University of Nottingham
Department of Mechanical, Materials and Manufacturing Engineering

MMME3086 (Computer Modelling Techniques)

MMME3086 Coursework submission template (NM)

(Edit and Submit as PDF file)

Student ID:	**20270217**
Student Name:	**Jakub Maslowski**
Module:	Computer Modelling Techniques
Coursework:	Numerical Methods (NM)

Task (A)

Student ID	Last digit of student ID	Resulting value of S_P
20270217	7	-160

A1

- Discretisation equation for 1st (leftmost) control volume:
- Discretisation equation for nth (rightmost) control volume:
- Discretisation equation for internal control volume:

```
%Discreisation Equation for 1st (leftmost) control volume:
A(1,1)*T(1)+A(1,2)*T(1)=B(1);
A(1,1)=(lambda/dx)-(Sp*DxB); A(1,2)=-lambda/dx; B(1)=(Sc*DxB)+qa;

%Discretisation equation for nth (rightmost) control volume:
A(n,n)*T(n)+A(n,n-1)*T(n-1)=B(n);
A(n,n)=1; A(n,n-1)=0; B(n)=Tb;

%Discretisation equation for internal control volume:
A(i,i)*T(i)+A(i,i-1)*T(i-1)+A(i,i+1)*T(i+1)=B(i)

A(i,i)=(2*lambda/dx)-(Sp*Dx); A(i,i-1)=-lambda/dx; A(i,i+1)=-lambda/dx;
B(i)=Sc*Dx;
```

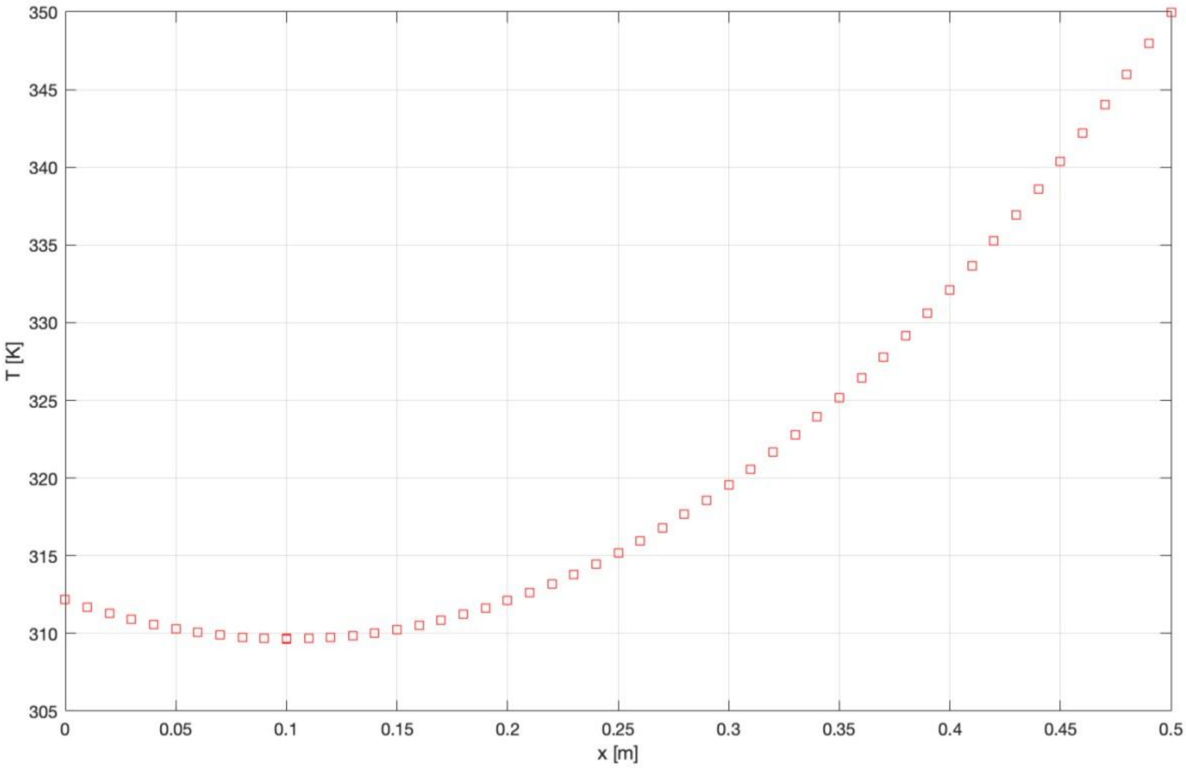
- Screen shot of matlab code for populating matrix **A** and vector **B**:

```
%Task A, Jakub Maslowski, egyjm6@nottingham.ac.uk

clear all; close all; clc; %clears workspace, figures, command window

%input parameters
L=0.5; n=51;
lambda=100;
qa=5000; Tb=350;
Sc=10; Sp=-160; %Source Term S=Sc+Sp*T

%Generate Grid
x0=linspace(0,L,n); dx=L/(n-1);   % x0:nodes position
Dx=dx; DxB=Dx/2;

%Creating Matrix
A=zeros(n,n); B=zeros(n,1); %Fill Matrix with zeros

A(1,1)=(lambda/dx)-(Sp*DxB); A(1,2)=-lambda/dx; B(1)=(Sc*DxB)+qa; % Node 1
A(n,n)=1; B(n)=Tb; % Node n

%Fill A and B%
for i=2:n-1
    A(i,i)=(2*lambda/dx)-(Sp*Dx); A(i,i-1)=-lambda/dx; A(i,i+1)=-lambda/dx;
    B(i)=Sc*Dx;
end

T=A\B; %Backslash
figure('color','w','units','Centimeters','position',[5 5 7.5 7])
plot(x0,T,'rs'); grid on; xlabel('x [m]'); ylabel('T [K]'); hold on
```

•

A2

Matlab plot of $T(x)$:

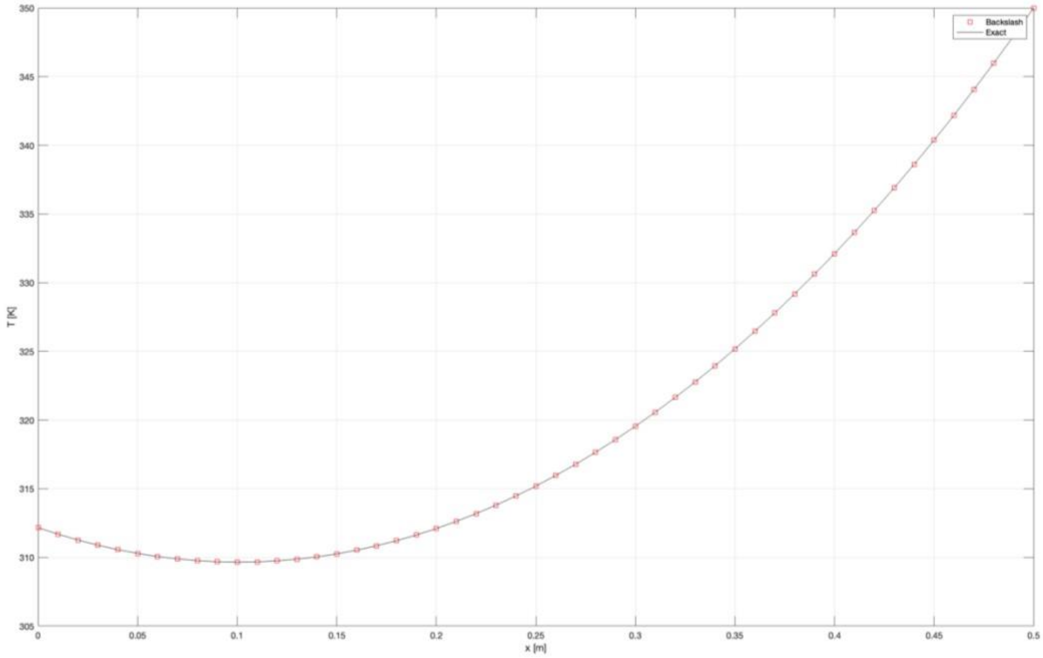


$\boldsymbol{T(x = L/2)}$	315.20

•

A3

Matlab plot of numerical and theoretical $T(x)$; you can make one single plot with A2.

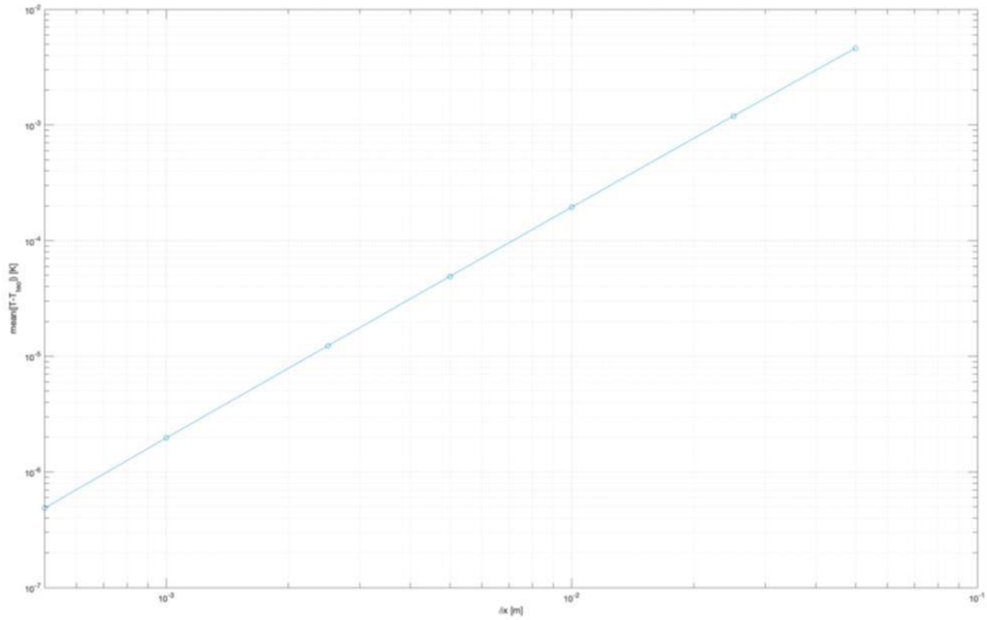


Definition of the error	Value of the error
error=mean(abs(T-Tteo'))	**1.9501*10^-4**

•

A4

Matlab log-log plot of error vs grid spacing:



Convergence order of the error	Converging towards 10^-2

• Comments (max 100 words):

The smaller the spacing between the nodes (i.e. the more nodes there are), the smaller the error becomes. This is because more nodes allow us to get as close to the actual value as possible.

Task (B)

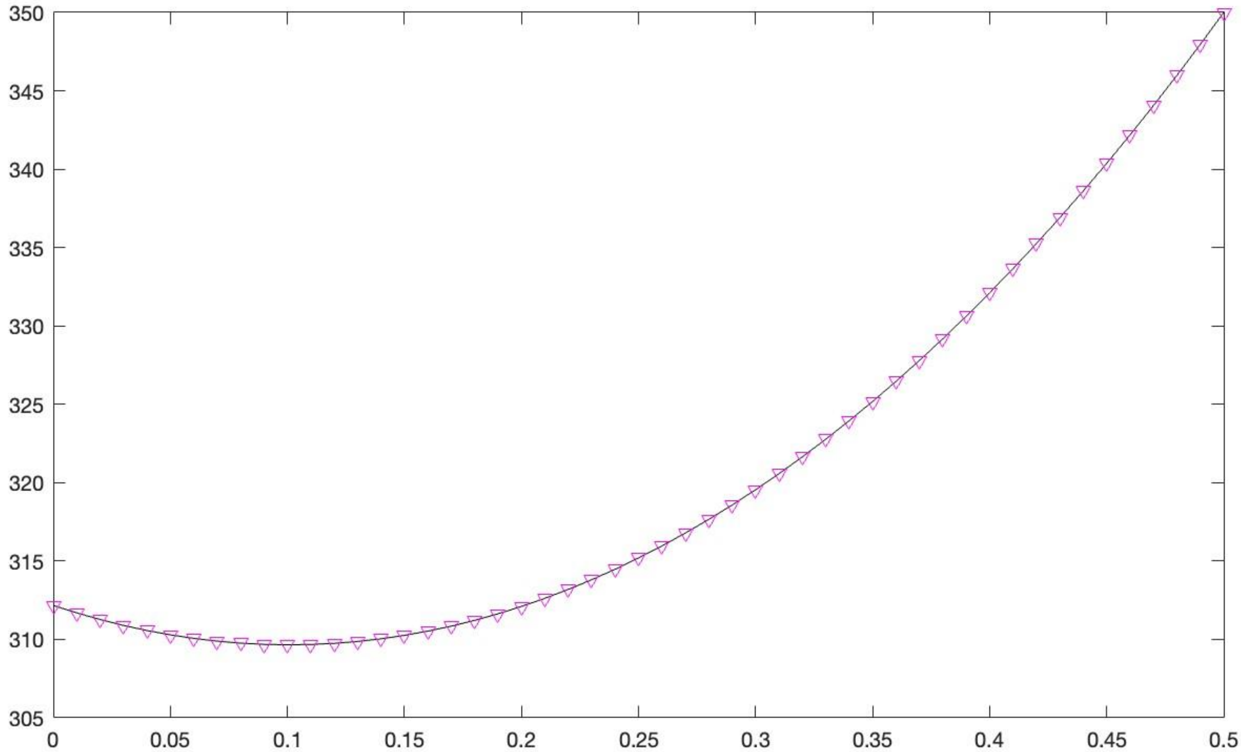
B1

• Screen shot of Matlab code for Gauss-Seidel method:

```
function [x,m,res]=GaussSeidel(x,A,B,maxit,toll)
res=sum(abs(B-A*x))/sum(abs(diag(A).*x));
m=0;

while (res>toll & m<maxit)
    m=m+1;
    for i=1:numel(x)
        x(i)=x(i)+(B(i)/A(i,i)-A(i,:)/A(i,i)*x);
    end
    res(m)=sum(abs(B-A*x))/sum(abs(diag(A).*x));
end

```

B2

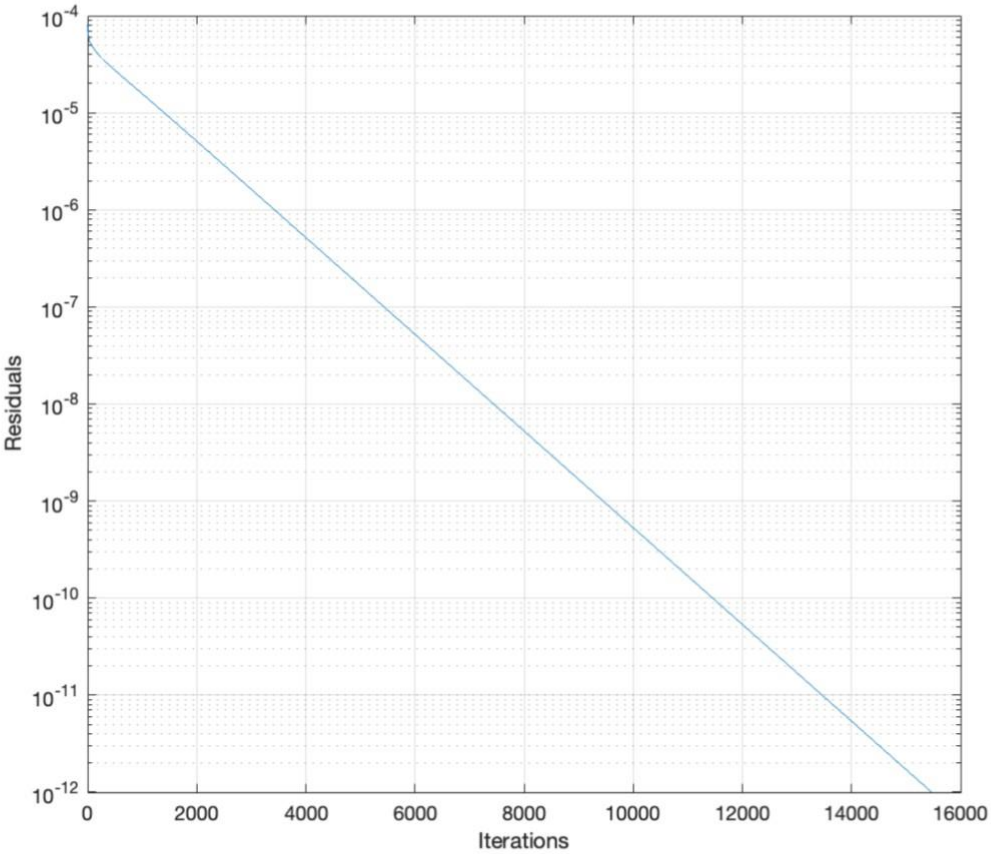
- Matlab plot of numerical and theoretical $T(x)$.



Definition of the error	Value of the error
error_gaussSeidel=mean(abs(T_gaussSeidel-Tteo))	**12.9346 (mean error of error_gaussSeidel)**

B3

- Matlab plot of residuals vs number of iterations graph in a semilogy scale



Number of iterations	Definition of residuals	Residuals at convergence
15471	residual=sum(abs(B-A*T))/sum(abs(diag(A).*T))	**10^-12**

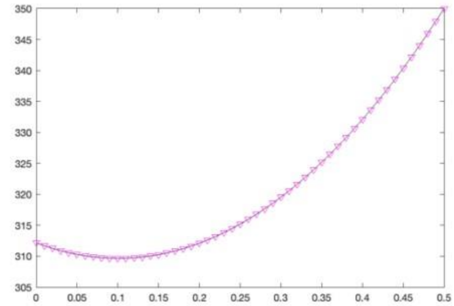
B4

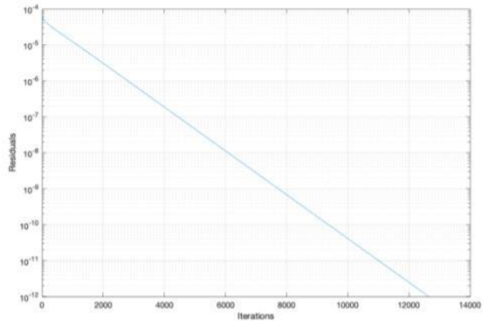
- Screen shot of matlab code for Gauss-Seidel with over-relaxation:

```
function [x,m,res]=GaussSeidel2(x,A,B,maxit,toll,omega)
res=sum(abs(B-A*x))/sum(abs(diag(A).*x));
m=0;

while (res>toll & m<maxit)
    m=m+1;
    for i=1:numel(x)
        x(i)=x(i)+omega*(B(i)/A(i,i)-A(i,:)/A(i,i)*x);
    end
    res(m)=sum(abs(B-A*x))/sum(abs(diag(A).*x));
end
```

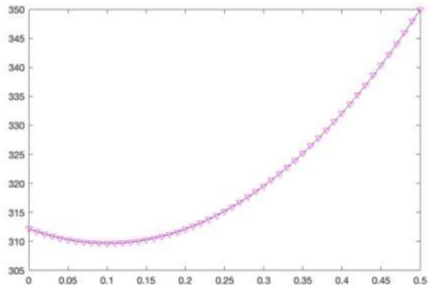
- Matlab plot of solution and residuals vs number of iterations graph for $\omega = 1.1, 1.2, \ldots, 2$

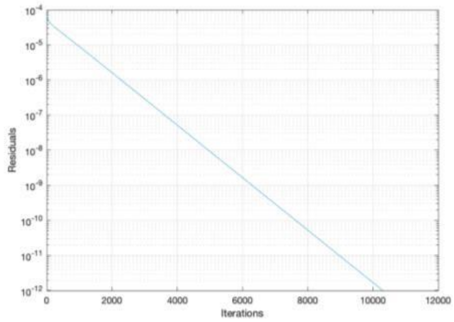
Omega=1.1



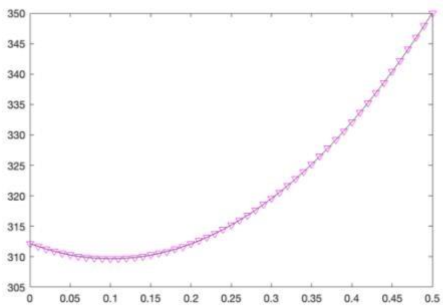


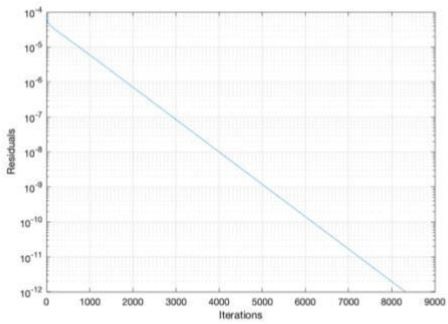
Omega=1.2



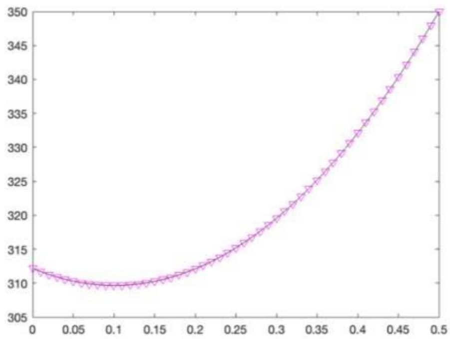


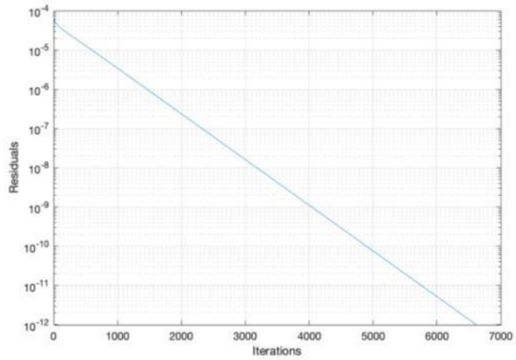
Omega=1.3



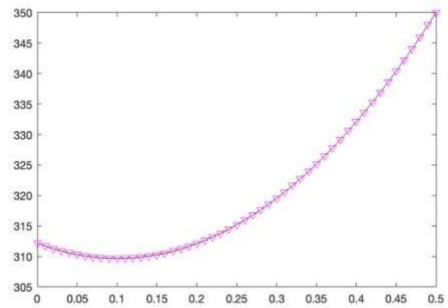


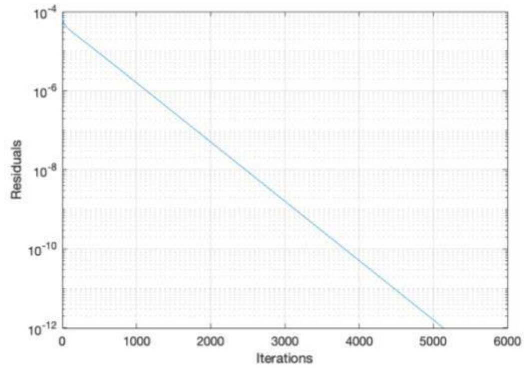
Omega=1.4



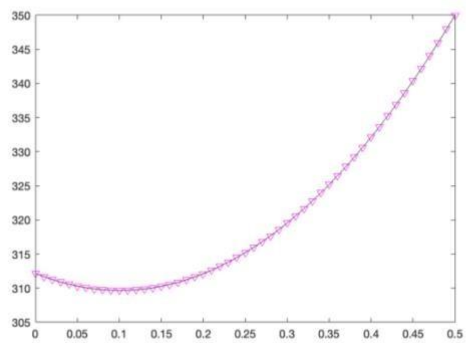


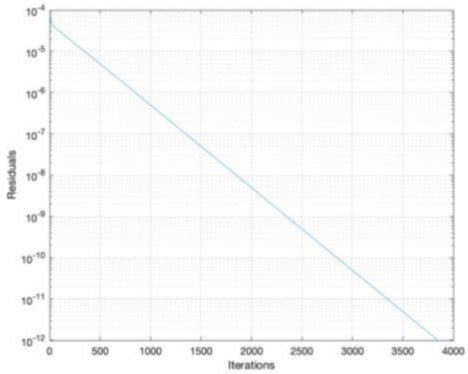
Omega=1.5



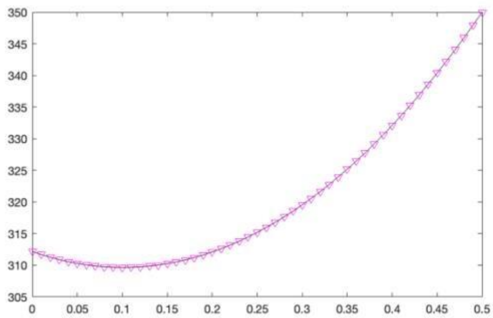


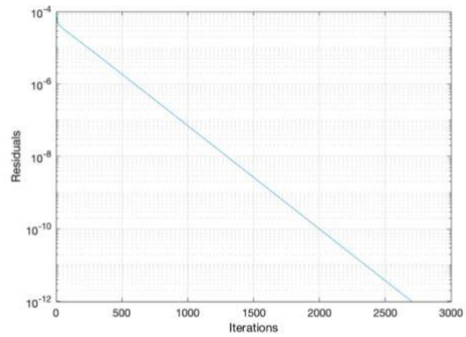
Omega=1.6



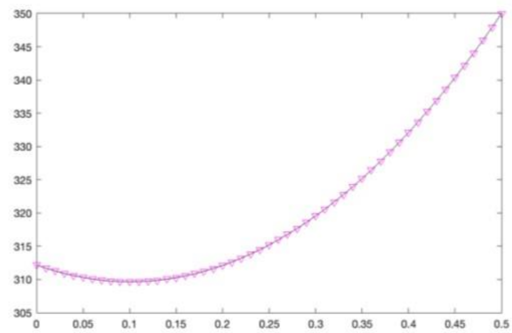


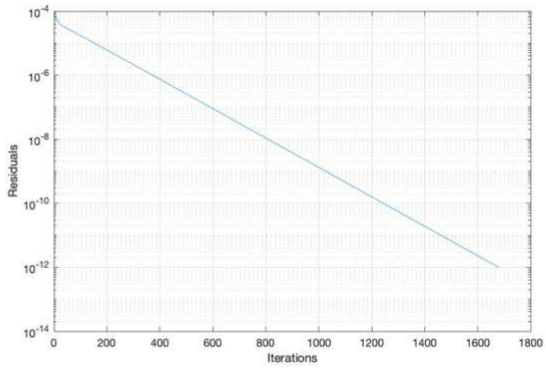
Omega=1.7



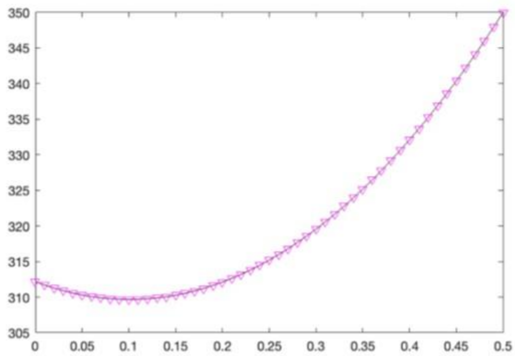


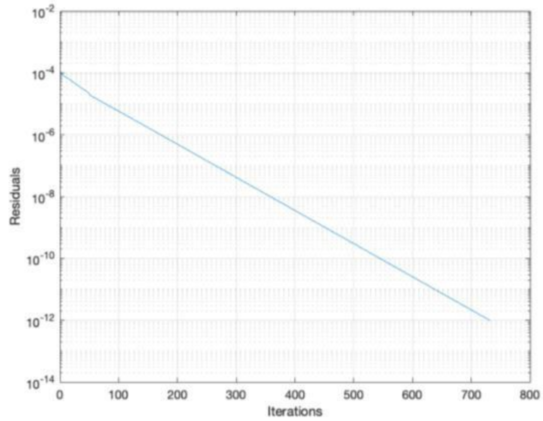
Omega=1.8



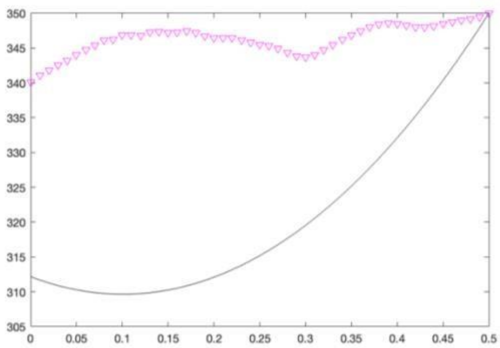


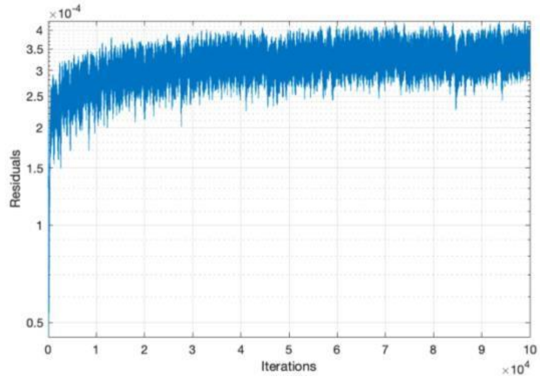
Omega=1.9





Omega=2.0





Omega	Successful (solution matches theory)?	N. of iterations	Best (Yes/No)?
$\omega = 1.1$	yes	12655	yes
$\omega = 1.2$	yes	10308	no
$\omega = 1.3$	yes	8321	no
$\omega = 1.4$	yes	6618	no
$\omega = 1.5$	yes	5141	no

$\omega = 1.6$	yes	3848	no
$\omega = 1.7$	yes	2703	no
$\omega = 1.8$	yes	1680	no
$\omega = 1.9$	yes	732	no
$\omega = 2$	no	100,000	no

Task (C)

My value for xy^2z=3.c=3.7 as 7 is my last ID digit

C1

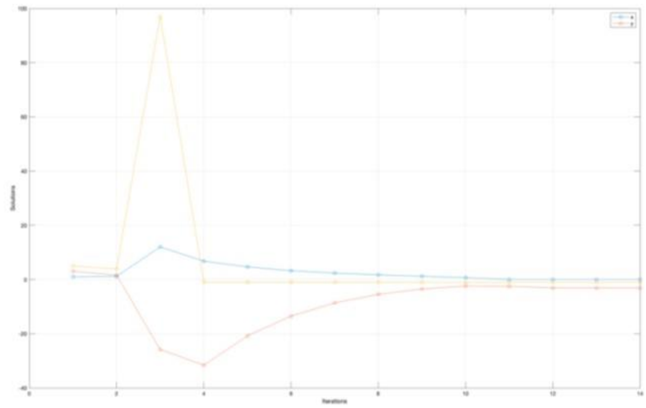
- Either write down the elements of the matrix in Microsoft Word, or write down clearly on paper and show a picture below

$$J = \begin{bmatrix} e^{y_i} & x_i e^{y_i} & 1 \\ -3x_i^2 & z_i & y_i \\ y_i^2 z_i & 2x_i y_i z_i & x_i y_i^2 \end{bmatrix}, F = \begin{bmatrix} x_i e^{y_i} + z_i + 1 \\ y_i z_i - x_i^3 - \square \\ x_i y_i^2 z_i - 3.7 \end{bmatrix}$$

C2

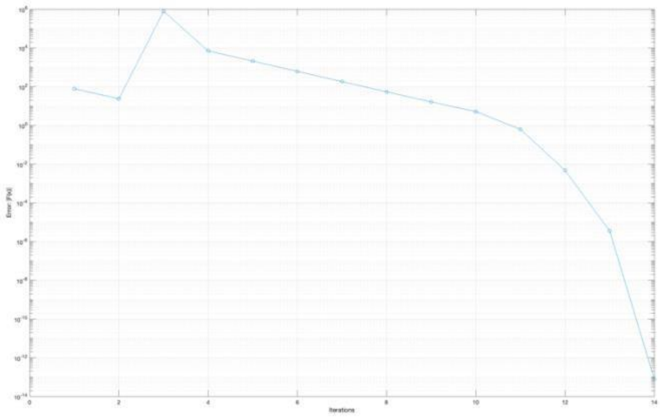
```
Task_C.m  ×  +
%Task C

clear all; close all; clc; %clears workspace, figures, command window

%functions
%u(x,y,z)=x*exp(y)+z+1
%v(x,y,z)=y*z-x^3-pi
%w(x,y,z)=x*y^2*z-3.7

maxIt=1000; tol=1e-8 %max iterations and tolerance
x=1; y=3; z=5; %initial guesses

err=sum(abs(x*(exp(y)+z+1)))+sum(abs(y*z-x^3-pi))+sum(abs(x*y^2*z-3.7));
%error=err=|u|+|v|+|w|

i=1;
while err(i)>tol & i<maxIt
    J(1,1)=exp(y(i)); % du/dx
    J(1,2)=x(i)*exp(y(i)); % du/dy
    J(1,3)=1; %du/dz
    J(2,1)=-1*(3*x(i)^2); %dv/dx
    J(2,2)=z(i); % dv/dy
    J(2,3)=y(i); % dv/dz
    J(3,1)=y(i)^2*z(i); % dw/dx
    J(3,2)=2*x(i)*y(i)*z(i); % dw/dy
    J(3,3)=x(i)*y(i)^2; % dw/dz
    F(1,1)=x(i)*exp(y(i))+z(i)+1;
    F(2,1)=y(i)*z(i)-x(i)^3-pi;
    F(3,1)=x(i)*y(i)^2*z(i);
    X(1,1)=x(i);
    X(2,1)=y(i);
    X(3,1)=z(i);

    X=J\(-F+J*X); % Backslash operator to solve the linear system
    x(i+1)=X(1); y(i+1)=X(2); z(i+1)=X(3); % New guess values

    F(1)=x(i+1)*exp(y(i+1))+z(i+1)+1; % Needed to compute new error
    F(2)=y(i+1)*z(i+1)-x(i+1)^3-pi; % Needed to compute new error
    F(3)=x(i+1)*y(i+1)^2*z(i+1); % Needed to compute new error

    err(i+1)=sum(abs(F));
    i=i+1;
end

figure('color','w','units','Centimeters','position',[5 5 7.5 7]);
plot(x,'o-'); hold on; plot(y,'o-'); hold on; plot(z,'o-')
grid on; xlabel('Iterations'); ylabel('Solutions'); legend('x','y');
figure('color','w'); semilogy(err,'o-'); grid on
xlabel('Iterations'); ylabel('Error: |F(x)|')
```

C3



Converged values of the solution	Use 5 significant digits
x	7.9662*10^-15
y	-3.1416
z	-1.0000

C4



Error at convergence	Number of iterations
7.91781*10^-14	14