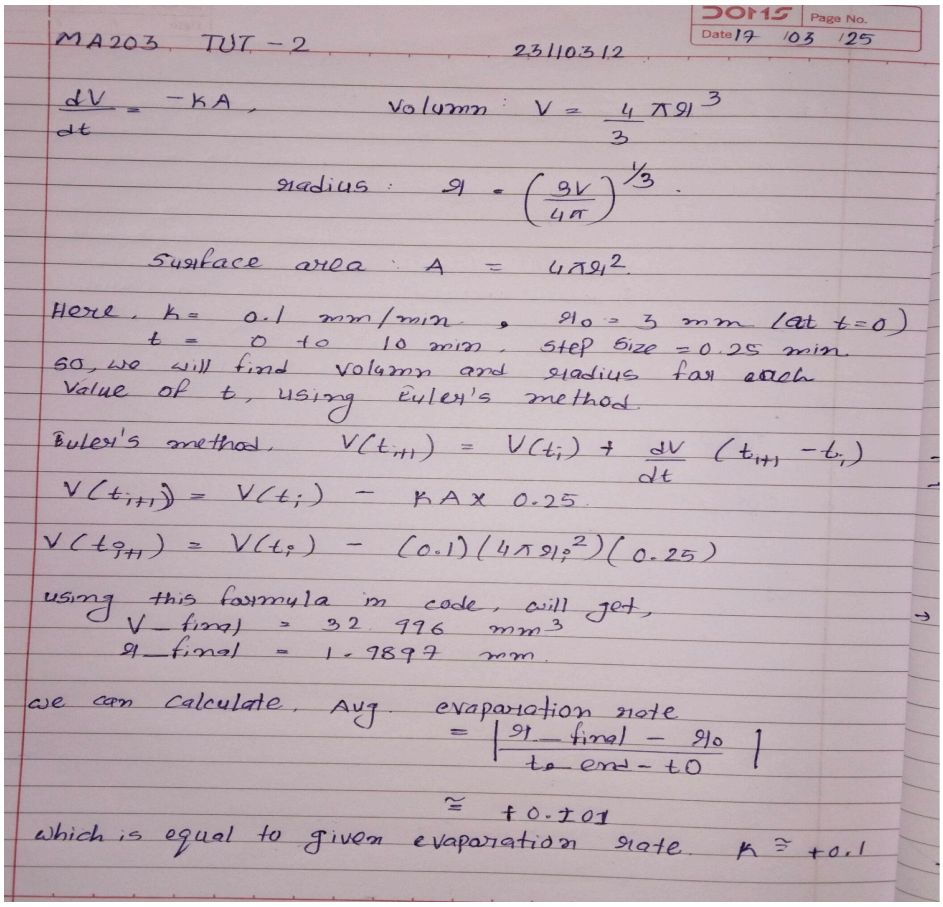
Colab link of the codes: MA203_TUT2.ipynb

Question 1:



MA203 TUT - 2 23/10/312

Page No. Date 17/03/25

$\frac{dV}{dt} = -kA$, Volume: $V = \frac{4}{3}\pi r^3$

radius: $r = \left(\frac{3V}{4\pi}\right)^{1/3}$.

Surface area: $A = 4\pi r^2$.

Here, $k = 0.1$ mm/min, $r_0 = 3$ mm (at $t=0$)
$t = 0$ to 10 min, step size = 0.25 min.
So, we will find volume and radius for each value of t, using Euler's method.

Euler's method, $V(t_{i+1}) = V(t_i) + \frac{dV}{dt}(t_{i+1} - t_i)$

$V(t_{i+1}) = V(t_i) - kA \times 0.25$

$V(t_{i+1}) = V(t_i) - (0.1)(4\pi r_i^2)(0.25)$

using this formula in code, will get,
V_final = 32.996 mm^3
r_final = 1.9897 mm.

we can calculate, Avg. evaporation rate
$= \left|\frac{r_{final} - r_0}{t_{end} - t_0}\right|$
$\approx \pm 0.101$
which is equal to given evaporation rate. $k \approx \pm 0.1$

Code:

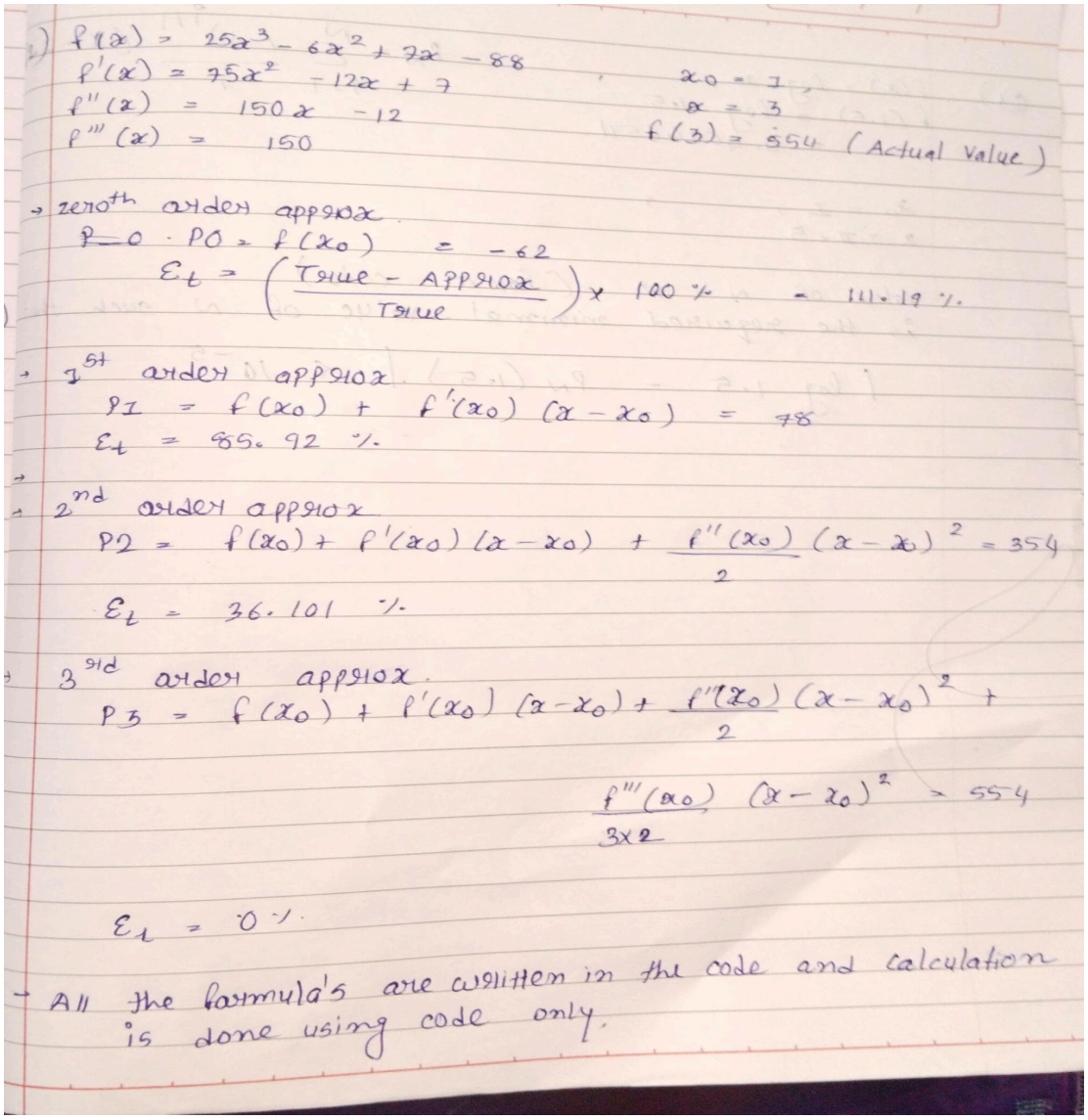
```
import numpy as np
import math

#given constants
k=0.1 #the evaporation rate (mm/min)
r0=3 #initial radius (mm)
t0 =0 #start time (min)
t_end = 10 #end time (min)
dt = 0.25 #step sizw (min)

V0= (4/3)* np.pi * r0**3

#all the time steps
```

```
pos_t=np.arange(t0, t_end+ dt, dt )
V=V0

for t in pos_t[1:]:
 r= (3 * V / (4 * np.pi)) ** (1/3) #radius for current volume
 A= 4 * np.pi * (r **2) #Surface area for given radius
 dV_dt = -k * A #Evaporation rate
 V = V+ dV_dt * dt #update the volume

#Final value of volume and radius at t_end time
V_final = V
r_final = (3 * V_final / (4 * np.pi)) ** (1/3)

avg_evaporation_rate = (r_final - r0) / (t_end - t0) #average evaporation
rate

print("Final volume: ", V_final,"mm3")
print("Final radius: ", r_final, "mm")
print("Avrage Evaporation Rate: ",abs(avg_evaporation_rate),"mm/min")
print("Given Evaporation Rate: ", k,"mm/min")
```

Output:

```
Final volume:  32.996088381706834 mm3
Final radius:  1.9897168736844184 mm
Avrage Evaporation Rate:  0.10102831263155816 mm/min
Given Evaporation Rate:  0.1 mm/min
```

Question 2:



Code:

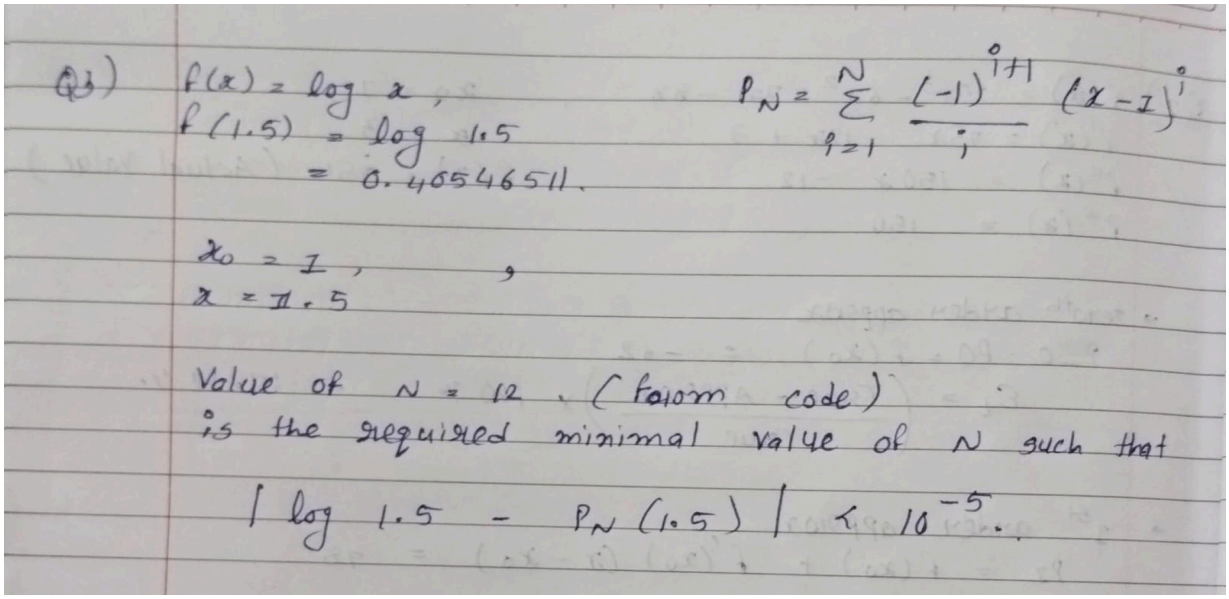
```
#given values
x0=1 #base point
x=3 #point we want to predict

f=lambda x: 25 * (x**3) - 6 * (x**2) + (7*x) - 88 #the given function
f_prime=lambda x: 75 * (x**2) - (12 * x) + 7 #first derivative
f_double_prime=lambda x: 150*x - 12 #second dericatice
f_triple_prime= lambda x: 150 #third derivative

true_val = f(3) #actual value of f at x=3
diff = x - x0 #difference between x and x0
```

```
#function to caculate relative error
def err(approx_val):
 return abs((true_val - approx_val) / true_val) * 100

#Taylor approximation
P_0=f(x0) #zeroth order approximation
err_0=err(P_0) #error related to that
P_1=P_0+f_prime(x0)*(x-x0) #first order approximation
err_1=err(P_1) #error related to that

P_2=P_1+f_double_prime(x0)*(x-x0)**2/2 #second order approximation
err_2=err(P_2) #error related to that

P_3=P_2+ (f_triple_prime(x0))*((x-x0)**3)/6 #third order approximation
err_3=err(P_3) #error related to that

#results
print("Actual value of f: ", true_val)
print("Zeroth order approximation: ", P_0)
print("First order approximation: ", P_1)
print("Second order approximation: ", P_2)
print("Third order approximation: ", P_3 ,"\n")
print("Errors" )
print("Relative error for zeroth order approximation: ", err_0, "%")
print("Relative error for first order approximation: ", err_1, "%")
print("Relative error for second order approximation: ", err_2, "%")
print("Relative error for third order approximation: ", err_3, "%")
```

Output:

```
Actual value of f:  554
Zeroth order approximation:  -62
First order approximation:  78
Second order approximation:  354.0
Third order approximation:  554.0

Errors
Relative error for zeroth order approximation:  111.1913357400722 %
Relative error for first order approximation:  85.92057761732852 %
Relative error for second order approximation:  36.101083032490976 %
Relative error for third order approximation:  0.0 %
```

Question 3:



Code:

```
import math

#value of log(1.5)
exact_val = 0.40546511

x0=1
x=1.5

#define a function to calculate the value of Nth polynomial
def approx(N,x=1.5 , x0=1):
 P_N=0
 for i in range(1,N+1):
   P_N+=((-1)**(i+1)/i)* (x-x0)**i
 return P_N

N=1
approx_val=0

#loop for getting the minimum value of N which satisfies the given
condition
while abs(exact_val - approx_val) > 1e-5:
 approx_val=approx(N)
 if abs(exact_val - approx_val) < 1e-5:
```

```
        break
 N+=1

print("Minimum value of N: ",N)
```

Output:

```
Minimum value of N:  12
```