# Homework 1

Aasim Zahoor

September 10, 2020

## 1 Problems

**Link** `https://github.com/AasimZahoor/Comp_methods.git`

### Problem 1

In the code for Problem 1 I have defined five functions. They are:

- **midpoint(f,a,b,n)**
- **trapezoid(f,a,b,n)**
- **simpson(f,a,b,n)**

  These functions have the same arguments and thet are f= 'the function being integrated', $[a,b]$='The range of integration', n='The number of steps'. All of them need to be inputted while calling the function.

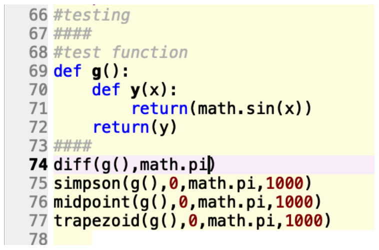
- **diff(f,a,h=0.000001)**

  This function differentiates the function f at point a. h represents the step size and has a default value of 0.000001

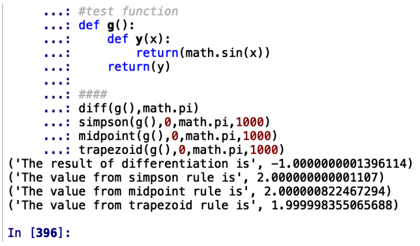
- **g()**

  This is a test function and can be used to test these function. You would need to make changes in the code to test different function.

*Testing the functions using sine function(the figure on the left), the results of the test(the figure on the right)*



```
#testing
####
#test function
def g():
    def y(x):
        return(math.sin(x))
    return(y)
####
diff(g(),math.pi)
simpson(g(),0,math.pi,1000)
midpoint(g(),0,math.pi,1000)
trapezoid(g(),0,math.pi,1000)
```



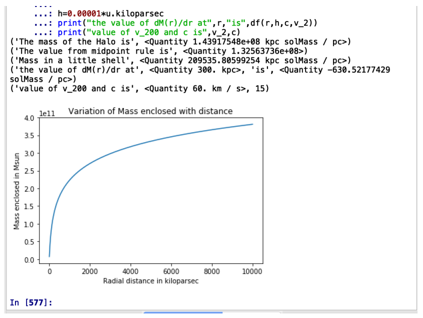
```
   ...: #test function
   ...: def g():
   ...:     def y(x):
   ...:         return(math.sin(x))
   ...:     return(y)
   ...:
   ...: ####
   ...: diff(g(),math.pi)
   ...: simpson(g(),0,math.pi,1000)
   ...: midpoint(g(),0,math.pi,1000)
   ...: trapezoid(g(),0,math.pi,1000)
('The result of differentiation is', -1.0000000001396114)
('The value from simpson rule is', 2.000000000001107)
('The value from midpoint rule is', 2.000000822467294)
('The value from trapezoid rule is', 1.999998355065688)

In [396]:
```

## Problem 2

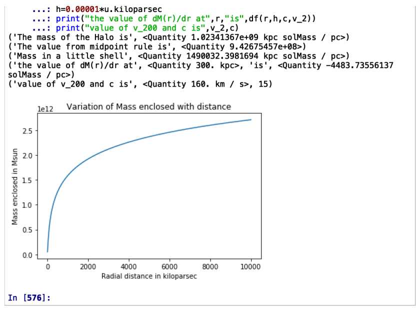
The problem asked us to do four things for a given value of c and V200. In my code(which hopefully has enough comments that it is easier to understand) I have defined three functions, two to give the Mass enclosed as output (one function gives a value of mass enclosed at r and one gives a function in terms of r, which I used to test my approximation I used in finding the mass in a shell) and one to give Vc. I calculated the mass enclosed in a halo by taking r=300kpc, the mass in a little shell was also found at r=300kpc with a thickness of 1 kpc. I also calculated the derivative at r=300kpc and has a h =0.00001kiloparsec (small increment which we use in the definition). Then I changed V200 values [60,160,260,360] while keeping c=15. Then I changed c values [5,15,25,35] while keeping V200=160km/s. From the figure it appears the curve flattens which is expected as the density decreases as we move away and hence the mass becomes constant. Also, the product of differentiation at r=300kpc is negative showing that mass of shells is decreasing at that point , making the previous observation more solid.

I calculated mass of the shell using midpoint too and it agrees reasonably well with my approximation.



```
    ...:
    ...: h=0.00001*u.kiloparsec
    ...: print("the value of dM(r)/dr at",r,"is",df(r,h,c,v_2))
    ...: print("value of v_200 and c is",v_2,c)
('The mass of the Halo is', <Quantity 1.43917548e+08 kpc solMass / pc>)
('The value from midpoint rule is', <Quantity 1.32563736e+08>)
('Mass in a little shell', <Quantity 209535.80599254 kpc solMass / pc>)
('the value of dM(r)/dr at', <Quantity 300. kpc>, 'is', <Quantity -630.52177429
solMass / pc>)
('value of v_200 and c is', <Quantity 60. km / s>, 15)
```

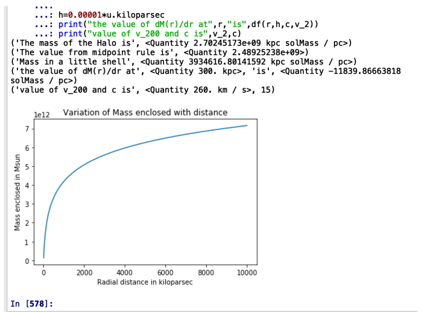
```
In [577]:
```



```
    ...: h=0.00001*u.kiloparsec
    ...: print("the value of dM(r)/dr at",r,"is",df(r,h,c,v_2))
    ...: print("value of v_200 and c is",v_2,c)
('The mass of the Halo is', <Quantity 1.02341367e+09 kpc solMass / pc>)
('The value from midpoint rule is', <Quantity 9.42675457e+08>)
('Mass in a little shell', <Quantity 1490032.3981694 kpc solMass / pc>)
('the value of dM(r)/dr at', <Quantity 300. kpc>, 'is', <Quantity -4483.73556137
solMass / pc>)
('value of v_200 and c is', <Quantity 160. km / s>, 15)
```

Variation of Mass enclosed with distance
1e12
2.5
2.0
1.5
1.0
0.5
0.0
Mass enclosed in Msun
0
2000
4000
6000
8000
10000
Radial distance in kiloparsec

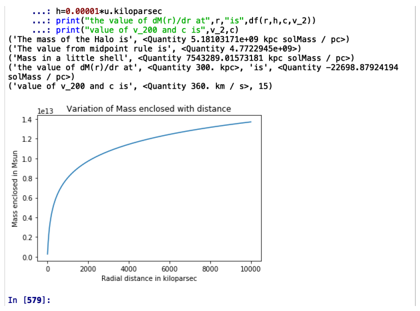
```
In [576]:
```



```
    ...:
    ...: h=0.00001*u.kiloparsec
    ...: print("the value of dM(r)/dr at",r,"is",df(r,h,c,v_2))
    ...: print("value of v_200 and c is",v_2,c)
('The mass of the Halo is', <Quantity 2.70245173e+09 kpc solMass / pc>)
('The value from midpoint rule is', <Quantity 2.48925238e+09>)
('Mass in a little shell', <Quantity 3934616.80141592 kpc solMass / pc>)
('the value of dM(r)/dr at', <Quantity 300. kpc>, 'is', <Quantity -11839.86663818
solMass / pc>)
('value of v_200 and c is', <Quantity 260. km / s>, 15)
```

Variation of Mass enclosed with distance
1e12
7
6
5
4
3
2
1
0
Mass enclosed in Msun
0
2000
4000
6000
8000
10000
Radial distance in kiloparsec

```
In [578]:
```

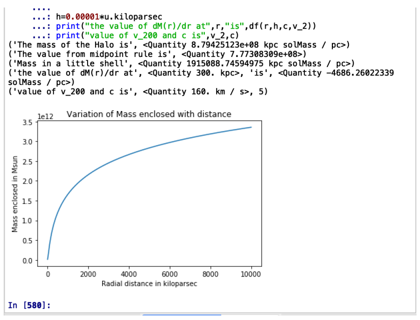


```
    ...: h=0.00001*u.kiloparsec
    ...: print("the value of dM(r)/dr at",r,"is",df(r,h,c,v_2))
    ...: print("value of v_200 and c is",v_2,c)
('The mass of the Halo is', <Quantity 5.18103171e+09 kpc solMass / pc>)
('The value from midpoint rule is', <Quantity 4.7722945e+09>)
('Mass in a little shell', <Quantity 7543289.01573181 kpc solMass / pc>)
('the value of dM(r)/dr at', <Quantity 300. kpc>, 'is', <Quantity -22698.87924194
solMass / pc>)
('value of v_200 and c is', <Quantity 360. km / s>, 15)
```

```
In [579]:
```

*These figures show output when I kept c fixed to 15 and changed V200.*

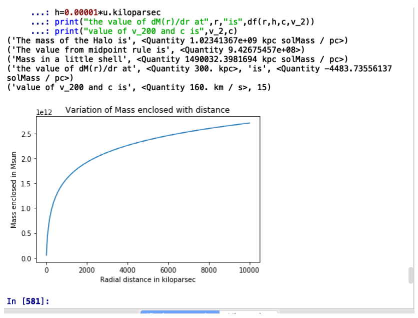
As I increased V200 while keeping c constant, value of mass in a halo, mass in the shell increased but the dM(r)/dr became more negative.



```
   ....
   ...: h=0.00001*u.kiloparsec
   ...: print("the value of dM(r)/dr at",r,"is",df(r,h,c,v_2))
   ...: print("value of v_200 and c is",v_2,c)
('The mass of the Halo is', <Quantity 8.79425123e+08 kpc solMass / pc>)
('The value from midpoint rule is', <Quantity 7.77308309e+08>)
('Mass in a little shell', <Quantity 1915088.74594975 kpc solMass / pc>)
('the value of dM(r)/dr at', <Quantity 300. kpc>, 'is', <Quantity -4686.26022339
solMass / pc>)
('value of v_200 and c is', <Quantity 160. km / s>, 5)
```

Variation of Mass enclosed with distance
1e12
Mass enclosed in Msun
Radial distance in kiloparsec

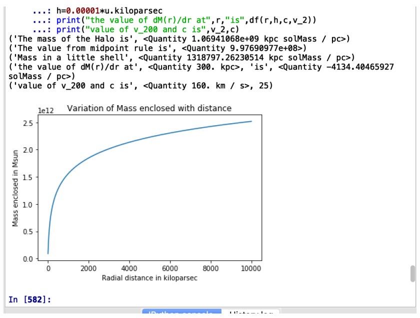
```
In [580]:
```



```
   ...: h=0.00001*u.kiloparsec
   ...: print("the value of dM(r)/dr at",r,"is",df(r,h,c,v_2))
   ...: print("value of v_200 and c is",v_2,c)
('The mass of the Halo is', <Quantity 1.02341367e+09 kpc solMass / pc>)
('The value from midpoint rule is', <Quantity 9.42675457e+08>)
('Mass in a little shell', <Quantity 1490032.3981694 kpc solMass / pc>)
('the value of dM(r)/dr at', <Quantity 300. kpc>, 'is', <Quantity -4483.73556137
solMass / pc>)
('value of v_200 and c is', <Quantity 160. km / s>, 15)
```

Variation of Mass enclosed with distance
1e12
Mass enclosed in Msun
Radial distance in kiloparsec

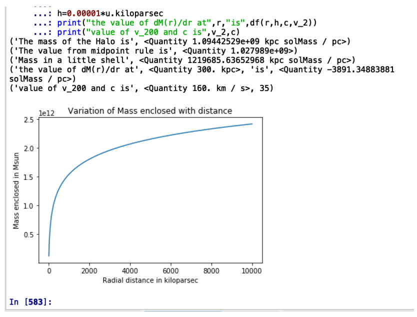
```
In [581]:
```



```
   ...: h=0.00001*u.kiloparsec
   ...: print("the value of dM(r)/dr at",r,"is",df(r,h,c,v_2))
   ...: print("value of v_200 and c is",v_2,c)
('The mass of the Halo is', <Quantity 1.06941068e+09 kpc solMass / pc>)
('The value from midpoint rule is', <Quantity 9.97690977e+08>)
('Mass in a little shell', <Quantity 1318797.26230514 kpc solMass / pc>)
('the value of dM(r)/dr at', <Quantity 300. kpc>, 'is', <Quantity -4134.40465927
solMass / pc>)
('value of v_200 and c is', <Quantity 160. km / s>, 25)
```

Variation of Mass enclosed with distance
1e12
Mass enclosed in Msun
Radial distance in kiloparsec

```
In [582]:
```



```
   ....
   ...: h=0.00001*u.kiloparsec
   ...: print("the value of dM(r)/dr at",r,"is",df(r,h,c,v_2))
   ...: print("value of v_200 and c is",v_2,c)
('The mass of the Halo is', <Quantity 1.09442529e+09 kpc solMass / pc>)
('The value from midpoint rule is', <Quantity 1.027989e+09>)
('Mass in a little shell', <Quantity 1219685.63652968 kpc solMass / pc>)
('the value of dM(r)/dr at', <Quantity 300. kpc>, 'is', <Quantity -3891.34883881
solMass / pc>)
('value of v_200 and c is', <Quantity 160. km / s>, 35)
```

Variation of Mass enclosed with distance
1e12
Mass enclosed in Msun
Radial distance in kiloparsec

```
In [583]:
```

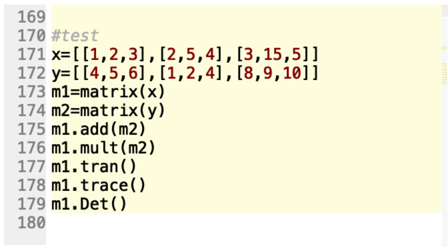
*These figures show output when I kept v200 fixed to 160 and changed c.*

As I increased c while keeping V200 constant, everything increased (dM(r)/dr becomes lesser negative).

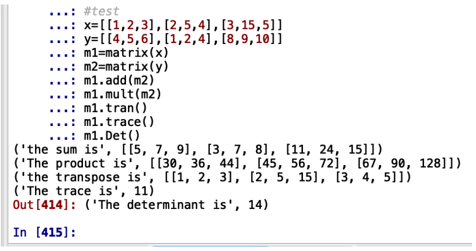
## Problem 3

In this problem I have made a matrix class in which I have defined eight instances.

- **init** It has self and the matrix array as the argument. Here I defined 3 instance attributes, self.g gives back the array and helps me handle the array in a better way.
- **add(self,other)** It takes self and the other(matrix) as the argument. I have included a check to see if addition is possible
- **mult(self,other)** It takes self and the other(matrix) as the argument. I have included a check to see if multiplication is possible
- **tran(self)**It takes self as argument and gives out Transpose as output.
- **trace(self)**It takes self as argument and gives out Trace as output.
- **Det(self)**It takes self as argument and gives out Determinant as output.



```
#test
x=[[1,2,3],[2,5,4],[3,15,5]]
y=[[4,5,6],[1,2,4],[8,9,10]]
m1=matrix(x)
m2=matrix(y)
m1.add(m2)
m1.mult(m2)
m1.tran()
m1.trace()
m1.Det()
```



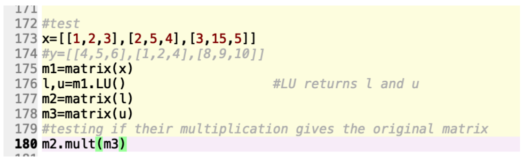
```
    ...: #test
    ...: x=[[1,2,3],[2,5,4],[3,15,5]]
    ...: y=[[4,5,6],[1,2,4],[8,9,10]]
    ...: m1=matrix(x)
    ...: m2=matrix(y)
    ...: m1.add(m2)
    ...: m1.mult(m2)
    ...: m1.tran()
    ...: m1.trace()
    ...: m1.Det()
('the sum is', [[5, 7, 9], [3, 7, 8], [11, 24, 15]])
('The product is', [[30, 36, 44], [45, 56, 72], [67, 90, 128]])
('the transpose is', [[1, 2, 3], [2, 5, 15], [3, 4, 5]])
('The trace is', 11)
Out[414]: ('The determinant is', 14)

In [415]:
```

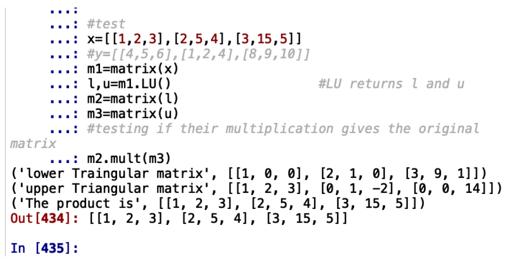
*Testing the instances using a matrix(the figure on the left), the results of the test(the figure on the right)*

- **LU(self)**Takes in self as argument and gives out lower and upper triangular matrix (in that order). In the figure I multiplies the L and U matrix and got the original matrix back.

*Testing the LU Instance using a matrix*



```
#test
x=[[1,2,3],[2,5,4],[3,15,5]]
#y=[[4,5,6],[1,2,4],[8,9,10]]
m1=matrix(x)
l,u=m1.LU()                    #LU returns l and u
m2=matrix(l)
m3=matrix(u)
#testing if their multiplication gives the original matrix
m2.mult(m3)
```

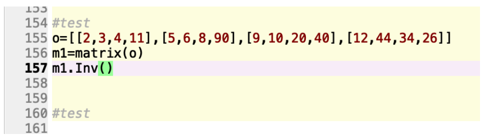


```
    ...:
    ...: #test
    ...: x=[[1,2,3],[2,5,4],[3,15,5]]
    ...: #y=[[4,5,6],[1,2,4],[8,9,10]]
    ...: m1=matrix(x)
    ...: l,u=m1.LU()                    #LU returns l and u
    ...: m2=matrix(l)
    ...: m3=matrix(u)
    ...: #testing if their multiplication gives the original
matrix
    ...: m2.mult(m3)
('lower Traingular matrix', [[1, 0, 0], [2, 1, 0], [3, 9, 1]])
('upper Triangular matrix', [[1, 2, 3], [0, 1, -2], [0, 0, 14]])
('The product is', [[1, 2, 3], [2, 5, 4], [3, 15, 5]])
Out[434]: [[1, 2, 3], [2, 5, 4], [3, 15, 5]]

In [435]:
```

*Results of the test*

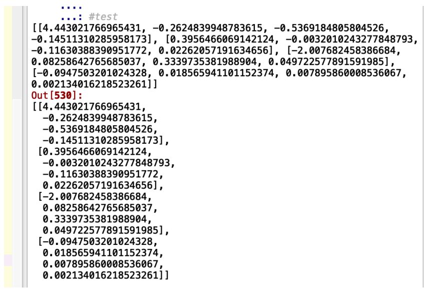
- **Inv(self)**Takes self as the argument and gives out the inverse.



```
#test
o=[[2,3,4,11],[5,6,8,90],[9,10,20,40],[12,44,34,26]]
m1=matrix(o)
m1.Inv()


#test
```

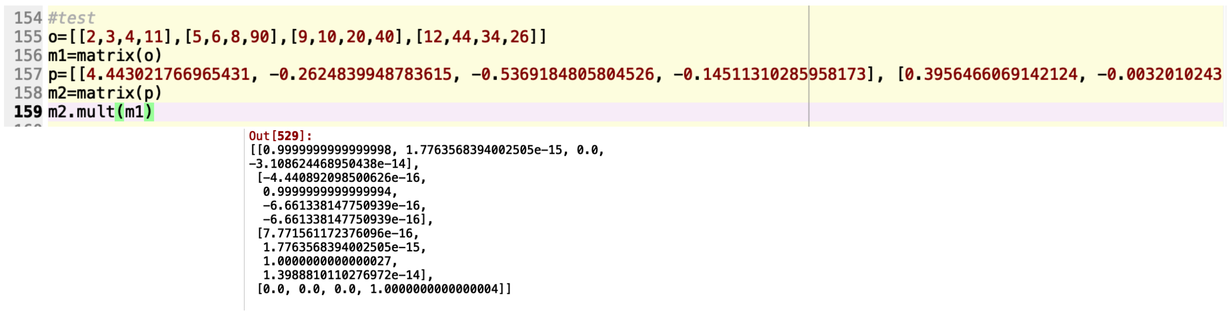
*Testing the Inv Instance using a matrix*



```
    ....
    ...: #test
[[4.443021766965431, -0.2624839948783615, -0.5369184805804526,
-0.14511310285958173], [0.3956466069142124, -0.0032010243277848793,
-0.11630388390951772, 0.02262057191634656], [-2.007682458386684,
0.08258642765685037, 0.3339735381988904, 0.04972257789159198],
[-0.0947503201024328, 0.018565941101152374, 0.007895860008536067,
0.002134016218523261]]
Out[530]:
[[4.443021766965431,
  -0.2624839948783615,
  -0.5369184805804526,
  -0.14511310285958173],
 [0.3956466069142124,
  -0.0032010243277848793,
  -0.11630388390951772,
  0.02262057191634656],
 [-2.007682458386684,
  0.08258642765685037,
  0.3339735381988904,
  0.04972257789159198],
 [-0.0947503201024328,
  0.018565941101152374,
  0.007895860008536067,
  0.002134016218523261]]
```

*Results of the test*

*I multiplied the result of the inverse with the original and got the identity matrix*



```
#test
o=[[2,3,4,11],[5,6,8,90],[9,10,20,40],[12,44,34,26]]
m1=matrix(o)
p=[[4.443021766965431, -0.2624839948783615, -0.5369184805804526, -0.14511310285958173], [0.3956466069142124, -0.0032010243
m2=matrix(p)
m2.mult(m1)
```

```
Out[529]:
[[0.9999999999999998, 1.7763568394002505e-15, 0.0,
-3.108624468950438e-14],
 [-4.440892098500626e-16,
  0.9999999999999994,
  -6.661338147750939e-16,
  -6.661338147750939e-16],
 [7.771561172376096e-16,
  1.7763568394002505e-15,
  1.0000000000000027,
  1.3988810110276972e-14],
 [0.0, 0.0, 0.0, 1.0000000000000004]]
```