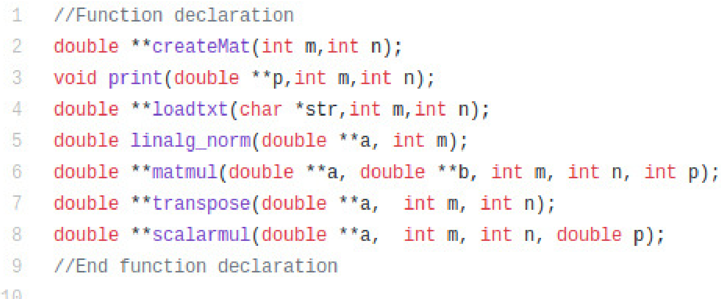
Then

$$(\mathbf{B} - \mathbf{P}) \times \mathbf{P} \tag{3.6}$$

$$= k\mathbf{B} \times \mathbf{A} = \begin{pmatrix} 6 \\ -6 \\ 4.5 \end{pmatrix} \tag{3.7}$$

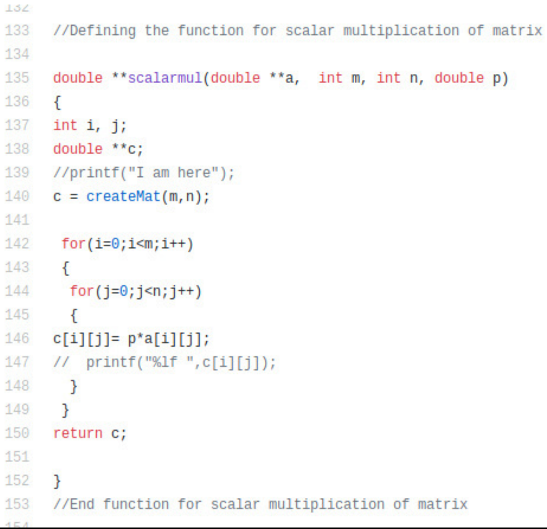
Functions

Following are the functions defined in the coeffs.h

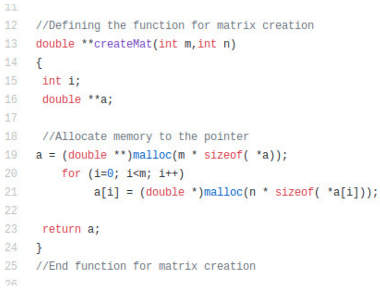


```c
//Function declaration
double **createMat(int m,int n);
void print(double **p,int m,int n);
double **loadtxt(char *str,int m,int n);
double linalg_norm(double **a, int m);
double **matmul(double **a, double **b, int m, int n, int p);
double **transpose(double **a,  int m, int n);
double **scalarmul(double **a,  int m, int n, double p);
//End function declaration
```

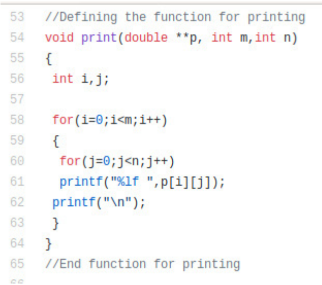
New Function



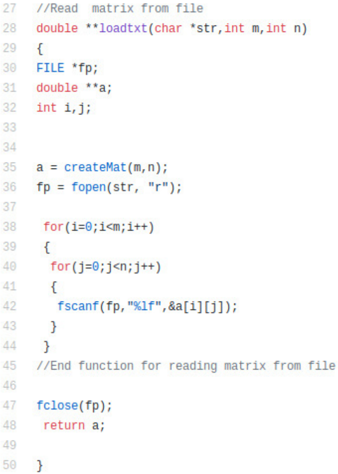
```
//Defining the function for scalar multiplication of matrix

double **scalarmul(double **a,  int m, int n, double p)
{
int i, j;
double **c;
//printf("I am here");
c = createMat(m,n);

 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
c[i][j]= p*a[i][j];
//  printf("%lf ",c[i][j]);
  }
 }
return c;

}
//End function for scalar multiplication of matrix
```



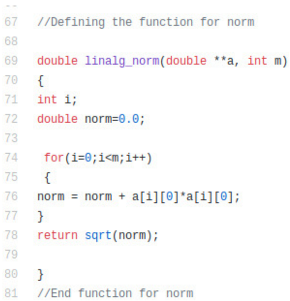
```
//Defining the function for matrix creation
double **createMat(int m,int n)
{
 int i;
 double **a;

 //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
```



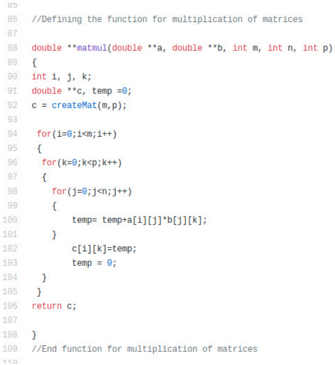
```
//Defining the function for printing
void print(double **p, int m,int n)
{
 int i,j;

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



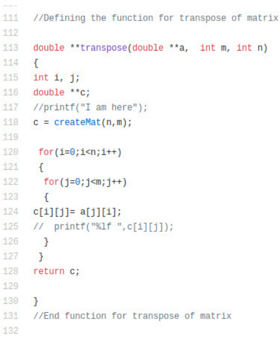
```
//Read  matrix from file
double **loadtxt(char *str,int m,int n)
{
FILE *fp;
double **a;
int i,j;


a = createMat(m,n);
fp = fopen(str, "r");

 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   fscanf(fp,"%lf",&a[i][j]);
  }
 }
//End function for reading matrix from file

fclose(fp);
 return a;

}
```



```
//Defining the function for norm

double linalg_norm(double **a, int m)
{
int i;
double norm=0.0;

 for(i=0;i<m;i++)
 {
norm = norm + a[i][0]*a[i][0];
}
return sqrt(norm);

}
//End function for norm
```

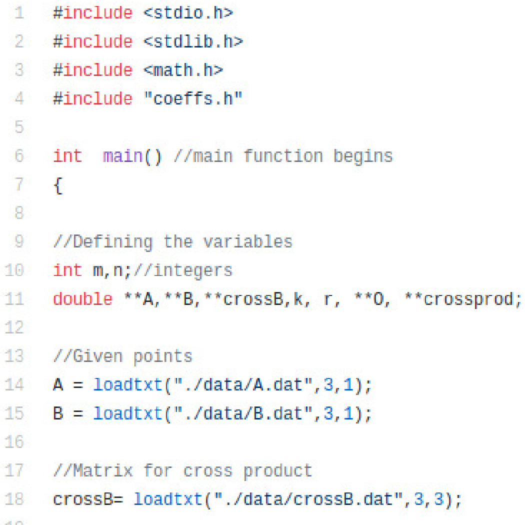


```
//Defining the function for multiplication of matrices

double **matmul(double **a, double **b, int m, int n, int p)
{
int i, j, k;
double **c, temp =0;
c = createMat(m,p);

 for(i=0;i<m;i++)
 {
  for(k=0;k<p;k++)
  {
    for(j=0;j<n;j++)
    {
        temp= temp+a[i][j]*b[j][k];
    }
        c[i][k]=temp;
        temp = 0;
  }
 }
return c;

}
//End function for multiplication of matrices
```

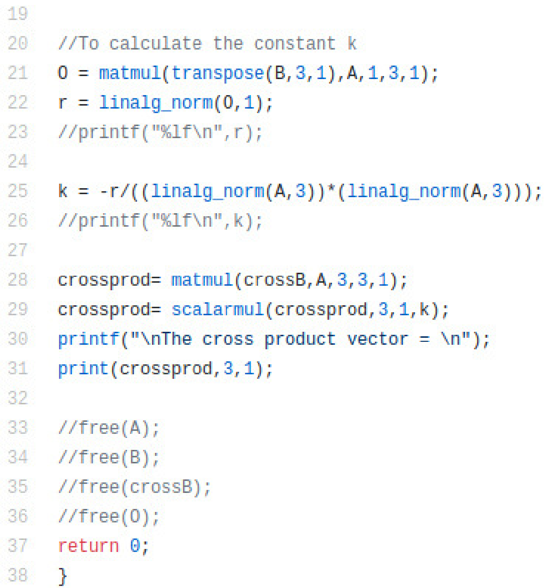


```
//Defining the function for transpose of matrix

double **transpose(double **a,  int m, int n)
{
int i, j;
double **c;
//printf("I am here");
c = createMat(n,m);

 for(i=0;i<n;i++)
 {
  for(j=0;j<m;j++)
  {
c[i][j]= a[j][i];
//  printf("%lf ",c[i][j]);
  }
 }
return c;

}
//End function for transpose of matrix
```

Code

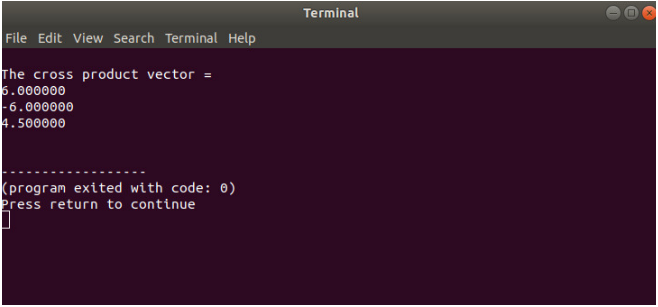


```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "coeffs.h"

int  main() //main function begins
{

//Defining the variables
int m,n;//integers
double **A,**B,**crossB,k, r, **O, **crossprod;

//Given points
A = loadtxt("./data/A.dat",3,1);
B = loadtxt("./data/B.dat",3,1);

//Matrix for cross product
crossB= loadtxt("./data/crossB.dat",3,3);
```



```
//To calculate the constant k
O = matmul(transpose(B,3,1),A,1,3,1);
r = linalg_norm(O,1);
//printf("%lf\n",r);

k = -r/((linalg_norm(A,3))*(linalg_norm(A,3)));
//printf("%lf\n",k);

crossprod= matmul(crossB,A,3,3,1);
crossprod= scalarmul(crossprod,3,1,k);
printf("\nThe cross product vector = \n");
print(crossprod,3,1);

//free(A);
//free(B);
//free(crossB);
//free(O);
return 0;
}
```

Result in Terminal



```
The cross product vector =
6.000000
-6.000000
4.500000


------------------
(program exited with code: 0)
Press return to continue
```

Plot

The code in

https://github.com/glitched−shadeslayer/Python−to−C−3D

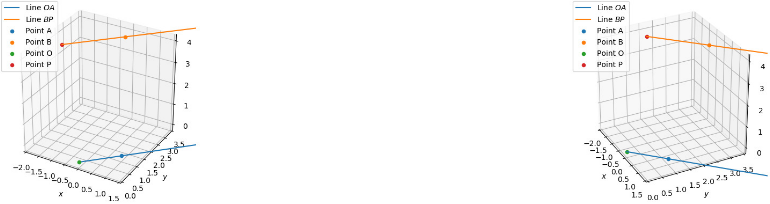


Figure: The lines **OA** and **BP**

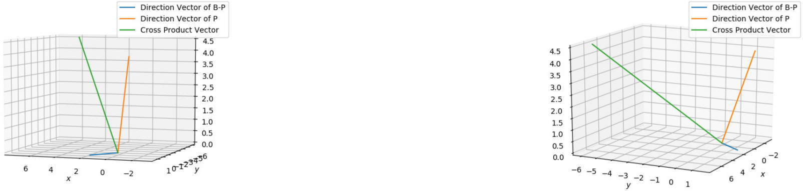


Figure: The cross product of **B**-**P** and **P**