NUMERICAL ANALYSIS UMA011

MATLAB Practicals (ODD Semester2021-2022)

B.E. Second Year

Thapar Institute of Engineering and Technology, Patiala

Name: Samarjot Singh

Roll No.: 102003242

Group: 2CO10

Contents

S.No.	Experiment	Date	Signature
1. (a) (b)	Use intermediate value theorem to find the interval of the roots. Find the root of non-linear equation $f(x) = 0$ using bisection method.		
2.	Find the root of non-linear equation $f(x) = 0$ using Newton's and secant methods		
3.	Find the root of non-linear equation $f(x) = 0$ using fixed-point iteration method		
4. (a) (b)	Solve system of linear equations Ax = b using Gauss elimination method. Further use it to apply LU factorization method for solving system of linear equations		
5.	Solve system of linear equations Ax = b using Gauss-Seidel and SOR iterative methods.		
6. (a) (b)	Find a dominant eigen-value and associated eigen-vector by Power method. Implement Lagrange interpolating polynomials of degree $\leq n$ on n+1 discrete data points.		
7.	Implement Newton's divided difference interpolating polynomials for n+1 discrete data points.		
8.	Fit a curve for given data points by using principle of least squares.		
9.	Integrate a function numerically using composite trapezoidal and Simpson's rules.		
10.	Find the solution of initial value problem using Euler and Runge-Kutta (fourth-order) methods.		

Experiment 1: Bisection Method

1. **Algorithm of Intermediate Value Theorem (IVT):** To determine all the subintervals $[a, b]$ of $[-N, N]$ that containing the roots of $f(x) = 0$.
   Input: function $f(x)$, and the values of h, N

```
for $i = -N : h : N$
if $f(i) * f(i + h) < 0$ then $a = i$ and $b = i + h$
end if
end i
```

2. **Algorithm of Bisection Method:** To determine a root of $f(x) = 0$ that is accurate within a specified tolerance value ϵ, given values a and b such that $f(a) * f(b) < 0$.

```
Define $c = (a + b)/2$.
if $f(a) * f(c) < 0$, then set $b = c$, otherwise $a = c$.
end if.
Until $|a - b| \leq \epsilon$ (tolerance value).
Print root as $c$.
```

   Stopping Criteria: Since this is an iterative method, we must determine some stopping criteria that will allow the iteration to stop. Criteria $|f(c_k)|$ very small can be misleading since it is possible to have $|f(c_k)|$ very small, even if c_k is not close to the root.
   The interval length after N iterations is $(b - a)/2^N$. So, to obtain an accuracy of ϵ, we must have

$$N \geq \frac{\log b - a \ - log \in}{\log 2}.$$

3. Students are required to write both the programs (IVT and Bisection) and implement it on the following examples.
   (i) Use bisection method in computing of $\overline{29}$ with $\epsilon = 0.001$, $N = 10$, $h = 1$.
   (ii) Determine the number of iterations necessary to solve $f(x) = x^3 + 4x^2 - 10 = 0$ with accuracy 10^{-3} using $a = 1$ and $b = 2$ and hence find the root with desired accuracy.
4. Thermistors are temperature-measuring devices based on the principle that the thermistor material exhibits a change in electrical resistance with a change in temperature.

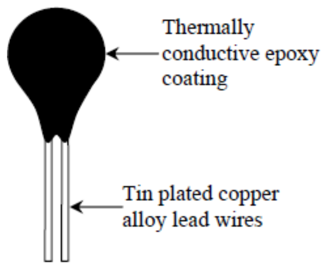

Thermally conductive epoxy coating
Tin plated copper alloy lead wires

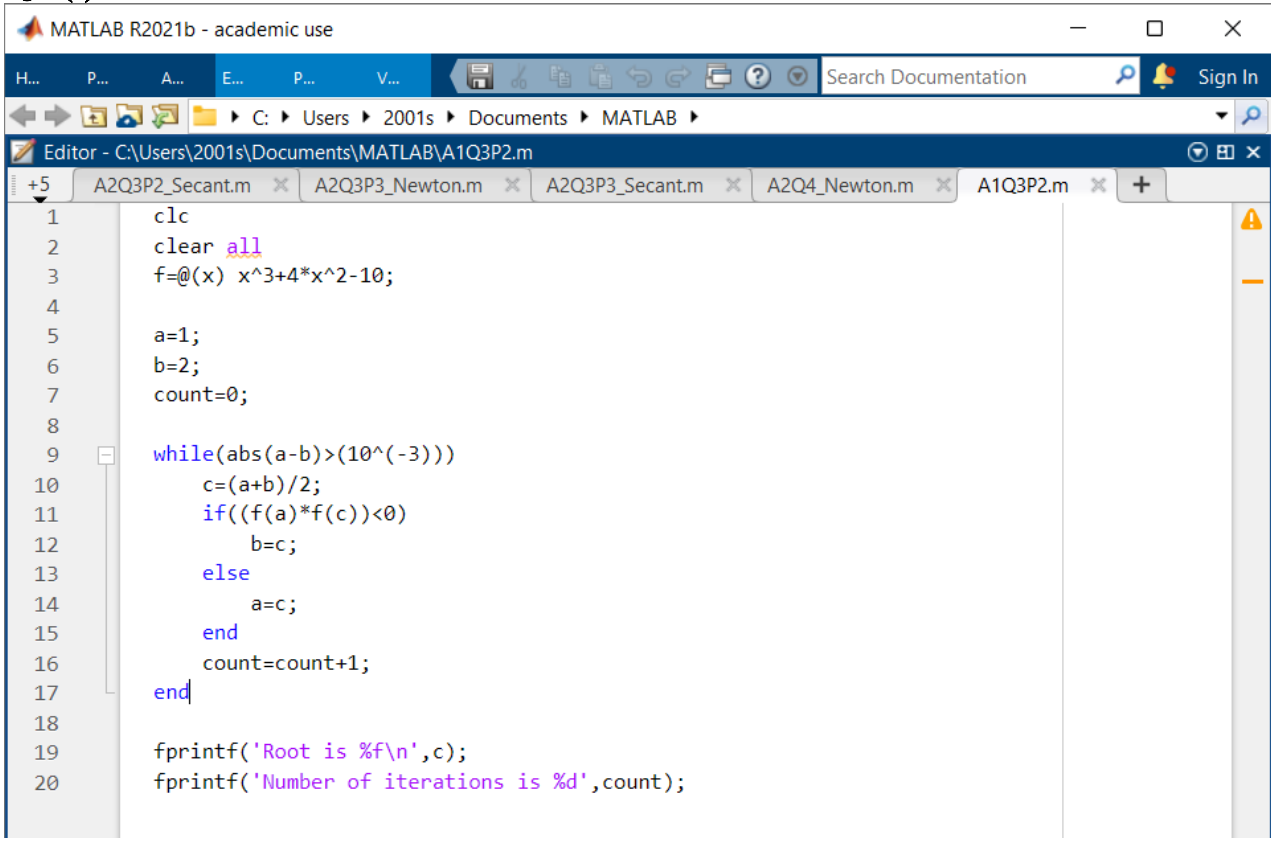
Figure 1 A ty 'cal thermistor.

By measuring the resistance of the thermistor material, one can then determine the temperature. For a 10K3A Betatherm thermistor, the relationship between the resistance R of the thermistor and the temperature is given by

$$\frac{1}{T} = 1.129241 \times 10^{-3} + 2.341077 \times 10^{-4} \ln R + 8.775468 \times 10^{-8} \ln(R)^{3}$$

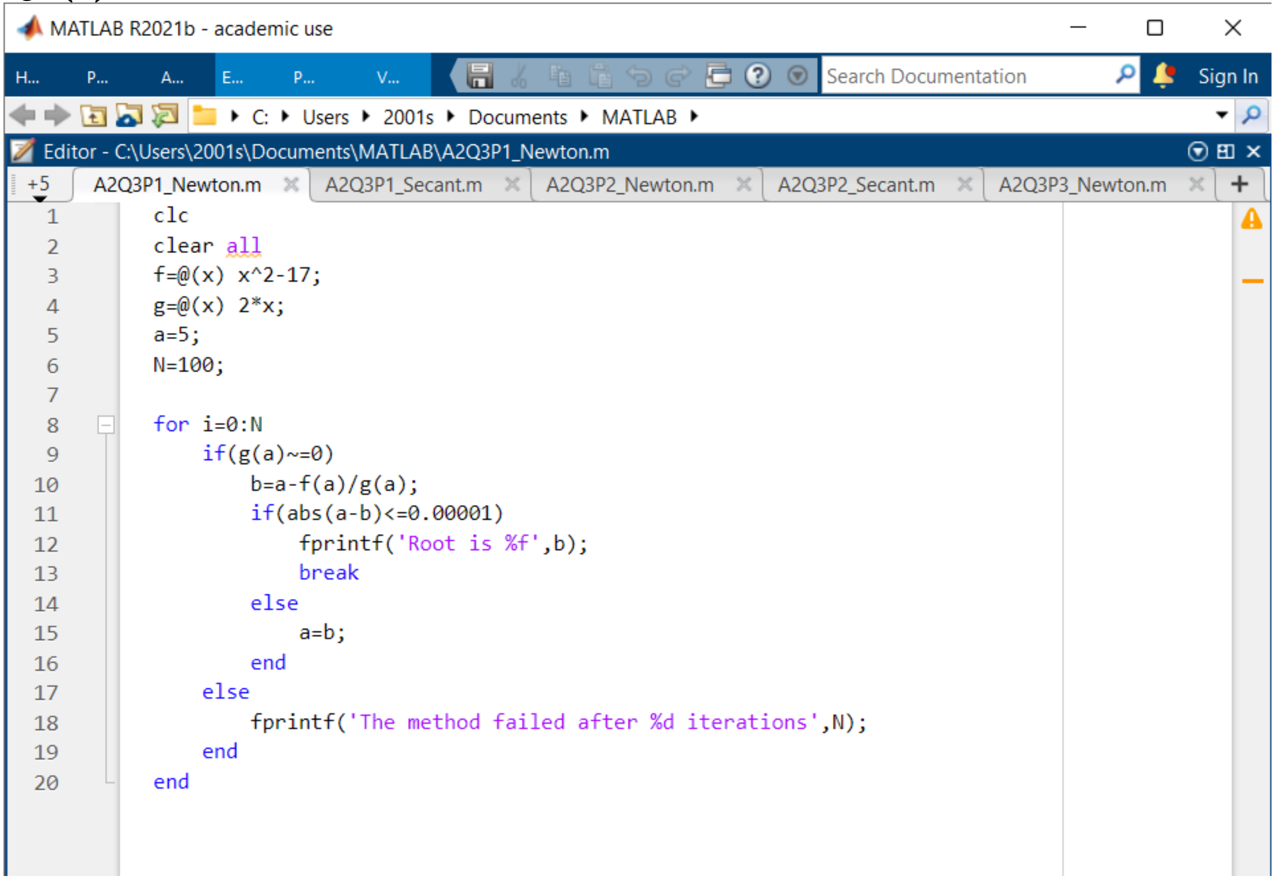
where T is in Kelvin and R is in ohms. Use the bisection method to find the resistance R at 18.99^{0}C.

Q3 (i): Root is 5.385742


MATLAB R2021b - academic use
H... P... A... E... P... V...
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A1Q3P2.m
+5
A2Q3P2_Secant.m
A2Q3P3_Newton.m
A2Q3P3_Secant.m
A2Q4_Newton.m
A1Q3P2.m

```
clc
clear all
f=@(x) x^3+4*x^2-10;

a=1;
b=2;
count=0;

while(abs(a-b)>(10^(-3)))
    c=(a+b)/2;
    if((f(a)*f(c))<0)
        b=c;
    else
        a=c;
    end
    count=count+1;
end

fprintf('Root is %f\n',c);
fprintf('Number of iterations is %d',count);
```

Q3 (ii): Root is 1.364258 Number of iterations is 10


MATLAB R2021b - academic use
H... P... A... E... P... V...
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A2Q3P1_Newton.m
+5
A2Q3P1_Newton.m
A2Q3P1_Secant.m
A2Q3P2_Newton.m
A2Q3P2_Secant.m
A2Q3P3_Newton.m

```
clc
clear all
f=@(x) x^2-17;
g=@(x) 2*x;
a=5;
N=100;

for i=0:N
    if(g(a)~=0)
        b=a-f(a)/g(a);
        if(abs(a-b)<=0.00001)
            fprintf('Root is %f',b);
            break
        else
            a=b;
        end
    else
        fprintf('The method failed after %d iterations',N);
    end
end
```

Experiment 2: Newton's and Secant Methods

1. **Algorithm for Newton's method:** Find a solution to $f(x) = 0$, given an initial approximation x_0.
   Input: Initial approximation x_0, tolerance value ϵ, maximum number of iterations N.
   Output: Approximate solution or message of failure.
   Step 1: Set $i = 1$.
   Step 2: While $i \le N$ do Steps 3 to 6.
   Step 3: Set $x_1 = x_0 - \frac{f(x_0)}{df(x_0)}$. (Compute x_i).
   Step 4: If $x_1 - x_0 \le \epsilon$ or $\frac{x_1 - x_0}{x_1} \le \epsilon$ then OUTPUT x_1; (The procedure is successful)
   STOP.

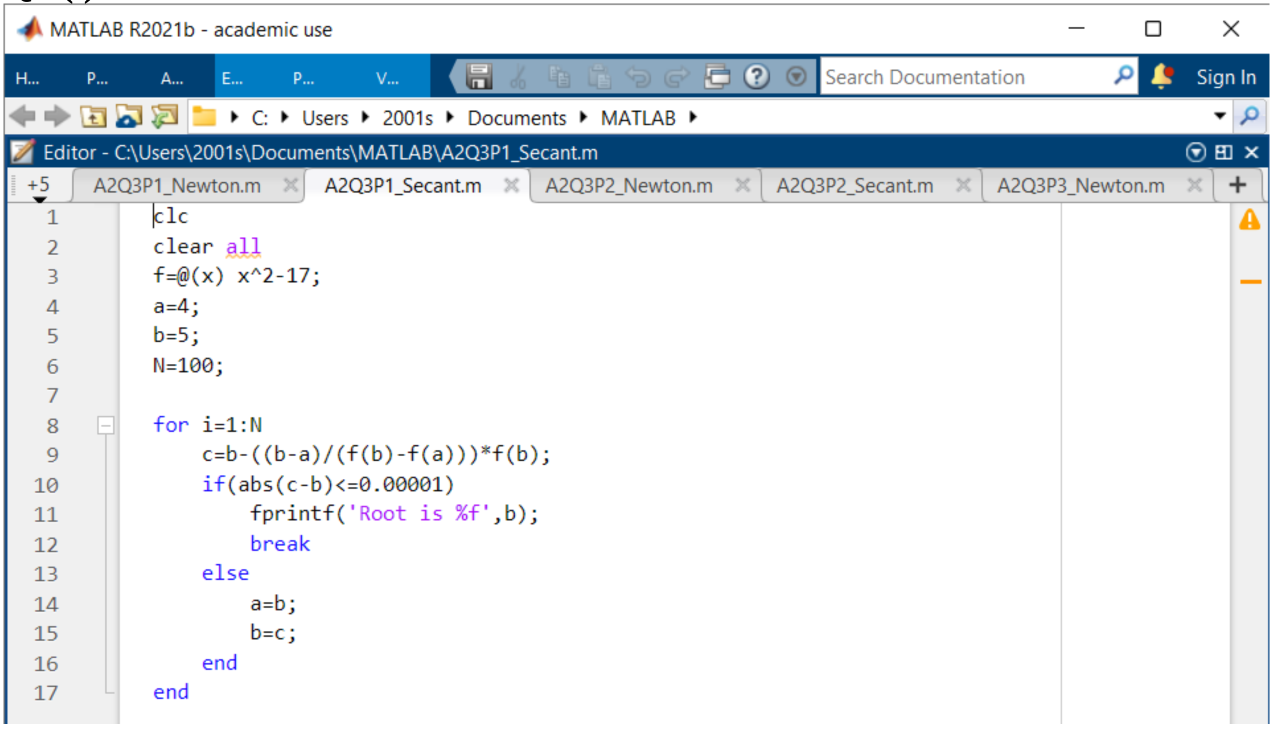
   Step 5: Set $i = i + 1$.
   Step 6: Set $x_0 = x_1$. (Update x_0)
   Step 7: Print ('The method failed after N iterations, N =', N); (The procedure is unsuccessful)
   STOP

2. **Algorithm for Secant method:** Find a solution to $f(x) = 0$, given an initial approximations x_0 and x_1.
   Input: Initial approximation x_0 and x_1, tolerance value ϵ, maximum number of iterations N.
   Output: Approximate solution or message of failure.
   Step 1: Set $i = 1$.
   Step 2: While $i \le N$ do Steps 3 to 6.
   Step 3: Set $x_2 = x_1 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_1)$. (Compute x_i).
   Step 4: If $x_2 - x_1 \le \epsilon$ or $\frac{x_2 - x_1}{x_2} \le \epsilon$ then OUTPUT x_2; (The procedure is successful)
   STOP.

   Step 5: Set $i = i + 1$.
   Step 6: Set $x_0 = x_1$ and $x_1 = x_2$. (Update x_0 and x_1)
   Step 7: Print ('The method failed after N iterations, N =', N); (The procedure is unsuccessful)
   STOP
3. Students are required to write both the program and implement it on the following examples. Take tolerance value $\epsilon = 0.00001$
   (i) Compute $\sqrt{17}$.
   (ii) The root of $exp(-x)(x^2 + 5x + 2) + 1 = 0$. Take initial guess –1.0.
   (iii) Find a non-zero solution of $x = 2\sin x$. (Apply IVT to find an initial guess)

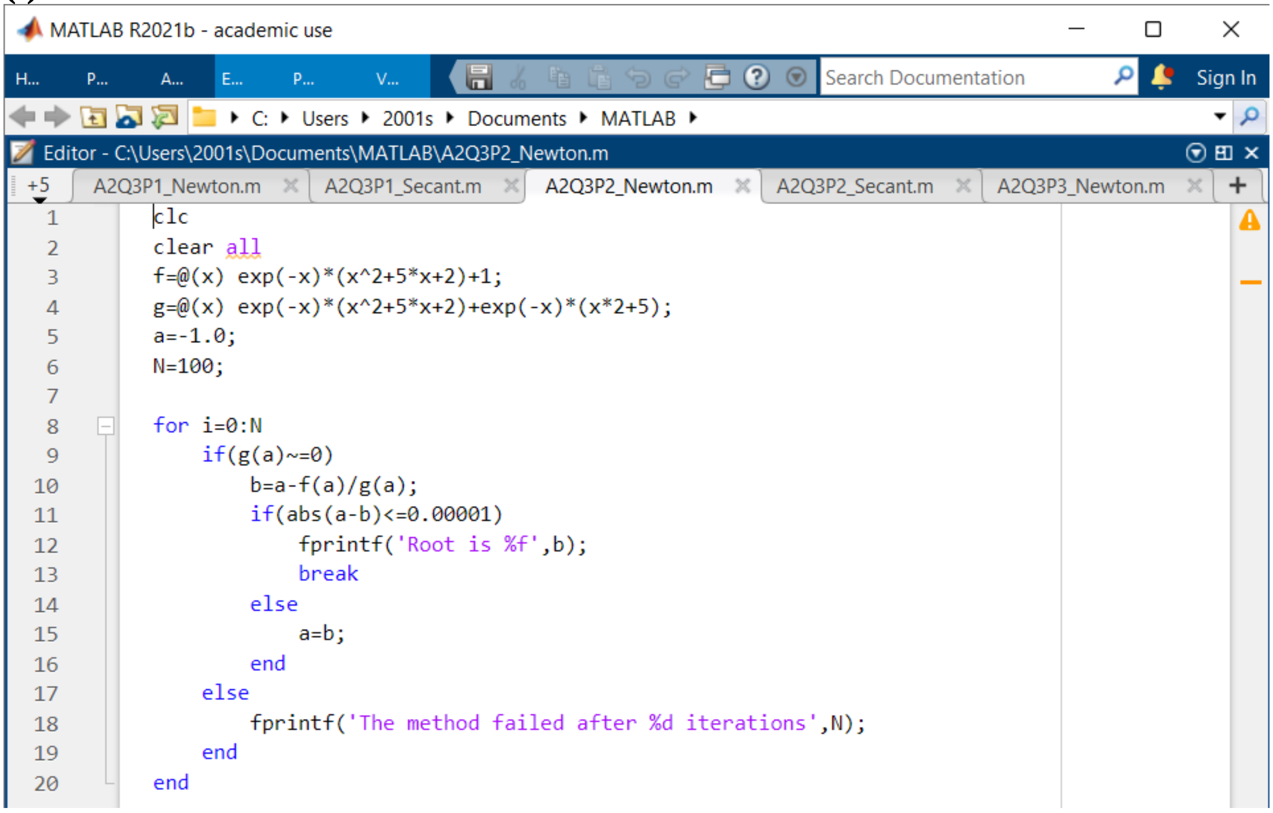
4. An oscillating current in an electric circuit is described by $i = 9e^{-t} \sin(2\pi t)$, where t is in seconds. Determine the lowest value of t such that $i = 3.5$.

Q3 (i): Newton's **Root is 4.123106**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB ▸
Editor - C:\Users\2001s\Documents\MATLAB\A2Q3P1_Secant.m
A2Q3P1_Newton.m
A2Q3P1_Secant.m
A2Q3P2_Newton.m
A2Q3P2_Secant.m
A2Q3P3_Newton.m

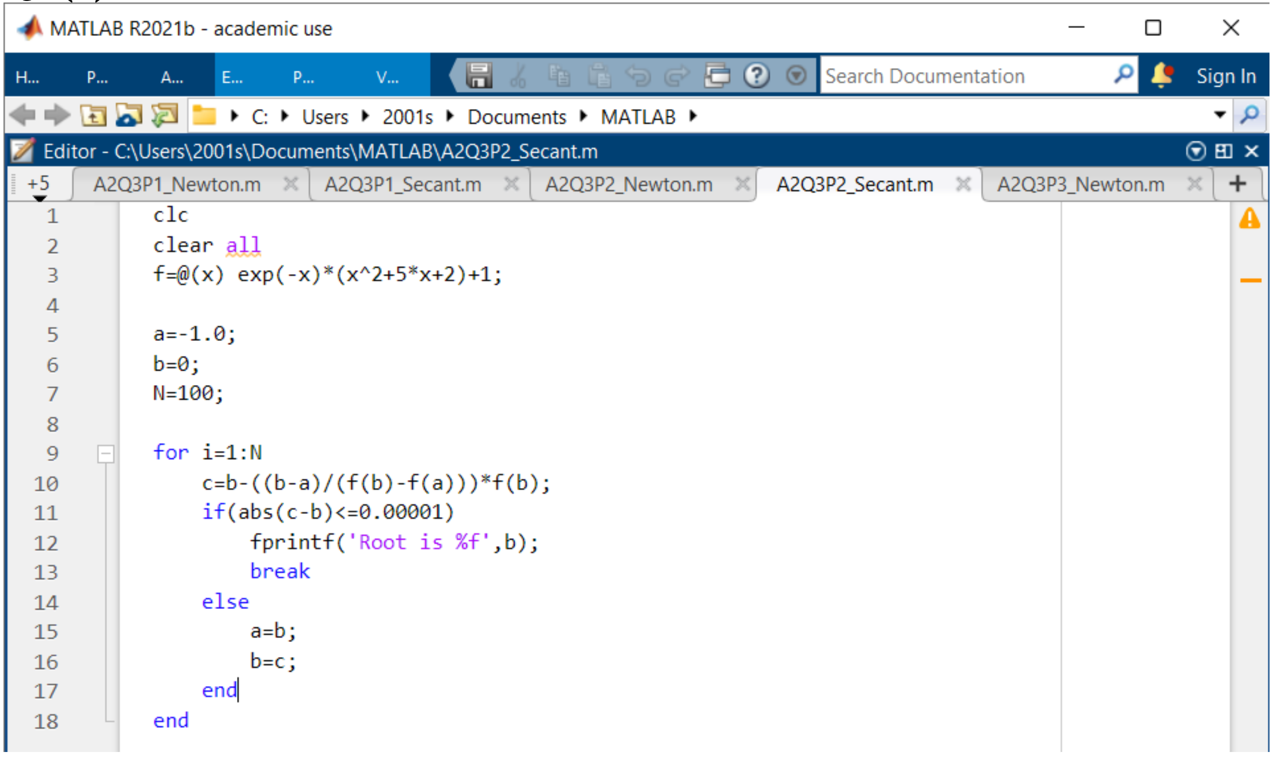
```
clc
clear all
f=@(x) x^2-17;
a=4;
b=5;
N=100;

for i=1:N
    c=b-((b-a)/(f(b)-f(a)))*f(b);
    if(abs(c-b)<=0.00001)
        fprintf('Root is %f',b);
        break
    else
        a=b;
        b=c;
    end
end
```

(i): Secant **Root is 4.123107**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB ▸
Editor - C:\Users\2001s\Documents\MATLAB\A2Q3P2_Newton.m
A2Q3P1_Newton.m
A2Q3P1_Secant.m
A2Q3P2_Newton.m
A2Q3P2_Secant.m
A2Q3P3_Newton.m

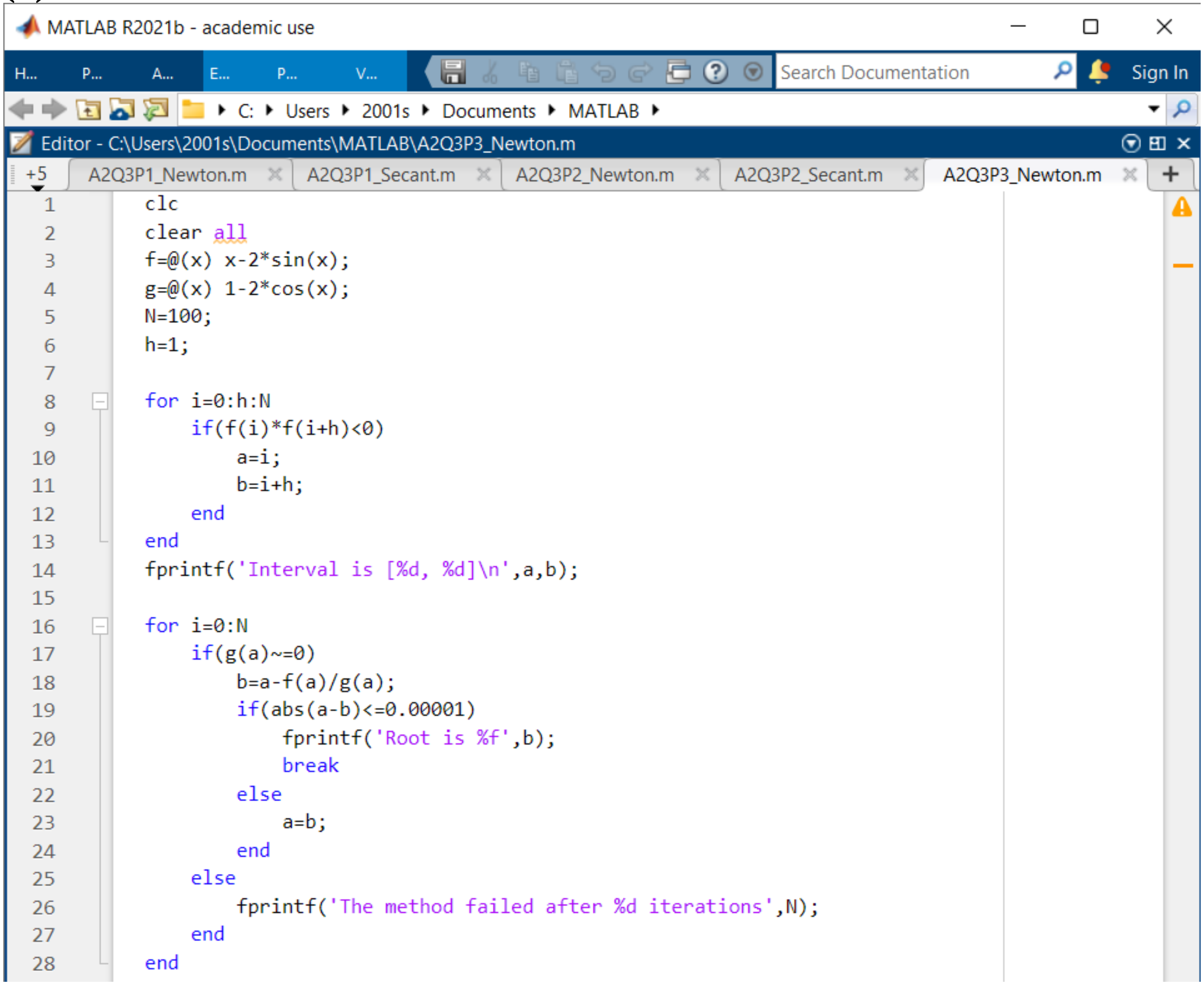
```
clc
clear all
f=@(x) exp(-x)*(x^2+5*x+2)+1;
g=@(x) exp(-x)*(x^2+5*x+2)+exp(-x)*(x*2+5);
a=-1.0;
N=100;

for i=0:N
    if(g(a)~=0)
        b=a-f(a)/g(a);
        if(abs(a-b)<=0.00001)
            fprintf('Root is %f',b);
            break
        else
            a=b;
        end
    else
        fprintf('The method failed after %d iterations',N);
    end
end
```

Q3 (ii): Newton's **Root is -0.579158**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A2Q3P2_Secant.m
A2Q3P1_Newton.m
A2Q3P1_Secant.m
A2Q3P2_Newton.m
A2Q3P2_Secant.m
A2Q3P3_Newton.m

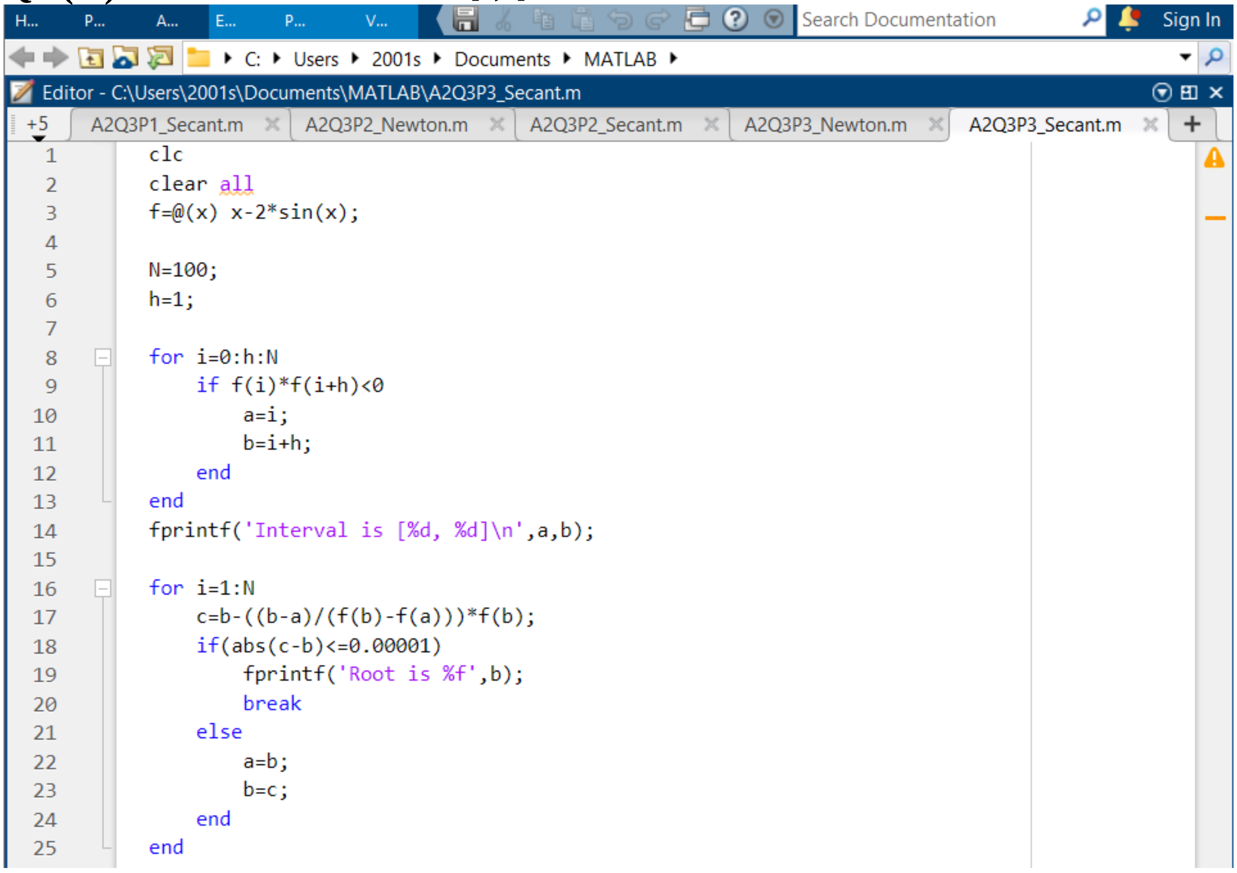
```
clc
clear all
f=@(x) exp(-x)*(x^2+5*x+2)+1;

a=-1.0;
b=0;
N=100;

for i=1:N
    c=b-((b-a)/(f(b)-f(a)))*f(b);
    if(abs(c-b)<=0.00001)
        fprintf('Root is %f',b);
        break
    else
        a=b;
        b=c;
    end
end
```

(ii): Secant **Root is -0.579167**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A2Q3P3_Newton.m
A2Q3P1_Newton.m
A2Q3P1_Secant.m
A2Q3P2_Newton.m
A2Q3P2_Secant.m
A2Q3P3_Newton.m

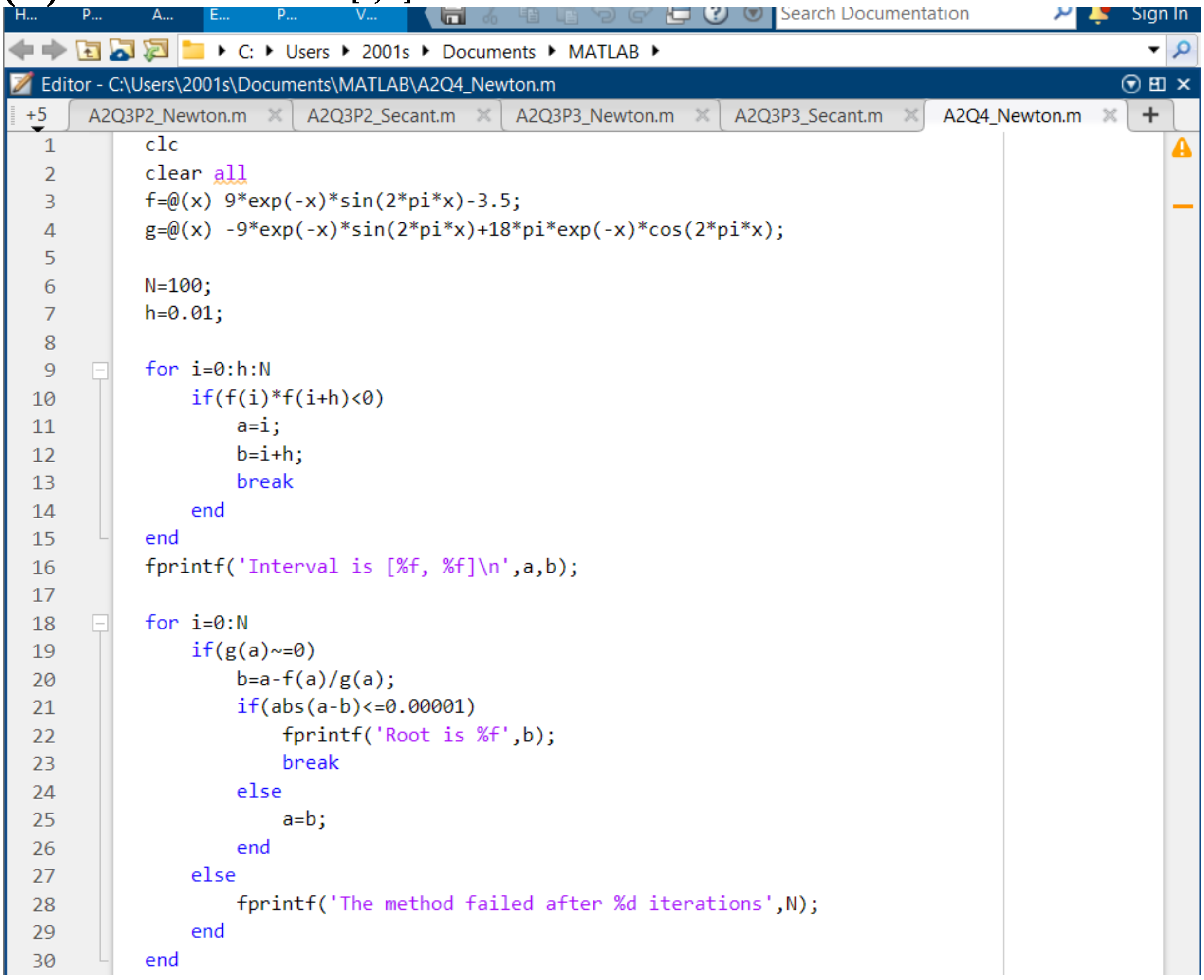
```
clc
clear all
f=@(x) x-2*sin(x);
g=@(x) 1-2*cos(x);
N=100;
h=1;

for i=0:h:N
    if(f(i)*f(i+h)<0)
        a=i;
        b=i+h;
    end
end
fprintf('Interval is [%d, %d]\n',a,b);

for i=0:N
    if(g(a)~=0)
        b=a-f(a)/g(a);
        if(abs(a-b)<=0.00001)
            fprintf('Root is %f',b);
            break
        else
            a=b;
        end
    else
        fprintf('The method failed after %d iterations',N);
    end
end
```

Q3 (iii): Newton's **Interval is [1,2] Root is 1.895494**



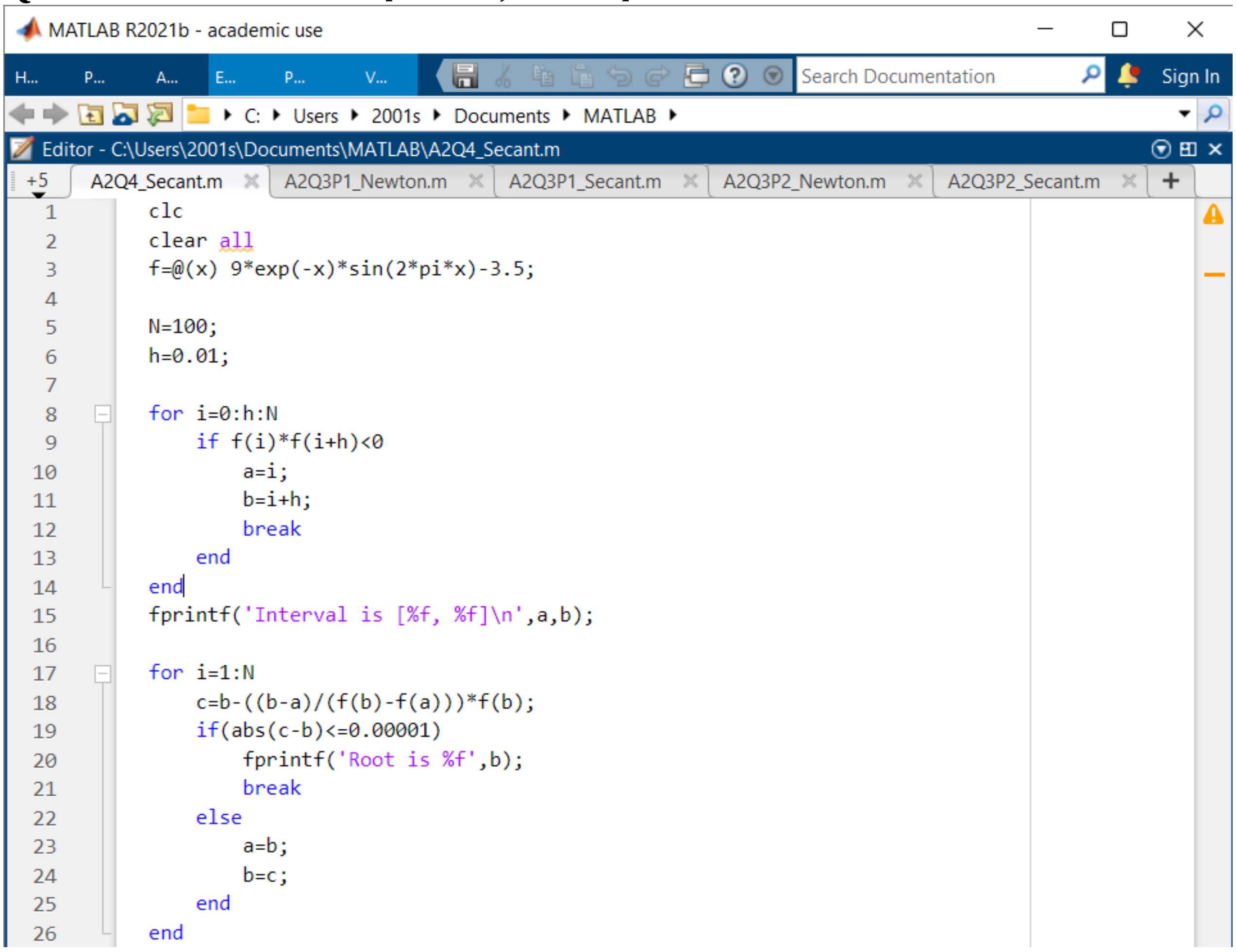
```
clc
clear all
f=@(x) x-2*sin(x);

N=100;
h=1;

for i=0:h:N
    if f(i)*f(i+h)<0
        a=i;
        b=i+h;
    end
end
fprintf('Interval is [%d, %d]\n',a,b);

for i=1:N
    c=b-((b-a)/(f(b)-f(a)))*f(b);
    if(abs(c-b)<=0.00001)
        fprintf('Root is %f',b);
        break
    else
        a=b;
        b=c;
    end
end
```

(iii): Secant **Interval is [1, 2] Root is 1.895493**



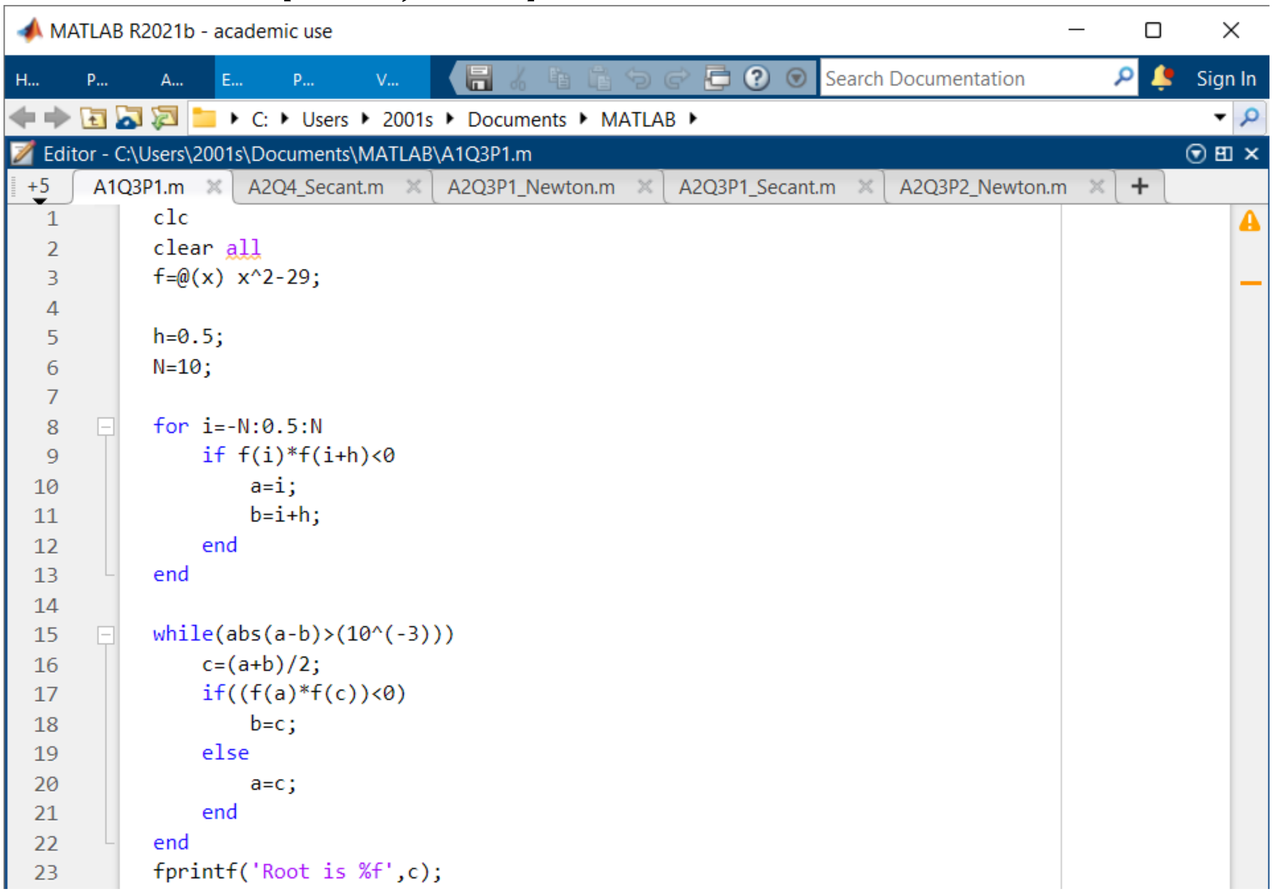
```
clc
clear all
f=@(x) 9*exp(-x)*sin(2*pi*x)-3.5;
g=@(x) -9*exp(-x)*sin(2*pi*x)+18*pi*exp(-x)*cos(2*pi*x);

N=100;
h=0.01;

for i=0:h:N
    if(f(i)*f(i+h)<0)
        a=i;
        b=i+h;
        break
    end
end
fprintf('Interval is [%f, %f]\n',a,b);

for i=0:N
    if(g(a)~=0)
        b=a-f(a)/g(a);
        if(abs(a-b)<=0.00001)
            fprintf('Root is %f',b);
            break
        else
            a=b;
        end
    else
        fprintf('The method failed after %d iterations',N);
    end
end
```

Q4: Newton's **Interval is [0.060000, 0.070000] Root is 0.068354**


MATLAB R2021b - academic use
H... P... A... E... P... V...
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A2Q4_Secant.m
+5 A2Q4_Secant.m A2Q3P1_Newton.m A2Q3P1_Secant.m A2Q3P2_Newton.m A2Q3P2_Secant.m

```
clc
clear all
f=@(x) 9*exp(-x)*sin(2*pi*x)-3.5;

N=100;
h=0.01;

for i=0:h:N
    if f(i)*f(i+h)<0
        a=i;
        b=i+h;
        break
    end
end
fprintf('Interval is [%f, %f]\n',a,b);

for i=1:N
    c=b-((b-a)/(f(b)-f(a)))*f(b);
    if(abs(c-b)<=0.00001)
        fprintf('Root is %f',b);
        break
    else
        a=b;
        b=c;
    end
end
```

Secant **Interval is [0.060000, 0.070000] Root is 0.068354**


MATLAB R2021b - academic use
H... P... A... E... P... V...
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A1Q3P1.m
+5 A1Q3P1.m A2Q4_Secant.m A2Q3P1_Newton.m A2Q3P1_Secant.m A2Q3P2_Newton.m

```
clc
clear all
f=@(x) x^2-29;

h=0.5;
N=10;

for i=-N:0.5:N
    if f(i)*f(i+h)<0
        a=i;
        b=i+h;
    end
end

while(abs(a-b)>(10^(-3)))
    c=(a+b)/2;
    if((f(a)*f(c))<0)
        b=c;
    else
        a=c;
    end
end
fprintf('Root is %f',c);
```

Experiment 3: Fixed-point Iteration Method

1. **Algorithm for Fixed-point iteration method:** To find a solution to $x = g(x)$, given an initial approximation x_0.

   Input: Initial approximation x_0, tolerance value ϵ, maximum number of iterations N.

   Output: Approximate solution or message of failure.

```
Step 1: Set $i = 1$.
Step 2: While $i \le N$ do Steps 3 to 6.
Step 3: Set $x_1 = g(x_0)$. (Compute $x_i$).
Step 4: If $|x_1 - x_0| \le \epsilon$ or $\left|\frac{x_1 - x_0}{x_1}\right| \le \epsilon$ then OUTPUT $x_1$; (The procedure is successful)
        STOP.
Step 5: Set $i = i + 1$.
Step 6: Set $x_0 = x_1$. (Update $x_0$)
Step 7: Print the output and STOP.
```

2. The equation f(x) = $x^3 + 4x^2 - 10 = 0$ has a unique root in [1,2]. There are many ways to change the equation to the fixed-point form $x = g(x)$ using simple algebraic manipulation. Let g_1, g_2, g_3, g_4 and g_5 are iteration functions obtained by the given function, then check which of the following iteration functions will converge to the fixed point? (Tolerance $\epsilon = 10^{-3}$)

   (a) $g_1(x) = x - x^3 - 4x^2 + 10$

   (b) $g_2(x) = \sqrt{\frac{10}{x} - 4x}$

   (c) $g_3(x) = 0.5\sqrt{10 - x^3}$

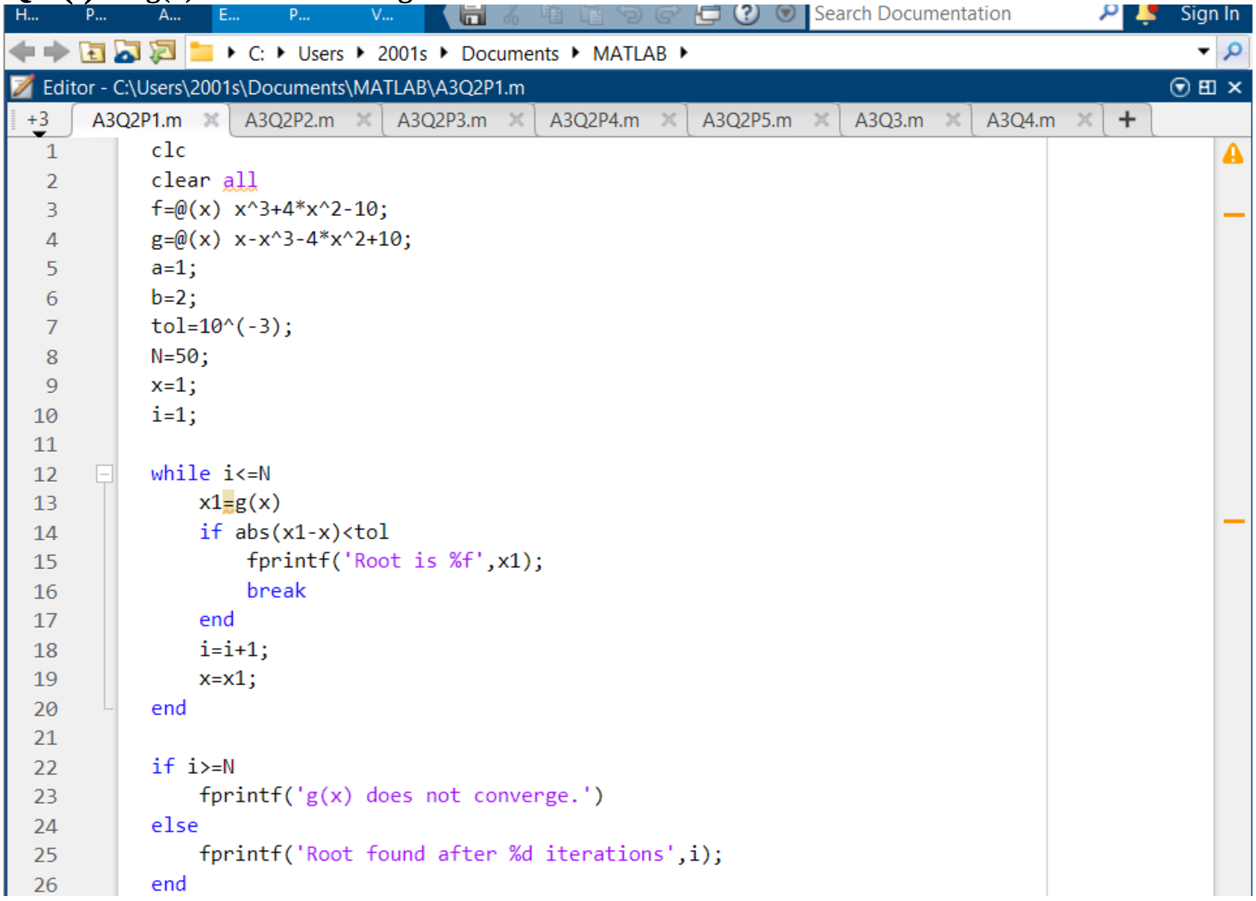
   (d) $g_4(x) = \sqrt{\frac{10}{4+x}}$

   (e) $g_5(x) = x - \frac{x^3+4x^2-10}{3x^2+8x}$

3. Find the smallest and second smallest positive roots of the equation tan(x) = 4x, with an accuracy of 10^{-3} using fixed-point iterations.

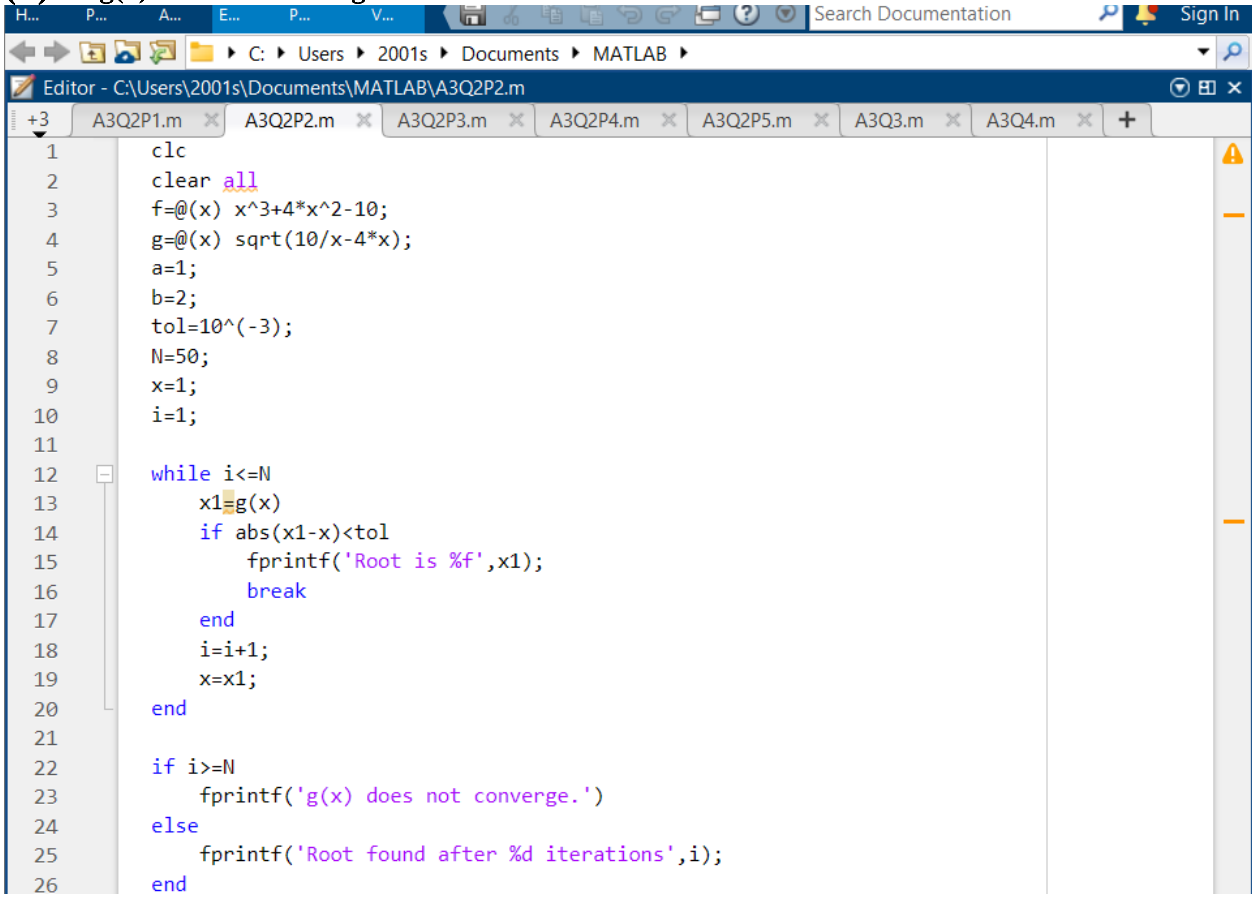
4. Use a fixed-point iteration method to determine a solution accurate to within 10^{-2} for $2\sin\pi x + x = 0$ on [1,2]. Use initial guess $x_0 = 1$.

Q2 (i): g(x) does not converge



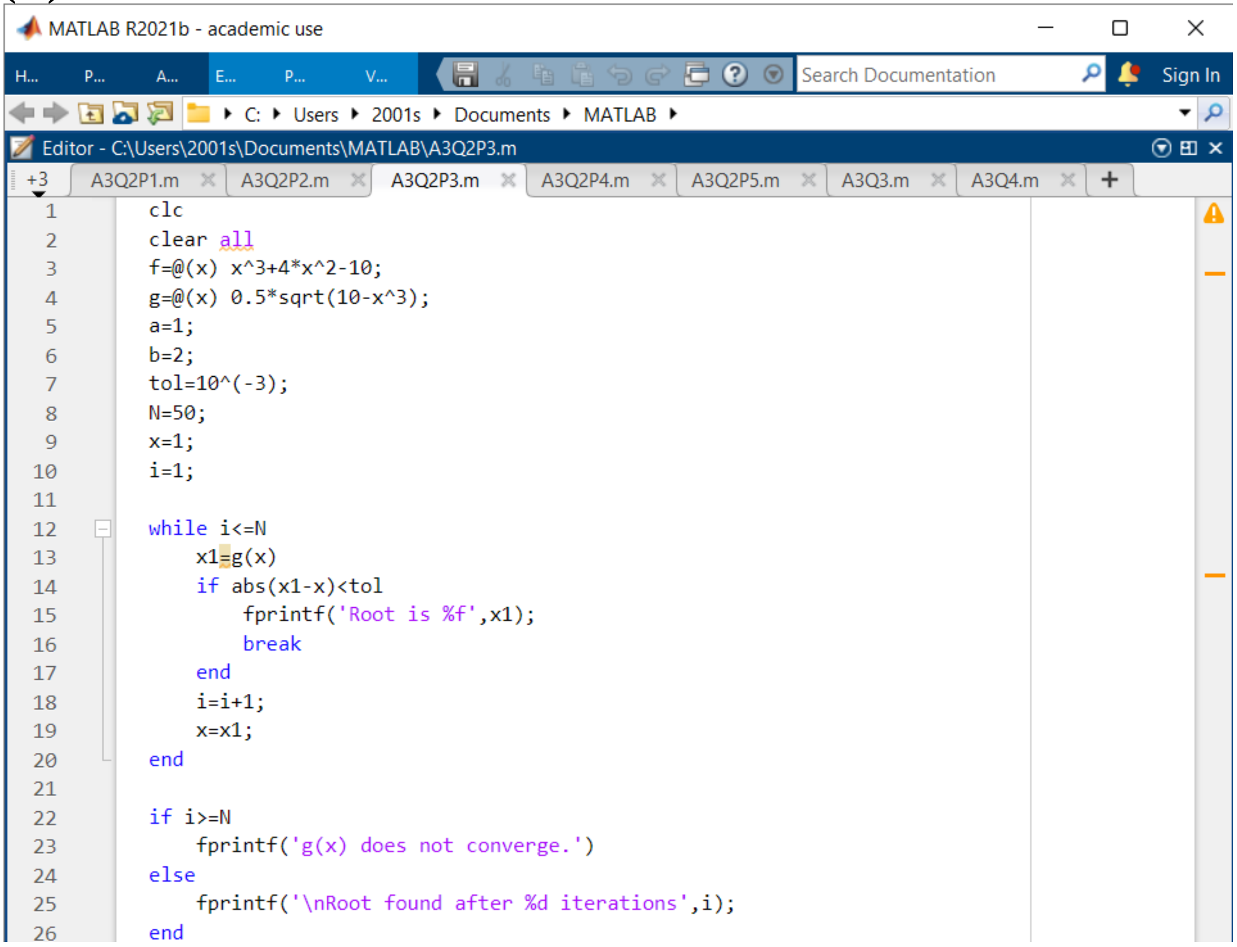
```
clc
clear all
f=@(x) x^3+4*x^2-10;
g=@(x) x-x^3-4*x^2+10;
a=1;
b=2;
tol=10^(-3);
N=50;
x=1;
i=1;

while i<=N
    x1=g(x)
    if abs(x1-x)<tol
        fprintf('Root is %f',x1);
        break
    end
    i=i+1;
    x=x1;
end

if i>=N
    fprintf('g(x) does not converge.')
else
    fprintf('Root found after %d iterations',i);
end
```

(ii): g(x) does not converge



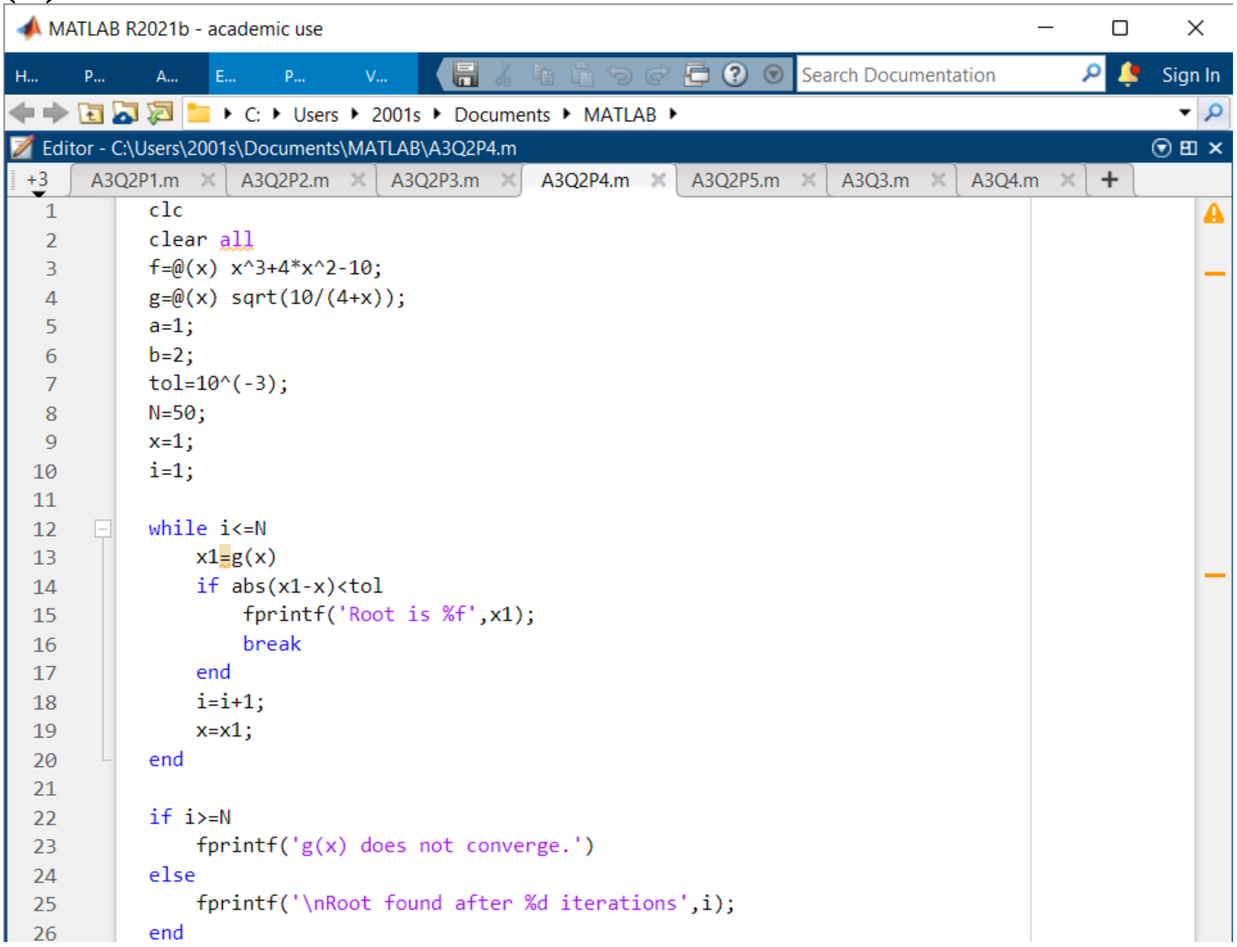
```
clc
clear all
f=@(x) x^3+4*x^2-10;
g=@(x) sqrt(10/x-4*x);
a=1;
b=2;
tol=10^(-3);
N=50;
x=1;
i=1;

while i<=N
    x1=g(x)
    if abs(x1-x)<tol
        fprintf('Root is %f',x1);
        break
    end
    i=i+1;
    x=x1;
end

if i>=N
    fprintf('g(x) does not converge.')
else
    fprintf('Root found after %d iterations',i);
end
```

(iii): Root is 1.365410 Root found after 11 iterations


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A3Q2P3.m
A3Q2P1.m
A3Q2P2.m
A3Q2P3.m
A3Q2P4.m
A3Q2P5.m
A3Q3.m
A3Q4.m

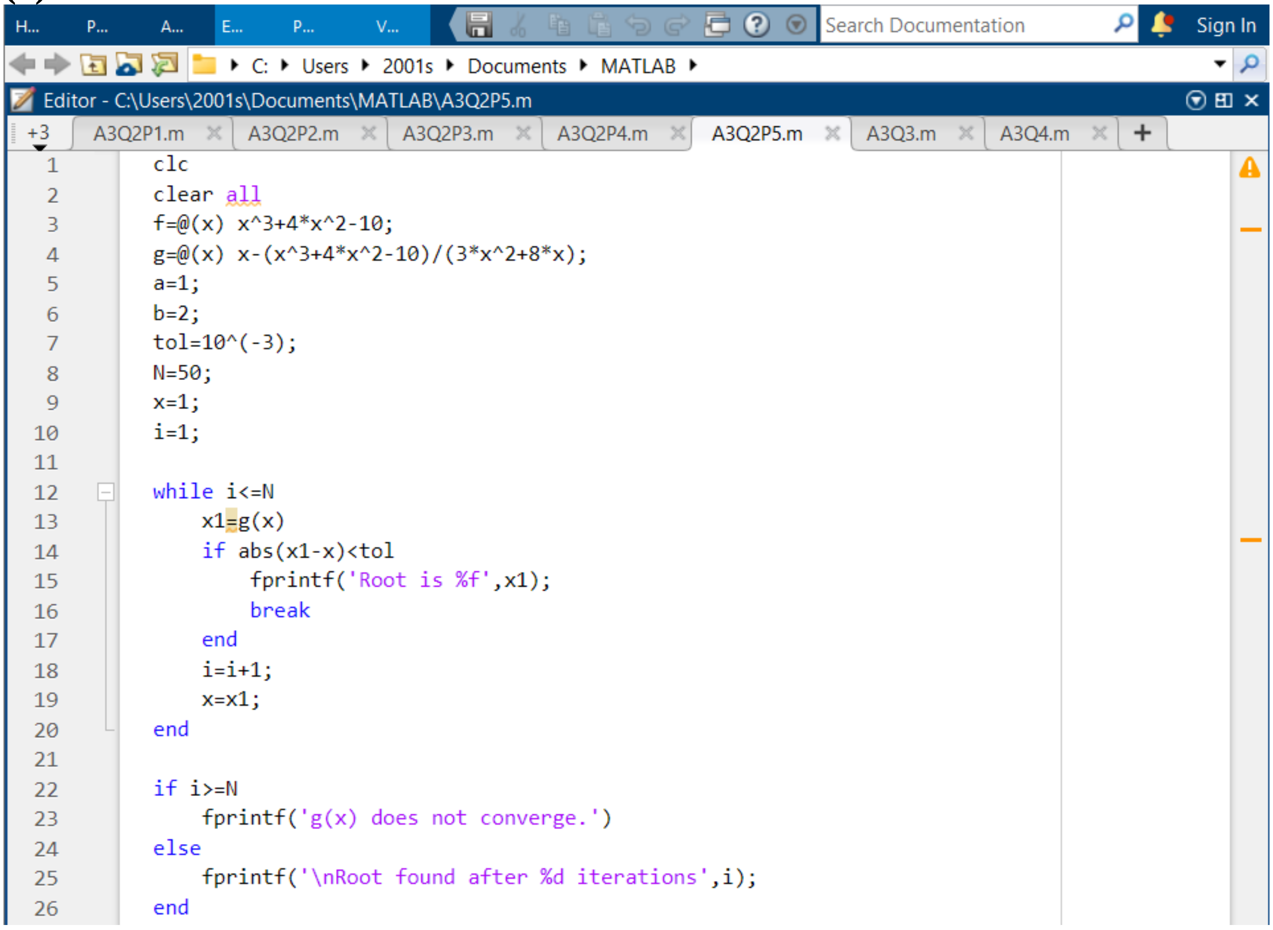
```
clc
clear all
f=@(x) x^3+4*x^2-10;
g=@(x) 0.5*sqrt(10-x^3);
a=1;
b=2;
tol=10^(-3);
N=50;
x=1;
i=1;

while i<=N
    x1=g(x)
    if abs(x1-x)<tol
        fprintf('Root is %f',x1);
        break
    end
    i=i+1;
    x=x1;
end

if i>=N
    fprintf('g(x) does not converge.')
else
    fprintf('\nRoot found after %d iterations',i);
end
```

(iv): Root is 1.365130 Root found after 4 iterations


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A3Q2P4.m
A3Q2P1.m
A3Q2P2.m
A3Q2P3.m
A3Q2P4.m
A3Q2P5.m
A3Q3.m
A3Q4.m

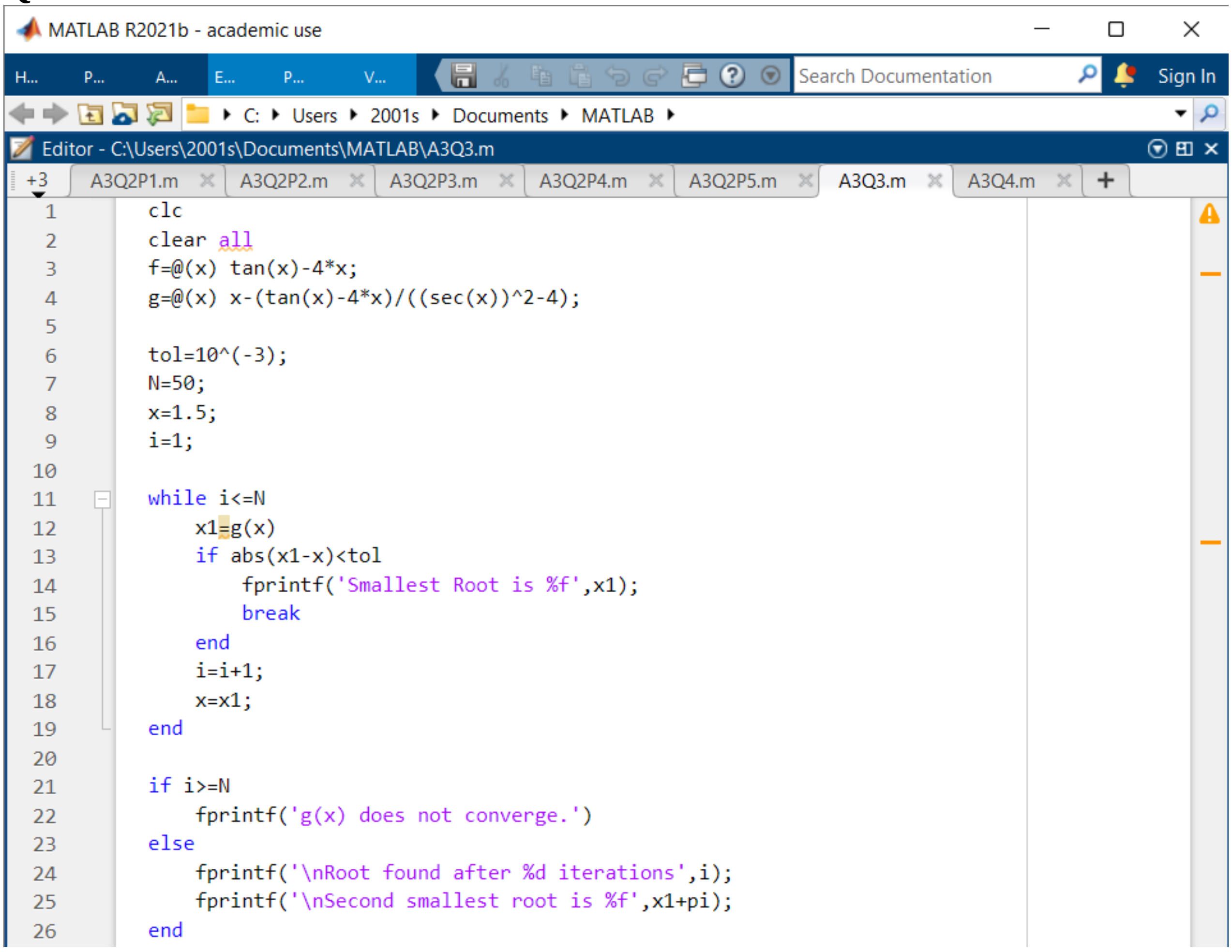
```
clc
clear all
f=@(x) x^3+4*x^2-10;
g=@(x) sqrt(10/(4+x));
a=1;
b=2;
tol=10^(-3);
N=50;
x=1;
i=1;

while i<=N
    x1=g(x)
    if abs(x1-x)<tol
        fprintf('Root is %f',x1);
        break
    end
    i=i+1;
    x=x1;
end

if i>=N
    fprintf('g(x) does not converge.')
else
    fprintf('\nRoot found after %d iterations',i);
end
```

(v): Root is 1.365230 Root found after 4 iterations


H... P... A... E... P... V...
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB ▸
Editor - C:\Users\2001s\Documents\MATLAB\A3Q2P5.m
+3 A3Q2P1.m A3Q2P2.m A3Q2P3.m A3Q2P4.m A3Q2P5.m A3Q3.m A3Q4.m

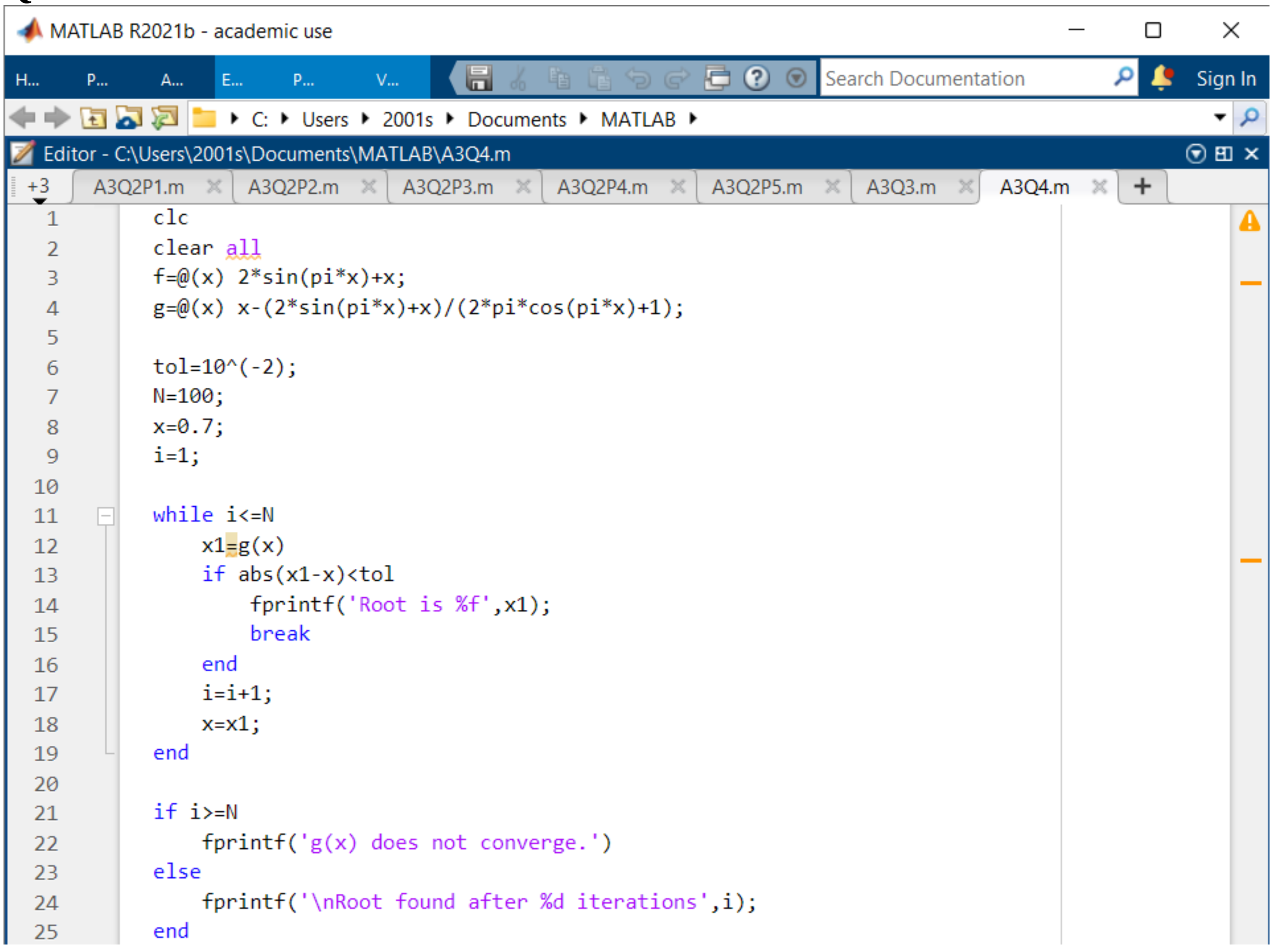
```
clc
clear all
f=@(x) x^3+4*x^2-10;
g=@(x) x-(x^3+4*x^2-10)/(3*x^2+8*x);
a=1;
b=2;
tol=10^(-3);
N=50;
x=1;
i=1;

while i<=N
    x1=g(x)
    if abs(x1-x)<tol
        fprintf('Root is %f',x1);
        break
    end
    i=i+1;
    x=x1;
end

if i>=N
    fprintf('g(x) does not converge.')
else
    fprintf('\nRoot found after %d iterations',i);
end
```

Q3: Smallest Root is 1.393249 Root found after 5 iterations Second smallest root is 4.534842


MATLAB R2021b - academic use
H... P... A... E... P... V...
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB ▸
Editor - C:\Users\2001s\Documents\MATLAB\A3Q3.m
+3 A3Q2P1.m A3Q2P2.m A3Q2P3.m A3Q2P4.m A3Q2P5.m A3Q3.m A3Q4.m

```
clc
clear all
f=@(x) tan(x)-4*x;
g=@(x) x-(tan(x)-4*x)/((sec(x))^2-4);

tol=10^(-3);
N=50;
x=1.5;
i=1;

while i<=N
    x1=g(x)
    if abs(x1-x)<tol
        fprintf('Smallest Root is %f',x1);
        break
    end
    i=i+1;
    x=x1;
end

if i>=N
    fprintf('g(x) does not converge.')
else
    fprintf('\nRoot found after %d iterations',i);
    fprintf('\nSecond smallest root is %f',x1+pi);
end
```

Q4: **Root is 1.682030 Root found after 4 iterations**


MATLAB R2021b - academic use
H... P... A... E... P... V...
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A3Q4.m
+3 A3Q2P1.m A3Q2P2.m A3Q2P3.m A3Q2P4.m A3Q2P5.m A3Q3.m A3Q4.m

```matlab
clc
clear all
f=@(x) 2*sin(pi*x)+x;
g=@(x) x-(2*sin(pi*x)+x)/(2*pi*cos(pi*x)+1);

tol=10^(-2);
N=100;
x=0.7;
i=1;

while i<=N
    x1=g(x)
    if abs(x1-x)<tol
        fprintf('Root is %f',x1);
        break
    end
    i=i+1;
    x=x1;
end

if i>=N
    fprintf('g(x) does not converge.')
else
    fprintf('\nRoot found after %d iterations',i);
end
```

Experiment 4: Gauss Elimination and LU Factorization Methods

1. **Algorithm for Gauss elimination method:** Find a solution of system of linear equations.
Input: Number of unknowns and equations n,
Augmented matrix $A = a_{ij}$, where $1 \le i \le n$, and $1 \le j \le n + 1$.
Output: Solution $(x_1, x_2, \cdots, x_n)$ or message that the linear system has no unique solution.

```
Step 1: For $i = 1,2,\cdots,n-1$ do Steps 2 – 4. (Elimination process)
Step 2: Let $p$ be the smallest integer with $i \le p \le n$ and $a_{pi} \ne 0$.
          If no integer $p$ can be found then
          OUTPUT (‘no unique solution exists’);
          STOP.
Step 3: If $p \ne i$ then perform $E_p \boxed{\leftrightarrow} E_i$ .
Step 4: For $j = i+1,\cdots,n$ do Steps 5 and 6.
Step 5: Set $m_{ji} = a_{ji} \;\; a_{ii}$.
Step 6: Perform $E_j - m_{ji} E_i \boxed{\leftrightarrow} E_j$
; Step 7: If $a_{nn} = 0$ then
          OUTPUT (‘no unique solution exists’);
          STOP.
Step 8: Set $x_n = a_{n,n+1} \, a_{nn}$ . (Start backward substitution)
Step 9: For $i = n-1,\ n-2,\cdots,1$ set $x_i = a_{i,n+1} - {}^{n}_{j=i+1} \; a_{ij}\, x_j \;\; a_{ii}$ .
Step 10: OUTPUT $(x_1, x_2, \cdots, x_n)$. (Procedure completed successfully) STOP.
```

2. **Algorithm for LU factorization method:** Find a solution of system of linear equations.
Input: Number of unknowns and equations n, matrix $A = a_{ij}$, $1 \le i \le n$, $1 \le j \le n$ evaluated by executing Steps 1 to 6 of Gauss Elimination method, m_{ji}, $1 \le i \le n$, $1 \le j \le n$ evaluated in Step 5 Gauss Elimination method.

```
Step 1: Take U = A.
Step 2: Set $l_{ji} = m_{ji}$.
Step 3: Set $l_{ii} = 1$
Step 4: Rewrite A$x$ = b as (LU)$x$ = b
Step 5: Solve L$y$ = b for $y$ and U$x$ = $y$ for $x$.
```

3. Use Gauss elimination method to find the solution of the following linear system of equations:

$$10x + 8y - 3z + u = 16$$
$$2x + 10y + z - 4u = 9$$
$$3x - 4y + 10z + u = 10$$
$$2x + 2y - 3z + 10u = 11$$

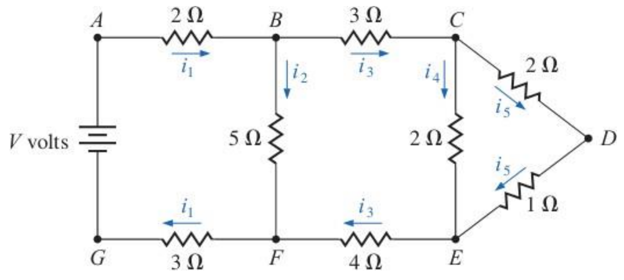
4. Solve the following linear system of equations:

$$\pi x_1 + \sqrt{2}x_2 - x_3 + x_4 = 0$$
$$e x_1 - x_2 + x_3 + 2x_4 = 1$$
$$x_1 + x_2 - \sqrt{3}x_3 + x_4 = 2$$
$$-x_1 - x_2 + x_3 - \sqrt{5}x_4 = 3$$

5. Kirchhoff’s laws of electrical circuits state that both the net flow of current through each junction and the net voltage drop around each closed loop of a circuit are zero. Suppose that a

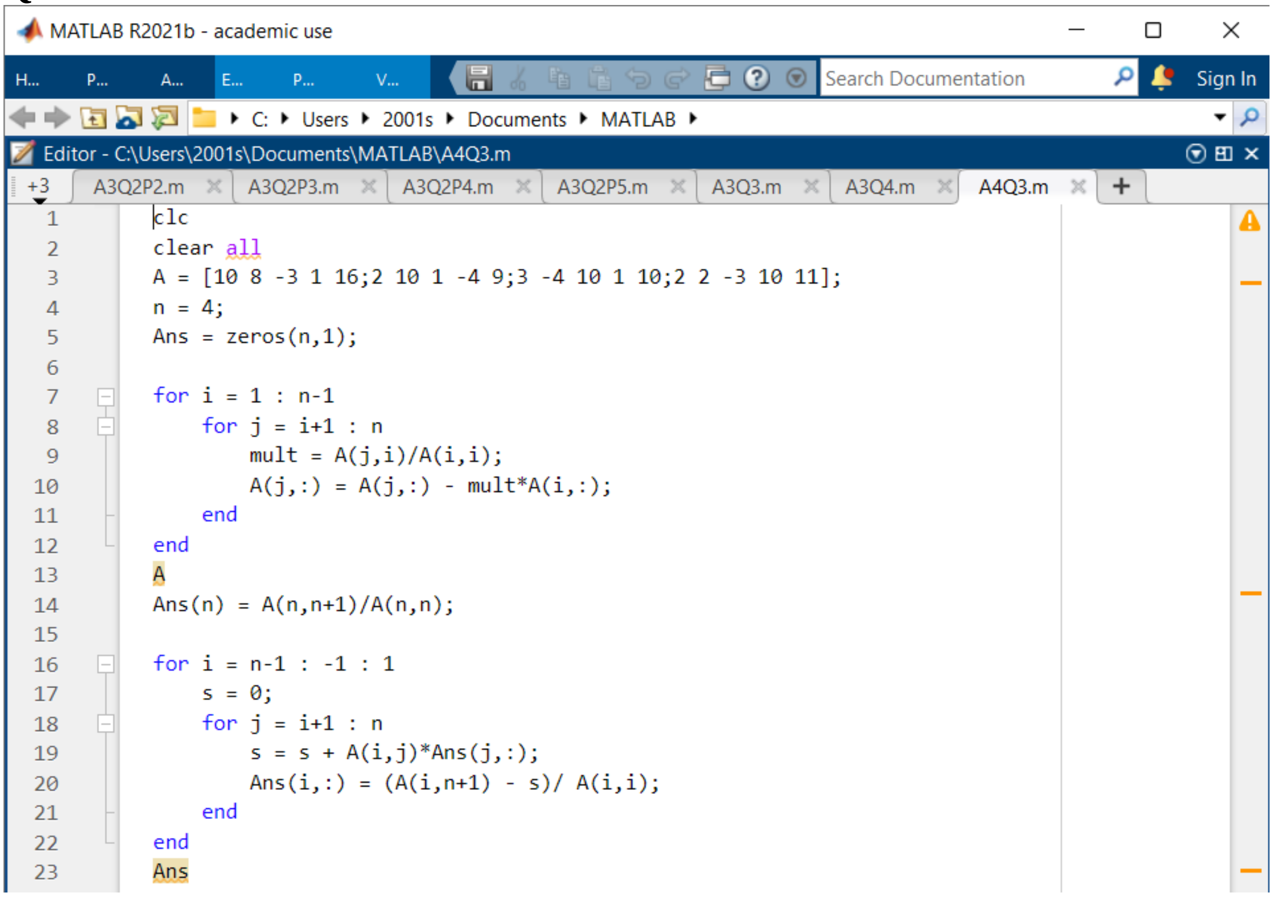
potential of V volts is applied between the points A and G in the circuit and that i_1, i_2, i_3, i_4 and i_5 represent current flow as shown in the diagram. Using G as a reference point, Kirchhoff's laws imply that the currents satisfy the following system of linear equations:

$5i_1 + 5i_2 = V$
$i_3 - i_4 - i_5 = 0$ $2i_4$
$- 3i_5 = 0$
$i_1 - i_2 - i_3 = 0$ $5i_2$
$- 7i_3 - 2i_4$
$= 0$



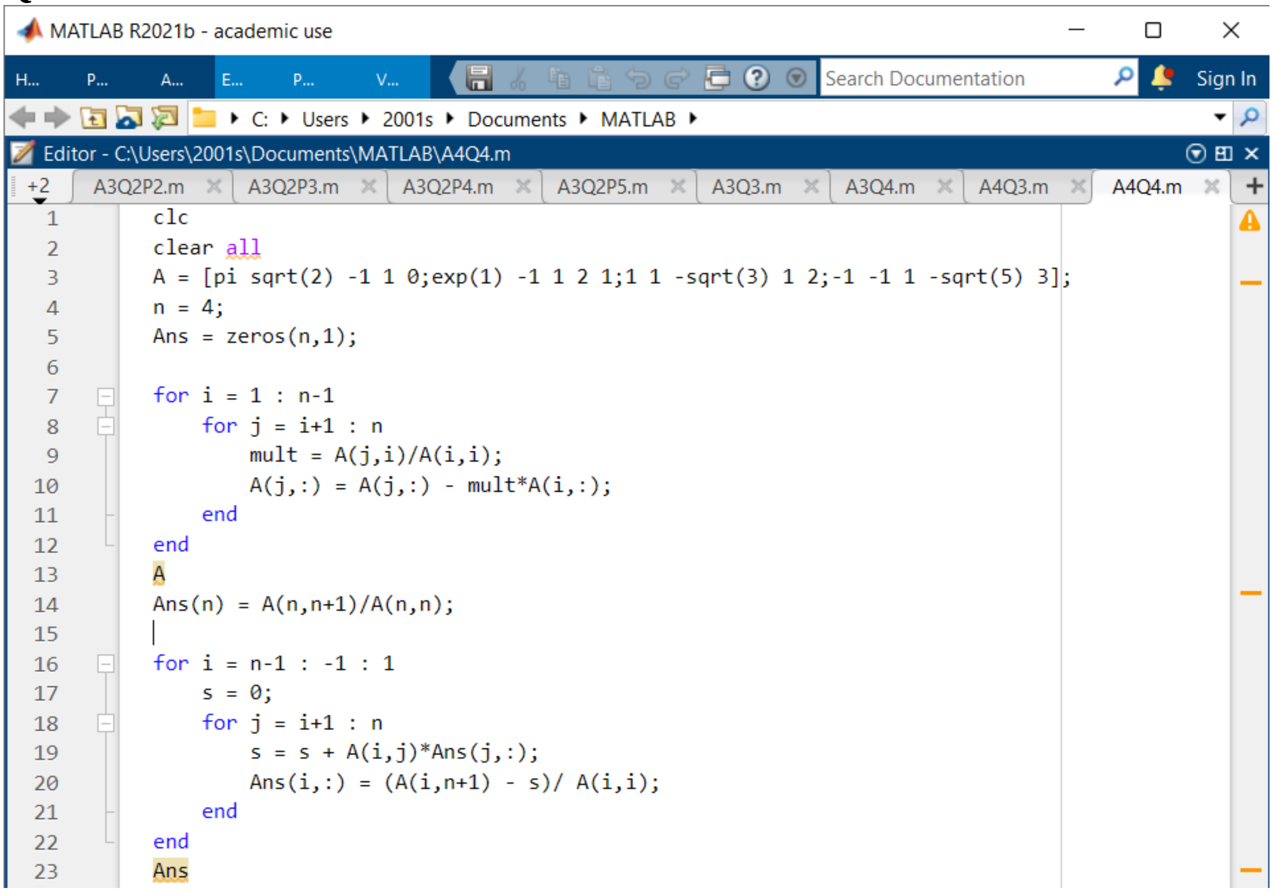
Take $V = 5.5$ and solve the system.

Q3: **Ans = 1.0000 1.0000 1.0000 1.0000**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A4Q3.m
A3Q2P2.m
A3Q2P3.m
A3Q2P4.m
A3Q2P5.m
A3Q3.m
A3Q4.m
A4Q3.m

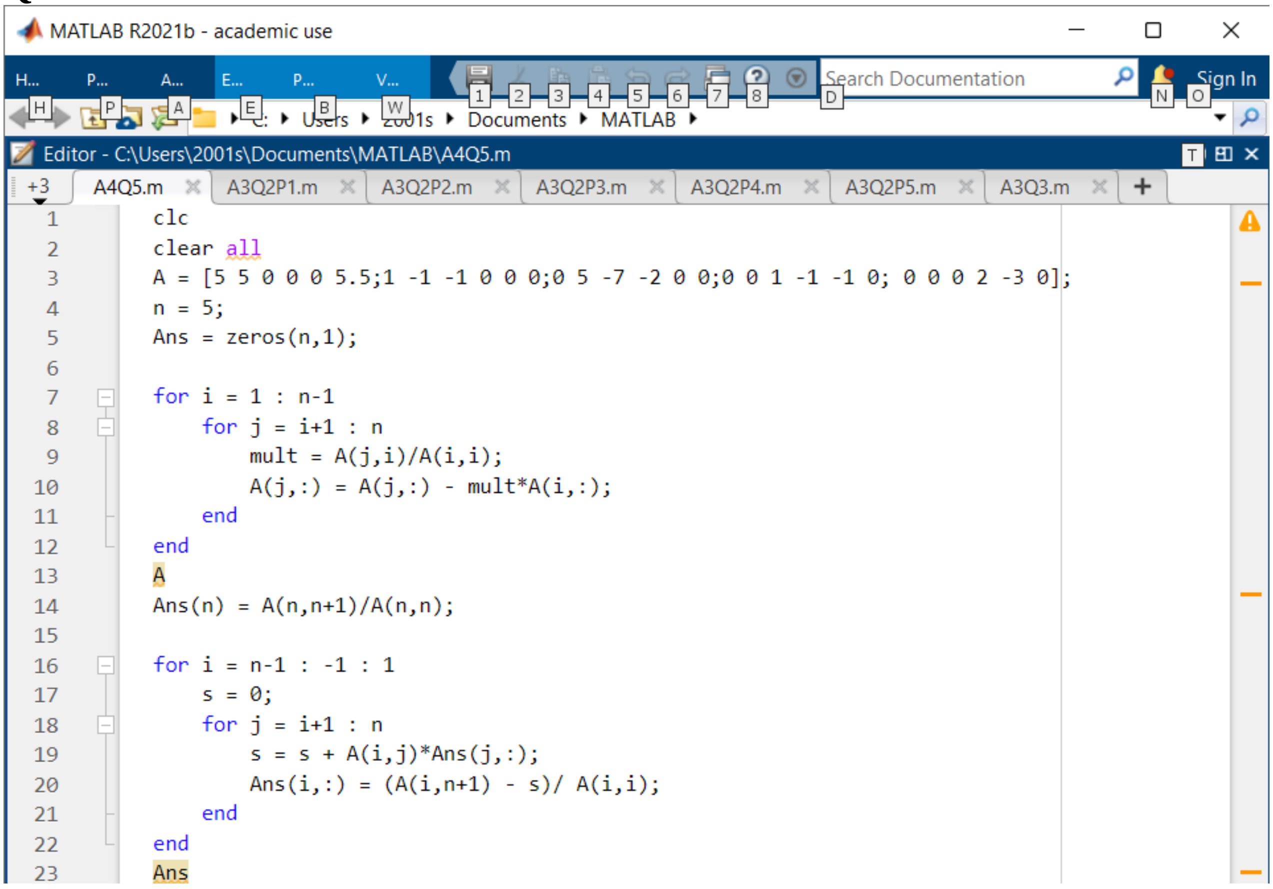
```
clc
clear all
A = [10 8 -3 1 16;2 10 1 -4 9;3 -4 10 1 10;2 2 -3 10 11];
n = 4;
Ans = zeros(n,1);

for i = 1 : n-1
    for j = i+1 : n
        mult = A(j,i)/A(i,i);
        A(j,:) = A(j,:) - mult*A(i,:);
    end
end
A
Ans(n) = A(n,n+1)/A(n,n);

for i = n-1 : -1 : 1
    s = 0;
    for j = i+1 : n
        s = s + A(i,j)*Ans(j,:);
        Ans(i,:) = (A(i,n+1) - s)/ A(i,i);
    end
end
Ans
```

Q4: **Ans = 1.3494 -4.6780 -4.0329 -1.6566**



```
clc
clear all
A = [pi sqrt(2) -1 1 0;exp(1) -1 1 2 1;1 1 -sqrt(3) 1 2;-1 -1 1 -sqrt(5) 3];
n = 4;
Ans = zeros(n,1);

for i = 1 : n-1
    for j = i+1 : n
        mult = A(j,i)/A(i,i);
        A(j,:) = A(j,:) - mult*A(i,:);
    end
end
A
Ans(n) = A(n,n+1)/A(n,n);

for i = n-1 : -1 : 1
    s = 0;
    for j = i+1 : n
        s = s + A(i,j)*Ans(j,:);
        Ans(i,:) = (A(i,n+1) - s)/ A(i,i);
    end
end
Ans
```

Q5: **Ans = 0.6785 0.4215 0.2570 0.1542 0.1028**



```
clc
clear all
A = [5 5 0 0 0 5.5;1 -1 -1 0 0 0;0 5 -7 -2 0 0;0 0 1 -1 -1 0; 0 0 0 2 -3 0];
n = 5;
Ans = zeros(n,1);

for i = 1 : n-1
    for j = i+1 : n
        mult = A(j,i)/A(i,i);
        A(j,:) = A(j,:) - mult*A(i,:);
    end
end
A
Ans(n) = A(n,n+1)/A(n,n);

for i = n-1 : -1 : 1
    s = 0;
    for j = i+1 : n
        s = s + A(i,j)*Ans(j,:);
        Ans(i,:) = (A(i,n+1) - s)/ A(i,i);
    end
end
Ans
```

Experiment 5: Gauss-Seidel and SOR Methods

1. **Algorithm for Gauss Seidel Method:** Find a solution of system of linear equations Ax = b.

   Input: Number of unknowns n; Coefficient matrix $A = a_{ij}$, where $1 \le i \le n$, and $1 \le j \le n$; column vector b; Initial solution vector $x0$; tolerance value tol; maximum number of iterations N.

   Output: Solution $(x_1, x_2, \cdots, x_n)$.

```
Step 1: For $k = 1,2,\cdots,N$ do Steps 2 – 4.
Step 2: For $i = 1,2,\cdots,n$
            $x_i = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} x0_j \right)$
Step 3: If $\| x - x0 \| < tol$ then OUTPUT $(x_1, x_2, \cdots, x_n)$.
            STOP
Step 4: Set $x0 = x$. (Update $x0$)
Step 5: Print OUTPUT $(x_1, x_2, \cdots, x_n)$ (Procedure completed successfully)
            STOP.
```

2. Write an algorithm for Successive-Over-Relaxation (SOR) method.

3. Use Gauss Seidel method and SOR method with w = 1.2 to find the solution of the following linear systems with an initial vector [0,0,0,0] and tolerance value 10^{-3} in the $\|.\|_\infty$ norm:

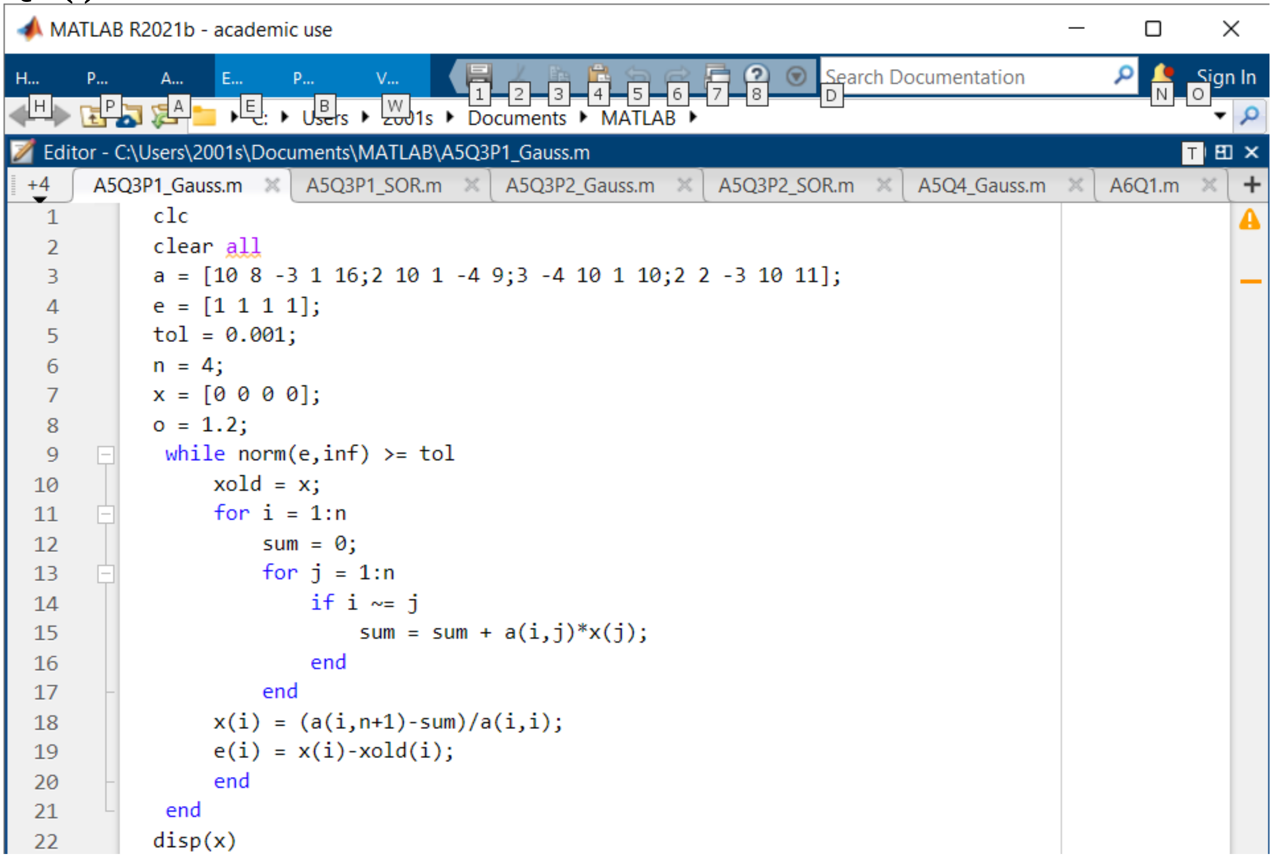
(a)
$$\begin{aligned} 10x + 8y - 3z + u &= 16 \\ 2x + 10y + z - 4u &= 9 \\ 3x - 4y + 10z + u &= 10 \\ 2x + 2y - 3z + 10u &= 11 \end{aligned}$$

(b)
$$\begin{aligned} 4x_1 + x_2 - x_3 + x_4 &= -2 \\ x_1 + 4x_2 - x_3 - x_4 &= -1 \\ -x_1 - x_2 + 5x_3 + x_4 &= 0 \\ x_1 - x_2 + x_3 + 3x_4 &= 1 \end{aligned}$$

4. Use Gauss Seidel method to solve the following linear system with an initial vector [0,0,0] and tolerance value 10^{-3} in the $\|.\|_\infty$ norm:

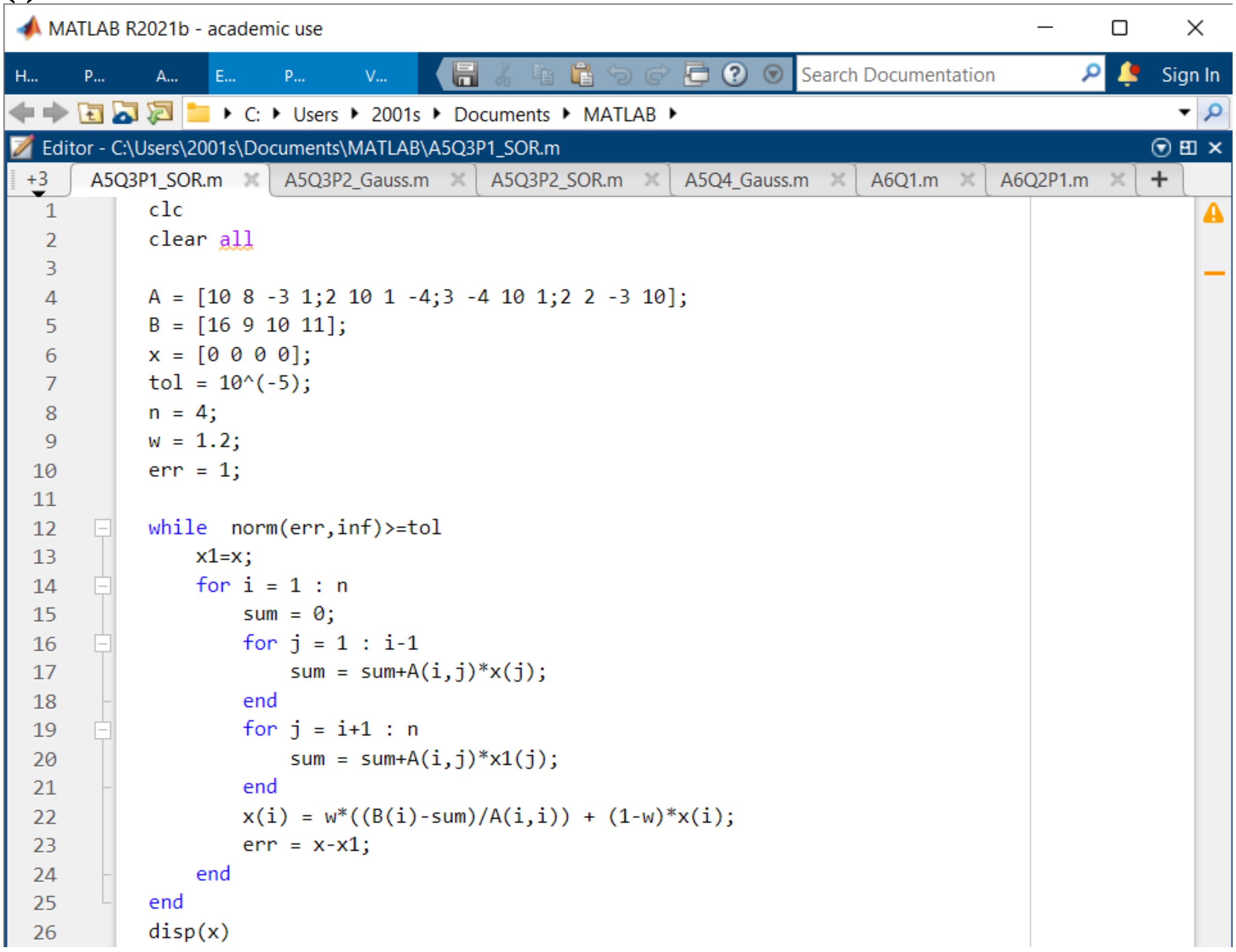
$$\begin{aligned} 4.63x_1 - 1.21x_2 + 3.22\,x_3 &= 2.22 \\ -3.07x_1 + 5.48x_2 + 2.11x_3 &= -3.17 \\ 1.26x_1 + 3.11x_2 + 4.57x_3 &= 5.11 \end{aligned}$$

Q3 (i): Gauss-Seidel 1.0001 0.9999 1.0000 1.0000



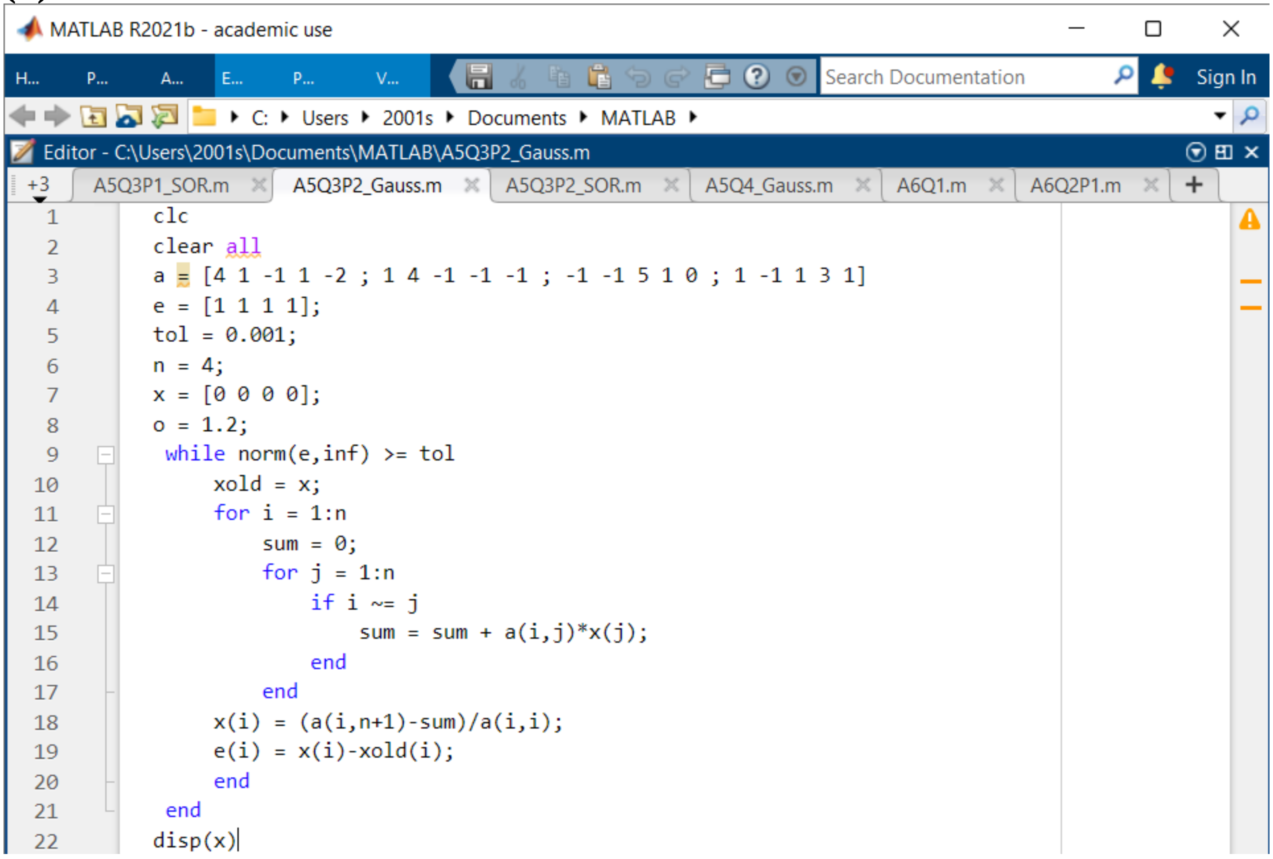
```
clc
clear all
a = [10 8 -3 1 16;2 10 1 -4 9;3 -4 10 1 10;2 2 -3 10 11];
e = [1 1 1 1];
tol = 0.001;
n = 4;
x = [0 0 0 0];
o = 1.2;
 while norm(e,inf) >= tol
     xold = x;
     for i = 1:n
         sum = 0;
         for j = 1:n
             if i ~= j
                 sum = sum + a(i,j)*x(j);
             end
         end
     x(i) = (a(i,n+1)-sum)/a(i,i);
     e(i) = x(i)-xold(i);
     end
 end
disp(x)
```

(i): SOR 1.0000 1.0000 1.0000 1.0000



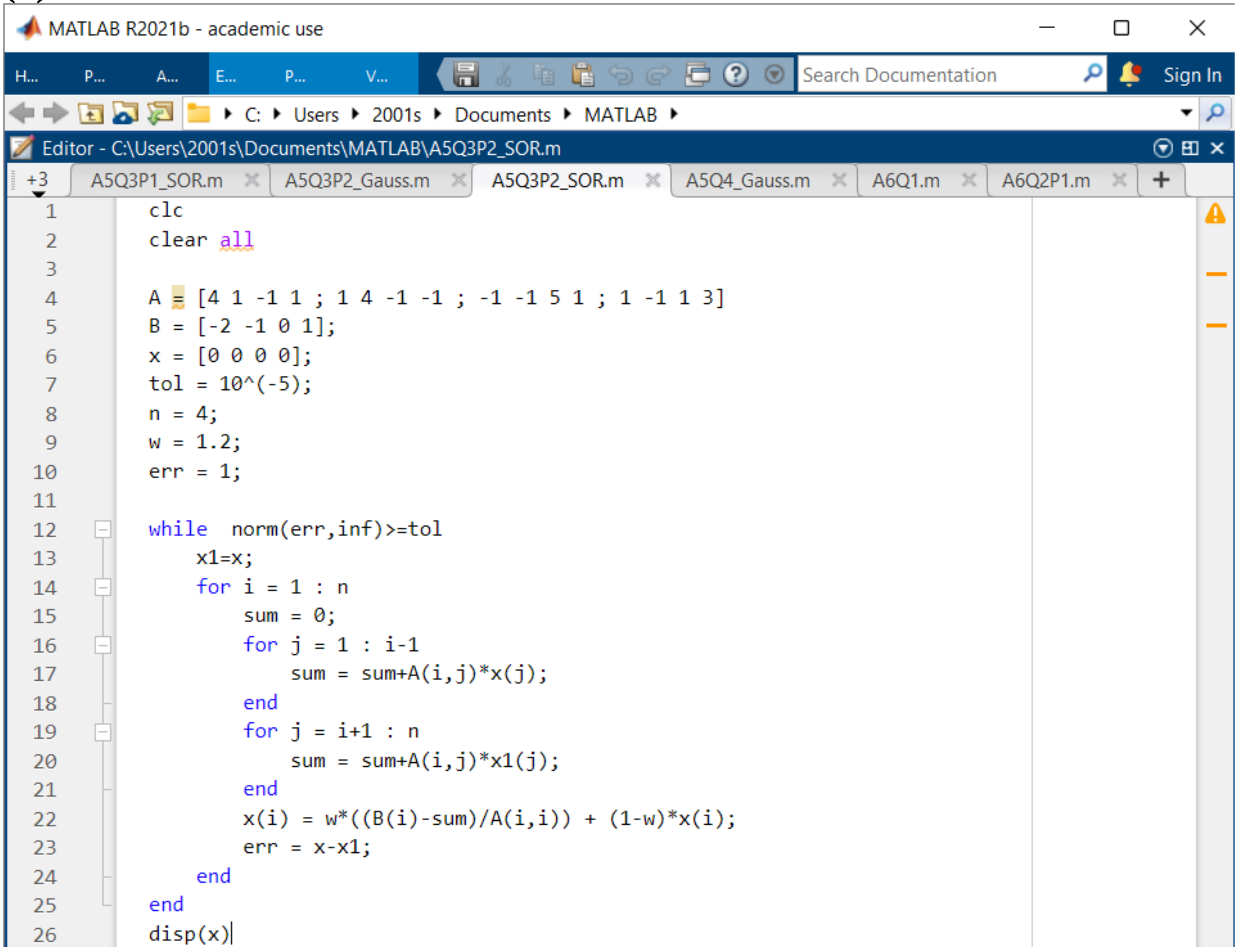
```
clc
clear all

A = [10 8 -3 1;2 10 1 -4;3 -4 10 1;2 2 -3 10];
B = [16 9 10 11];
x = [0 0 0 0];
tol = 10^(-5);
n = 4;
w = 1.2;
err = 1;

while  norm(err,inf)>=tol
    x1=x;
    for i = 1 : n
        sum = 0;
        for j = 1 : i-1
            sum = sum+A(i,j)*x(j);
        end
        for j = i+1 : n
            sum = sum+A(i,j)*x1(j);
        end
        x(i) = w*((B(i)-sum)/A(i,i)) + (1-w)*x(i);
        err = x-x1;
    end
end
disp(x)
```

(ii): Gauss-Seidel **-0.7532 0.0410 -0.2807 0.6916**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A5Q3P2_Gauss.m
A5Q3P1_SOR.m
A5Q3P2_Gauss.m
A5Q3P2_SOR.m
A5Q4_Gauss.m
A6Q1.m
A6Q2P1.m

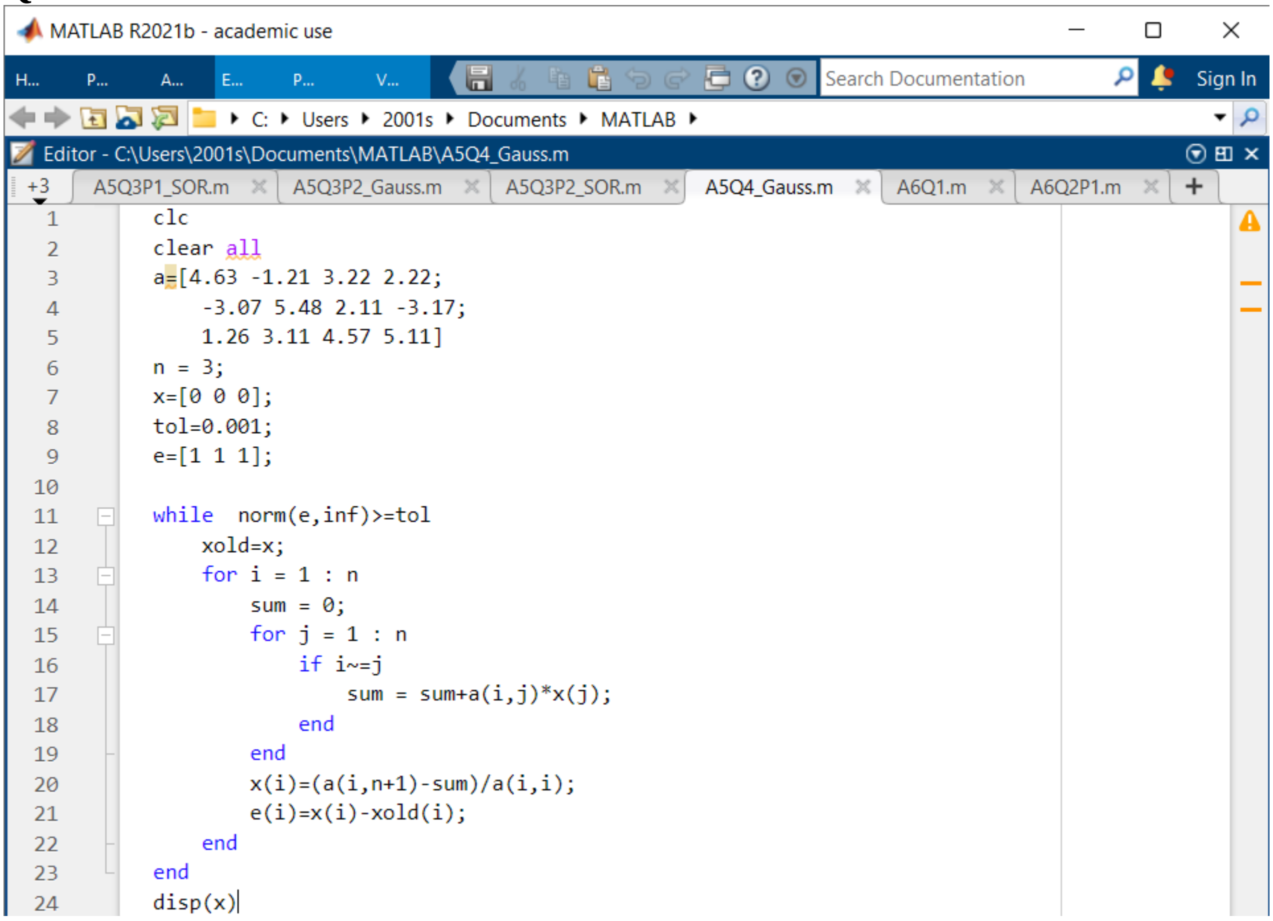
```
clc
clear all
a = [4 1 -1 1 -2 ; 1 4 -1 -1 -1 ; -1 -1 5 1 0 ; 1 -1 1 3 1]
e = [1 1 1 1];
tol = 0.001;
n = 4;
x = [0 0 0 0];
o = 1.2;
 while norm(e,inf) >= tol
     xold = x;
     for i = 1:n
         sum = 0;
         for j = 1:n
             if i ~= j
                 sum = sum + a(i,j)*x(j);
             end
         end
     x(i) = (a(i,n+1)-sum)/a(i,i);
     e(i) = x(i)-xold(i);
     end
 end
disp(x)
```

(ii): SOR **-0.7534 0.0411 -0.2808 0.6918**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A5Q3P2_SOR.m
A5Q3P1_SOR.m
A5Q3P2_Gauss.m
A5Q3P2_SOR.m
A5Q4_Gauss.m
A6Q1.m
A6Q2P1.m

```
clc
clear all

A = [4 1 -1 1 ; 1 4 -1 -1 ; -1 -1 5 1 ; 1 -1 1 3]
B = [-2 -1 0 1];
x = [0 0 0 0];
tol = 10^(-5);
n = 4;
w = 1.2;
err = 1;

while  norm(err,inf)>=tol
    x1=x;
    for i = 1 : n
        sum = 0;
        for j = 1 : i-1
            sum = sum+A(i,j)*x(j);
        end
        for j = i+1 : n
            sum = sum+A(i,j)*x1(j);
        end
        x(i) = w*((B(i)-sum)/A(i,i)) + (1-w)*x(i);
        err = x-x1;
    end
end
disp(x)
```

Q4: Gauss-Seidel **-8.9807 -9.4762 10.0430**


MATLAB R2021b - academic use
H... P... A... E... P... V...
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A5Q4_Gauss.m
+3 A5Q3P1_SOR.m A5Q3P2_Gauss.m A5Q3P2_SOR.m A5Q4_Gauss.m A6Q1.m A6Q2P1.m

```
clc
clear all
a=[4.63 -1.21 3.22 2.22;
    -3.07 5.48 2.11 -3.17;
    1.26 3.11 4.57 5.11]
n = 3;
x=[0 0 0];
tol=0.001;
e=[1 1 1];

while  norm(e,inf)>=tol
    xold=x;
    for i = 1 : n
        sum = 0;
        for j = 1 : n
            if i~=j
                sum = sum+a(i,j)*x(j);
            end
        end
        x(i)=(a(i,n+1)-sum)/a(i,i);
        e(i)=x(i)-xold(i);
    end
end
disp(x)
```

Experiment 6: Power Method and Lagrange Interpolation

1. **Algorithm for Power method:**
   Step 1: START
   Step 2: Define matrix A and initial guess *x*.
   Step 3: Calculate *y* = A*x*
   Step 4: Find the largest element in magnitude of matrix y and assign to *K*.
   Step 5: Calculate fresh value *x* = (1/*K*) * y.
   Step 6: If $K\ n - K\ n - 1$ > error, goto Setp
   3. Step 7: STOP.

2. Determine the largest eigen-value and the corresponding eigen-vector of the following matrices using the power method. Use x_0 = [1,1,1]T and ϵ = 10^{-3}:

   (a) $\begin{matrix} 4 & 1 & 0 \\ 1 & 20 & 1 \\ 0 & 1 & 4 \end{matrix}$. Use x_0 = [1,1,1]T and ϵ = 10^{-3}

   (b) $\begin{matrix} 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 \\ 0 & 0 & 3 & 3 \\ 0 & 1 & 2 & 3 \end{matrix}$. Use x_0 = [1,1,0,1]T and ϵ = 10^{-3}

3. **Algorithm for Lagrange interpolation:** Given a set of function values

x	x_1	x_2	……	x_n
$f(x)$	$f(x_1)$	$f(x_2)$	……	$f(x_n)$

To approximate the value of a function *f*(*x*) at *x* = *p* using Lagrange's interpolating polynomial $P_{n-1}\ x$ of degree ≤ *n* – 1, given by

$$P_{n-1}\ x\ = l_1(x)\, f(x_1) + l_2(x)\, f(x_2) + \ldots .... + l_n(x)\, f(x_n)$$

where $l_i\ x = \prod_{j=1;\, i\neq j}^{n} \frac{(p-x_j)}{(x_i-x_j)}$ at *x* = *p*.

We write the following algorithm by taking *n* points and thus we will obtain a polynomial of degree ≤ *n* – 1.

Input: The degree of the polynomial, the values *x*(*i*) and *f*(*i*), *i* = 1, 2,….,*n* and the point of interpolation *p*.

Output: Value of $P_{n-1}\ p$.

Algorithm:

Step 1. Calculate the Lagrange's fundamental polynomials $l_i(x)$ using the following loop:

```
for $i$ = 1 to $n$
$l(i)$ = 1
for $j$ = 1 to $n$
if $j \neq i$
$l\ i\ = \frac{p - x_j}{x\ i - x\ j} l\ i$

end $j$
end $i$
```

Step 2. Calculate the approximate value of the function at x=p using the following loop:

```
sum = 0
for $i$ = 1 to $n$
sum = sum + $l(i)*f(i)$
end $i$
```

Step 3. Print sum.

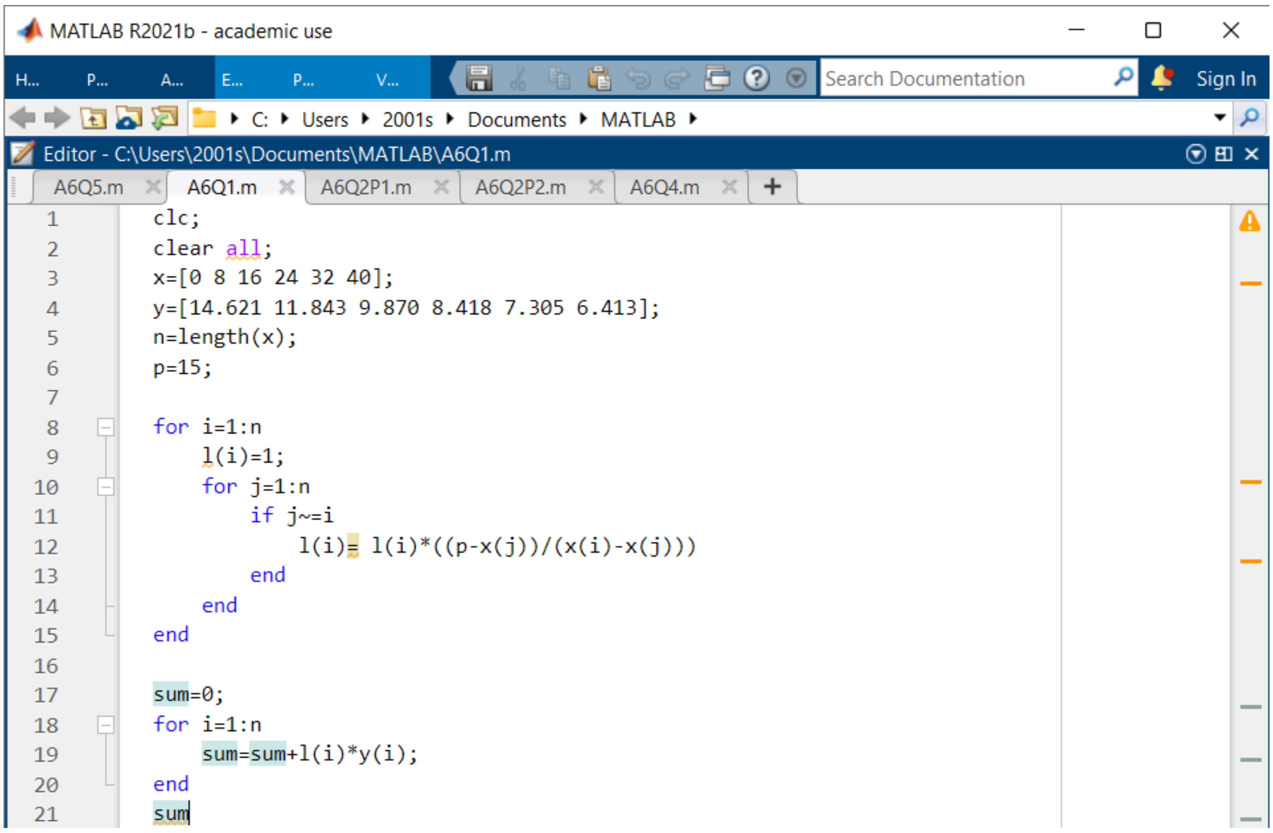
4. The following data define the sea-level concentration of dissolved oxygen for fresh water as a function of temperature:

t	0	8	16	24	32	40
$O(t)$	14.621	11.843	9.870	8.418	7.305	6.413

Use Lagrange's interpolation formula to approximate the value of $O(15)$ and $O(27)$.

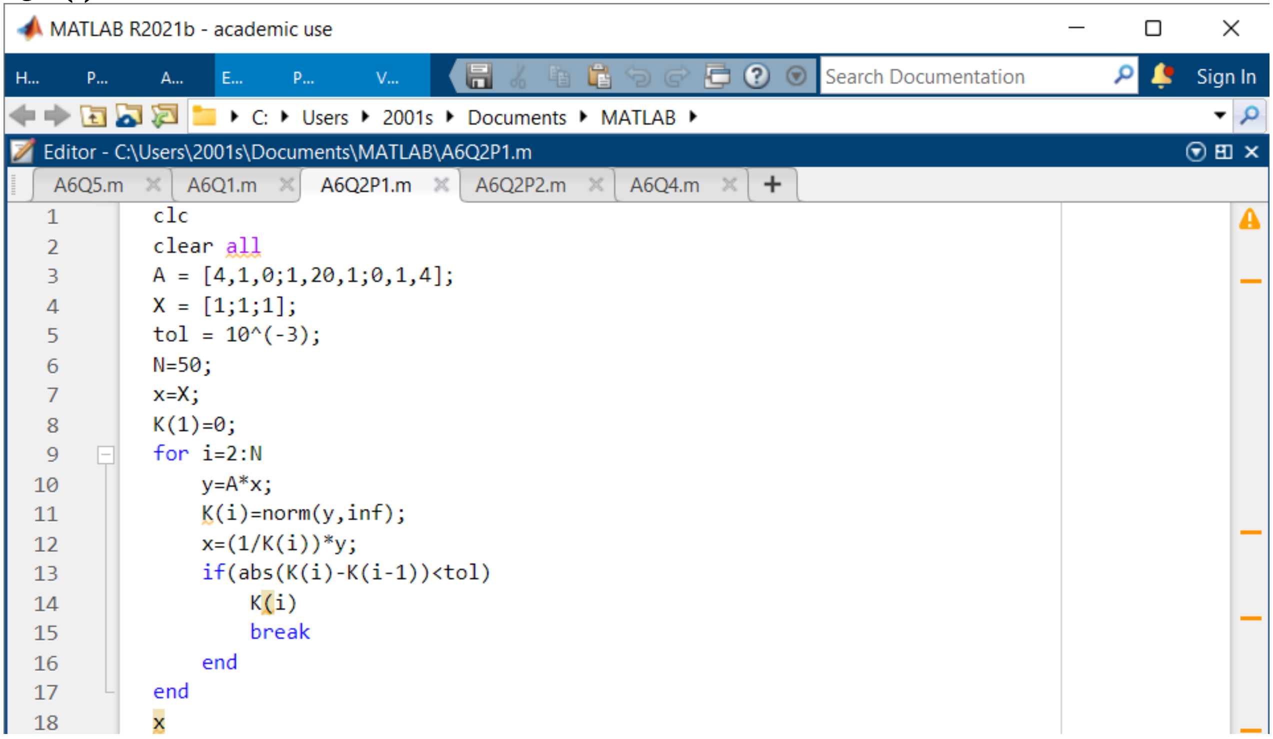
5. Generate eight equally-spaced points from the function $f(x) = \sin^2 x$ from $x = 0$ to 2π. Use Lagrange interpolation to approximate $f(0.5)$, $f(3.5)$, $f(5.5)$ and $f(6.0)$.

Q1: Power **Sum = 10.0834**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A6Q1.m
A6Q5.m
A6Q1.m
A6Q2P1.m
A6Q2P2.m
A6Q4.m

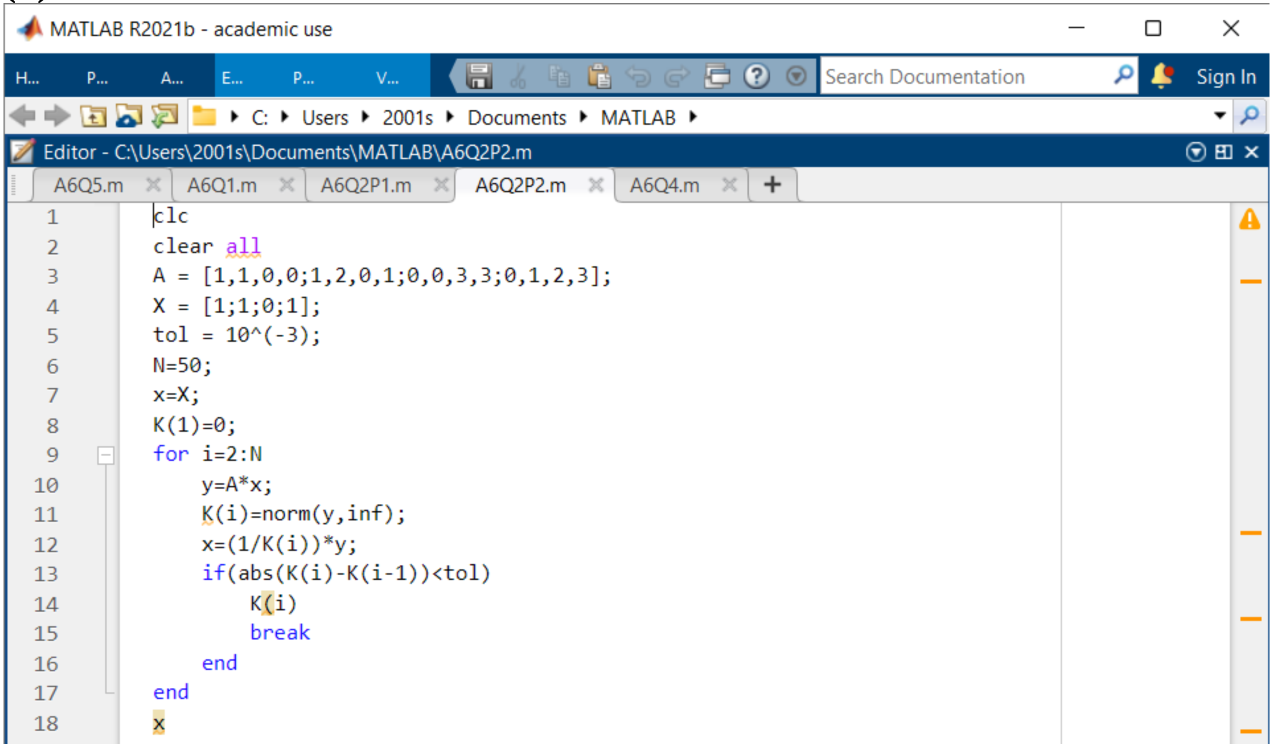
```
clc;
clear all;
x=[0 8 16 24 32 40];
y=[14.621 11.843 9.870 8.418 7.305 6.413];
n=length(x);
p=15;

for i=1:n
    l(i)=1;
    for j=1:n
        if j~=i
            l(i)= l(i)*((p-x(j))/(x(i)-x(j)))
        end
    end
end

sum=0;
for i=1:n
    sum=sum+l(i)*y(i);
end
sum
```

Q2 (i): Power x = 0.0620 1.0000 0.0620


MATLAB R2021b - academic use
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A6Q2P1.m
A6Q5.m
A6Q1.m
A6Q2P1.m
A6Q2P2.m
A6Q4.m

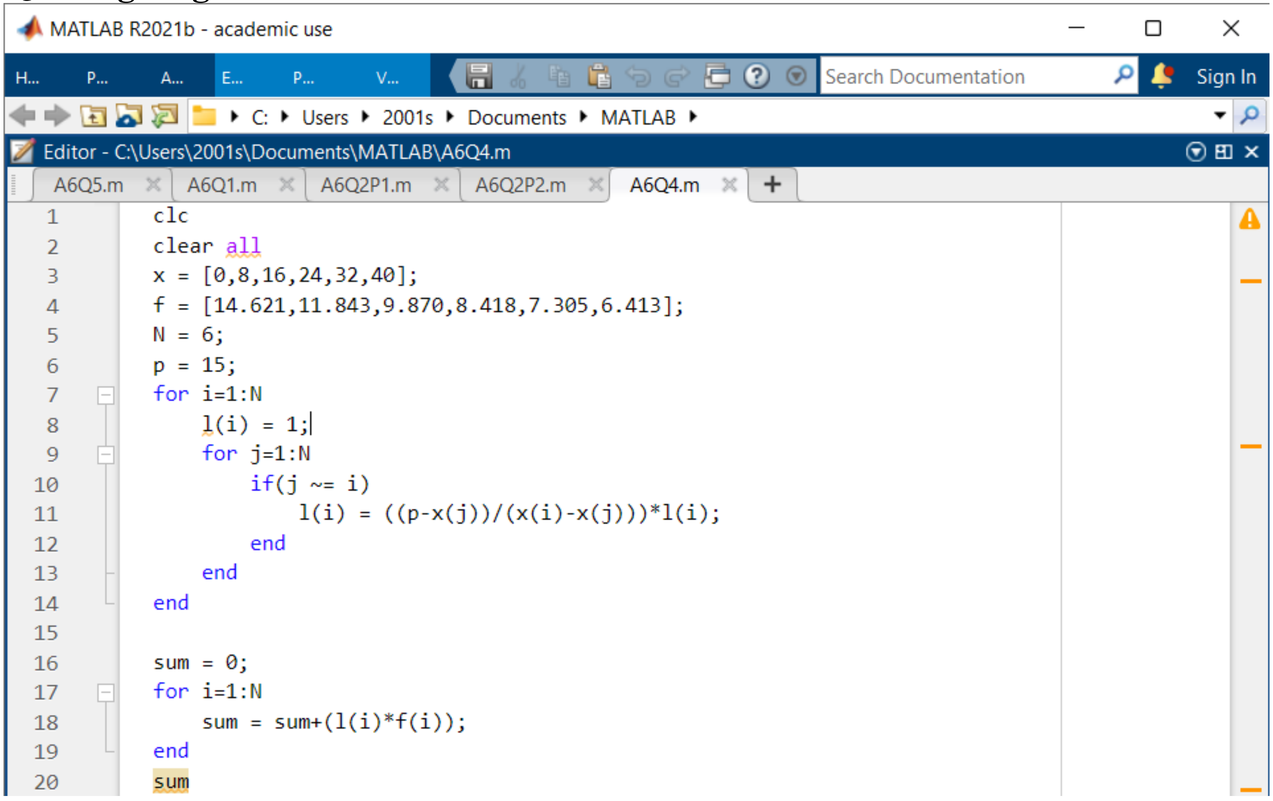
```
clc
clear all
A = [4,1,0;1,20,1;0,1,4];
X = [1;1;1];
tol = 10^(-3);
N=50;
x=X;
K(1)=0;
for i=2:N
    y=A*x;
    K(i)=norm(y,inf);
    x=(1/K(i))*y;
    if(abs(K(i)-K(i-1))<tol)
        K(i)
        break
    end
end
x
```

(ii): Power x = 0.0558 0.2564 1.0000 0.8673


MATLAB R2021b - academic use
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A6Q2P2.m
A6Q5.m
A6Q1.m
A6Q2P1.m
A6Q2P2.m
A6Q4.m

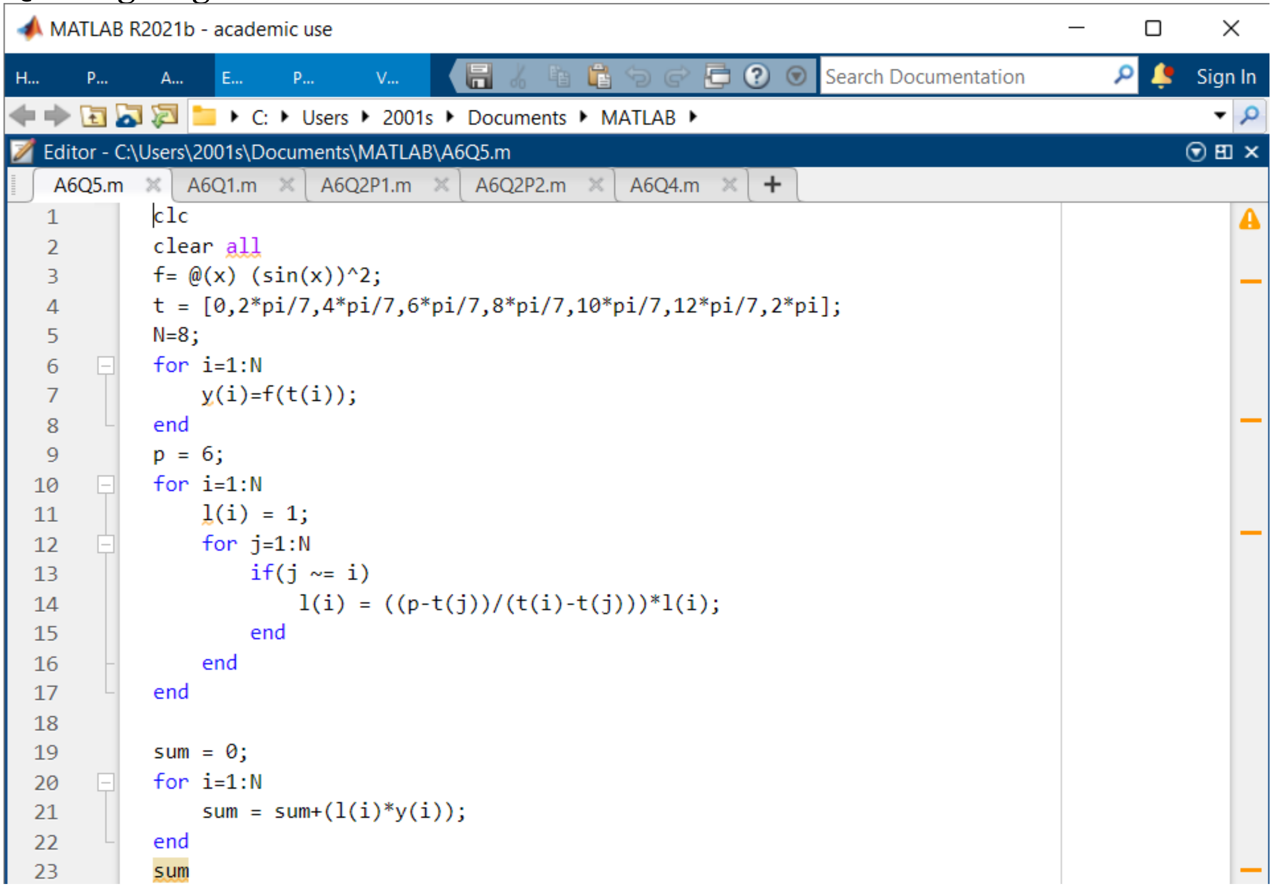
```
clc
clear all
A = [1,1,0,0;1,2,0,1;0,0,3,3;0,1,2,3];
X = [1;1;0;1];
tol = 10^(-3);
N=50;
x=X;
K(1)=0;
for i=2:N
    y=A*x;
    K(i)=norm(y,inf);
    x=(1/K(i))*y;
    if(abs(K(i)-K(i-1))<tol)
        K(i)
        break
    end
end
x
```

Q4: Lagrange's **Sum = 10.0834**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB ▸
Editor - C:\Users\2001s\Documents\MATLAB\A6Q4.m
A6Q5.m
A6Q1.m
A6Q2P1.m
A6Q2P2.m
A6Q4.m

```
clc
clear all
x = [0,8,16,24,32,40];
f = [14.621,11.843,9.870,8.418,7.305,6.413];
N = 6;
p = 15;
for i=1:N
    l(i) = 1;
    for j=1:N
        if(j ~= i)
            l(i) = ((p-x(j))/(x(i)-x(j)))*l(i);
        end
    end
end

sum = 0;
for i=1:N
    sum = sum+(l(i)*f(i));
end
sum
```

Q5: Lagrange's **Sum = -0.1810**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB ▸
Editor - C:\Users\2001s\Documents\MATLAB\A6Q5.m
A6Q5.m
A6Q1.m
A6Q2P1.m
A6Q2P2.m
A6Q4.m

```
clc
clear all
f= @(x) (sin(x))^2;
t = [0,2*pi/7,4*pi/7,6*pi/7,8*pi/7,10*pi/7,12*pi/7,2*pi];
N=8;
for i=1:N
    y(i)=f(t(i));
end
p = 6;
for i=1:N
    l(i) = 1;
    for j=1:N
        if(j ~= i)
            l(i) = ((p-t(j))/(t(i)-t(j)))*l(i);
        end
    end
end

sum = 0;
for i=1:N
    sum = sum+(l(i)*y(i));
end
sum
```

Experiment 7: Newton's Divided Difference Interpolation

1. **Algorithm for Newton's divided difference interpolation:**

Given n distinct numbers $x_1, x_2, \dots, x_n$ and their corresponding function values $f(x_1), f(x_2), \dots, f(x_n)$.
Approximate the value of a function $f(x)$ at $x = p$ using Newton's divided difference interpolating polynomial $P_{n-1}(x)$ of degree $\le n - 1$.

Input: Enter n the number of data points; enter n distinct numbers $x_1, x_2, \dots, x_n$; enter corresponding function values $f(x_1), f(x_2), \dots, f(x_n)$ as $F_{1,1}, F_{1,2}, \dots, F_{1,n}$; enter an interpolating point p.
Output: the numbers $F_{2,2}, F_{3,3}, \dots, F_{n,n}$ such that

$$P_{n-1}(p) = \sum_{i=1}^{n} F_{i,i} \prod_{j=1}^{i-1} (p - x_j)$$

where $F_{k,k}$ is the (k-1)th divided difference $f[x_1, x_2, \dots, x_k]$

```
Step 1: Evaluate $F_{2,2}, F_{3,3}, \dots, F_{n,n}$
      for $i = 2$ to $n$
          for $j = i$ to $n$
              Evaluate $F_{j,i} = \frac{F_{j,i-1} - F_{j-1,i-1}}{x_j - x_{j-i+1}}$.
          end $j$
        end $i$
Step 2: Evaluate $\prod_{j=1}^{i-1}(p - x_j)$ for each $i = 1$ to $n$
        for $i = 1$ to $n$
          Set product ($i$) = 1
              for $j = 1$ to $i - 1$
                  product ($i$) = product ($i$) * $(p - x_j)$
              end $j$
        end $i$
Step 3: Evaluate $P_{n-1}(p)$
        Set Sum = 0
        for $i = 1$ to $n$
           Sum = Sum + ( $F_{i,i}$ * product ($i$))
        end $i$
Step 4: OUTPUT   Sum $\equiv P_{n-1}(p)$
        STOP
```

2. The following data represents the function $f(x) = e^x$.

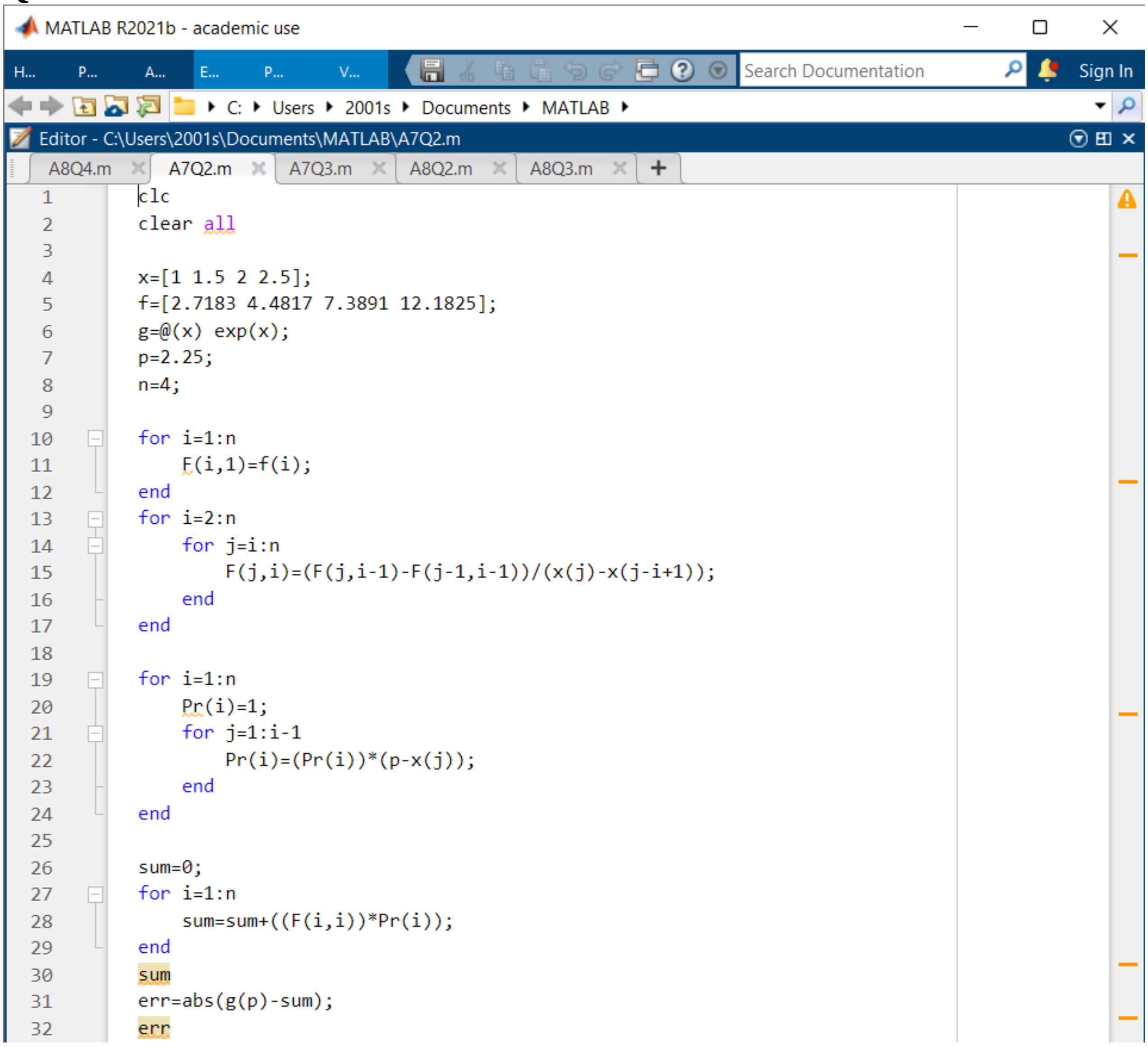
x	1	1.5	2.0	2.5
$f(x)$	2.7183	4.4817	7.3891	12.1825

Estimate the value of $f(2.25)$ using the Newton's divided difference interpolation. Compare with the exact value.

3. Approximate $f(0.43)$ by using Newton's divided difference interpolation, construct the interpolating polynomials for the following data.

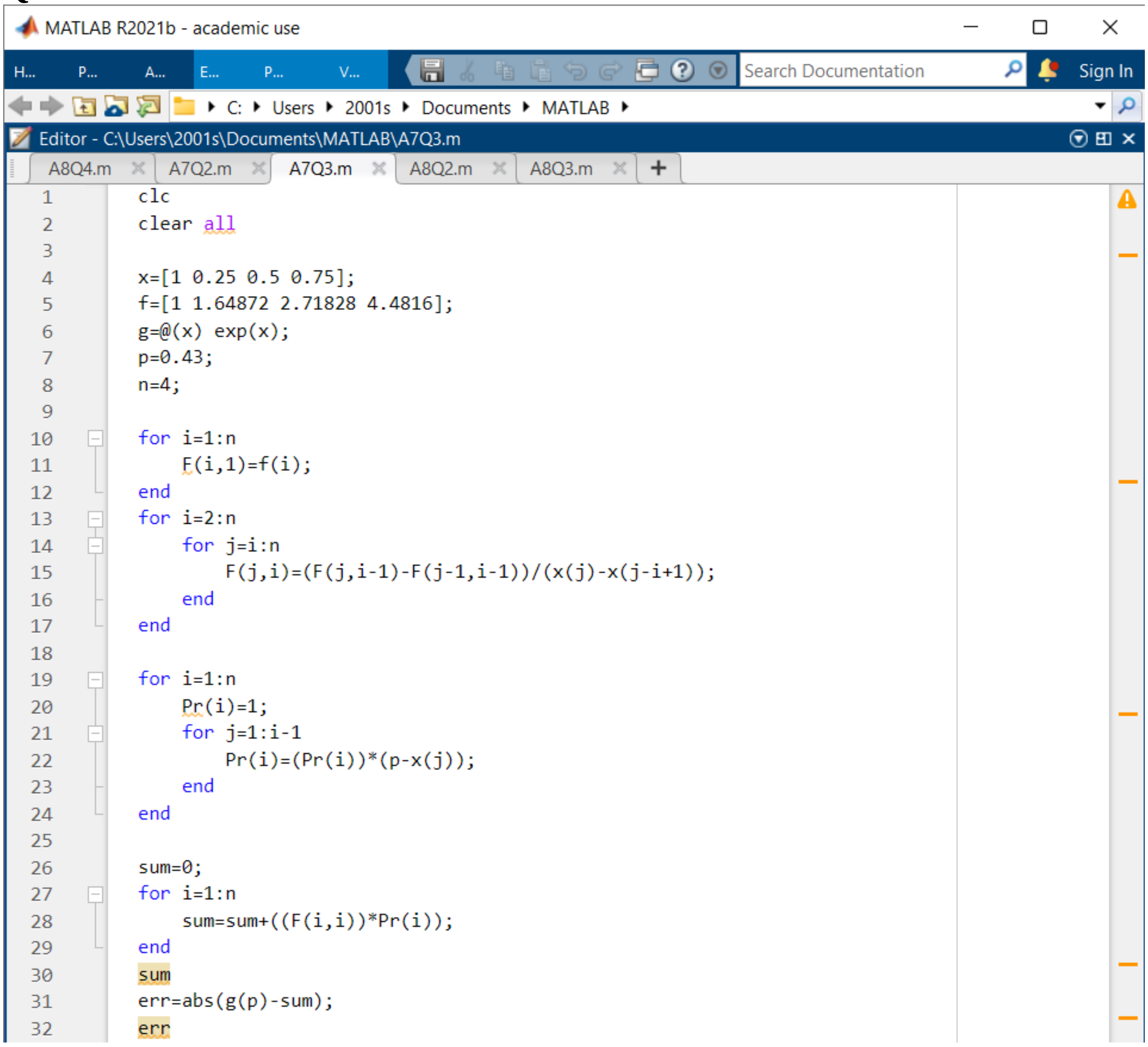
$f(0) = 1, f(0.25) = 1.64872, f(0.5) = 2.71828, f(0.75) = 4.4816.$

Q2: **sum = 9.5037 err = 0.0159**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A7Q2.m
A8Q4.m
A7Q2.m
A7Q3.m
A8Q2.m
A8Q3.m

```
clc
clear all

x=[1 1.5 2 2.5];
f=[2.7183 4.4817 7.3891 12.1825];
g=@(x) exp(x);
p=2.25;
n=4;

for i=1:n
    F(i,1)=f(i);
end
for i=2:n
    for j=i:n
        F(j,i)=(F(j,i-1)-F(j-1,i-1))/(x(j)-x(j-i+1));
    end
end

for i=1:n
    Pr(i)=1;
    for j=1:i-1
        Pr(i)=(Pr(i))*(p-x(j));
    end
end

sum=0;
for i=1:n
    sum=sum+((F(i,i))*Pr(i));
end
sum
err=abs(g(p)-sum);
err
```

Q3: sum = 2.0935 err = 0.5562


MATLAB R2021b - academic use
H... P... A... E... P... V...
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB ▸
Editor - C:\Users\2001s\Documents\MATLAB\A7Q3.m
A8Q4.m
A7Q2.m
A7Q3.m
A8Q2.m
A8Q3.m

```
clc
clear all

x=[1 0.25 0.5 0.75];
f=[1 1.64872 2.71828 4.4816];
g=@(x) exp(x);
p=0.43;
n=4;

for i=1:n
    F(i,1)=f(i);
end
for i=2:n
    for j=i:n
        F(j,i)=(F(j,i-1)-F(j-1,i-1))/(x(j)-x(j-i+1));
    end
end

for i=1:n
    Pr(i)=1;
    for j=1:i-1
        Pr(i)=(Pr(i))*(p-x(j));
    end
end

sum=0;
for i=1:n
    sum=sum+((F(i,i))*Pr(i));
end
sum
err=abs(g(p)-sum);
err
```

Experiment 8: Least Square Approximation

1. **Write an algorithm for least square approximations to fit any curve of the forms:**
   $y = a + bx$, $y = a + bx + cx^2$, $y = Ax + \frac{B}{x}$, …

2. Use the method of least squares to fit the linear and quadratic polynomial to the following data:

x	-2	-1	0	1	2
$f(x)$	15	1	1	3	19

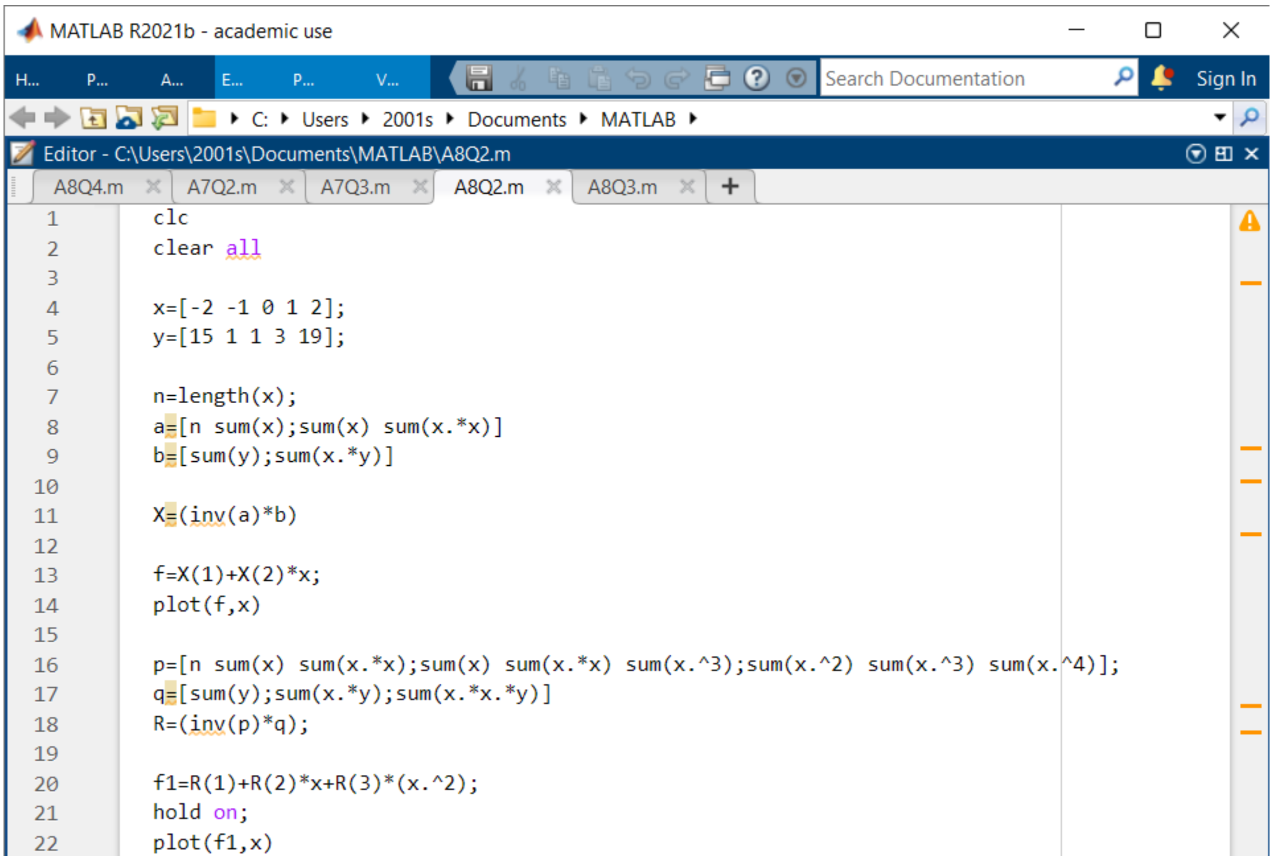
3. By the method of least square fit a curve of the form $y = ax^b$ to the following data:

x	2	3	4	5
y	27.8	62.1	110	161

4. Use the method of least squares to fit a curve $y = Ax + \frac{B}{x}$ to the following data:

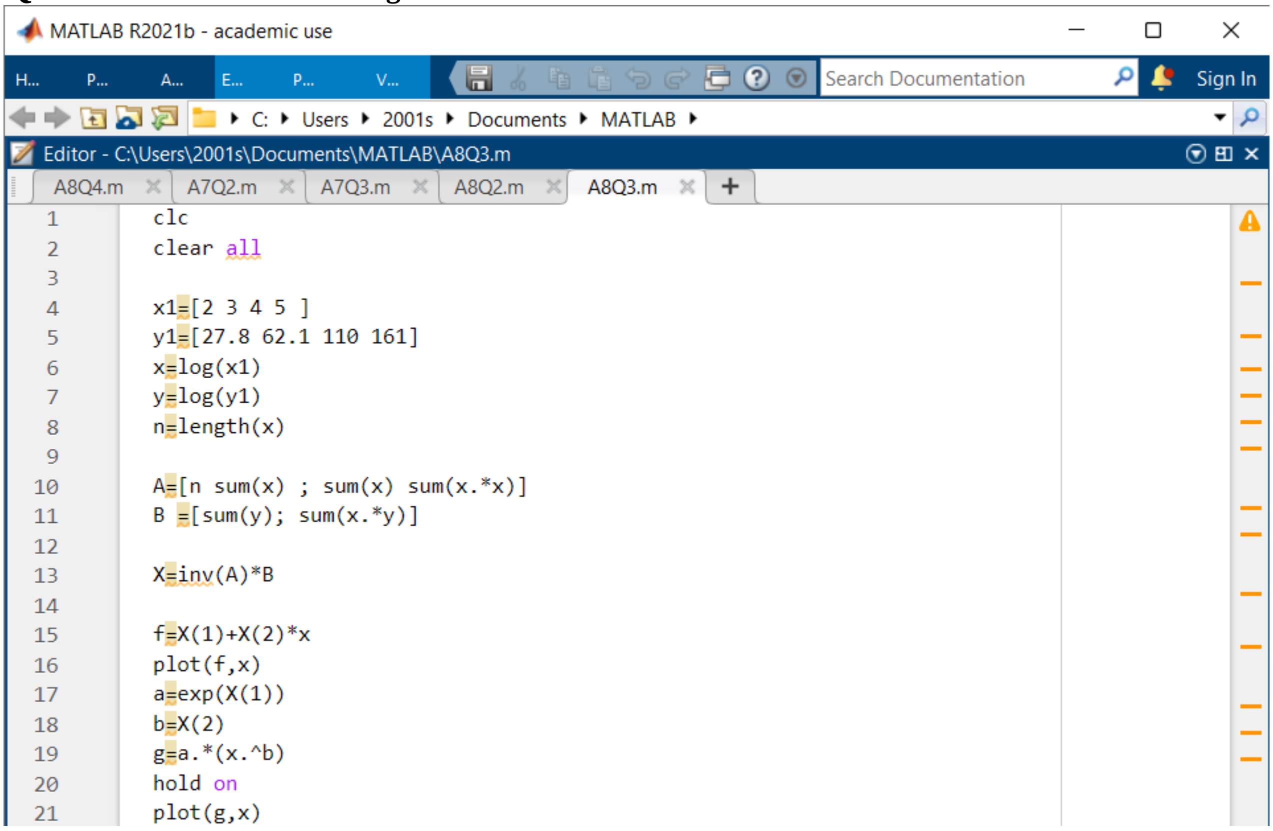
x	0.1	0.2	0.4	0.5	1	2
y	21	11	7	6	5	6

Q2: **a = 5 0 b = 39 X = 7.8000 q = 39 10 140**
0 10 10 1.0000


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A8Q2.m
A8Q4.m
A7Q2.m
A7Q3.m
A8Q2.m
A8Q3.m

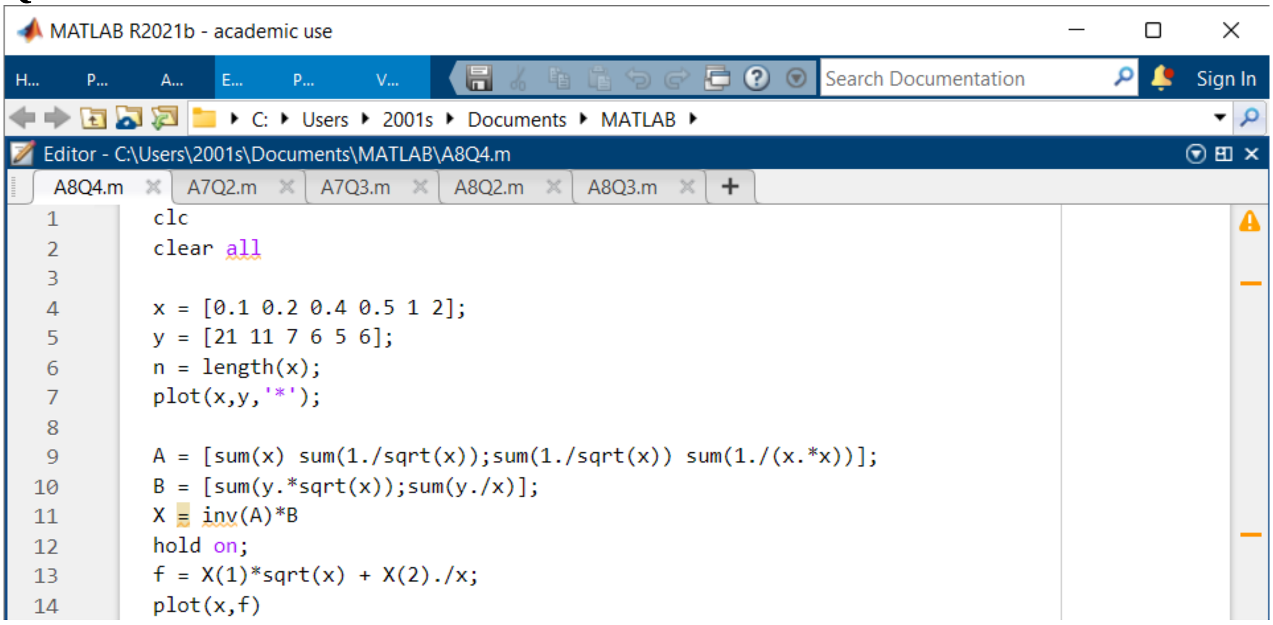
```
clc
clear all

x=[-2 -1 0 1 2];
y=[15 1 1 3 19];

n=length(x);
a=[n sum(x);sum(x) sum(x.*x)]
b=[sum(y);sum(x.*y)]

X=(inv(a)*b)

f=X(1)+X(2)*x;
plot(f,x)

p=[n sum(x) sum(x.*x);sum(x) sum(x.*x) sum(x.^3);sum(x.^2) sum(x.^3) sum(x.^4)];
q=[sum(y);sum(x.*y);sum(x.*x.*y)]
R=(inv(p)*q);

f1=R(1)+R(2)*x+R(3)*(x.^2);
hold on;
plot(f1,x)
```

Q3: **a =7.3799 b =1.9302 g =3.6376 8.8488 13.8629 18.4912 f =3.3366 4.1193 4.6745 5.1052**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A8Q3.m
A8Q4.m A7Q2.m A7Q3.m A8Q2.m A8Q3.m

```
clc
clear all

x1=[2 3 4 5 ]
y1=[27.8 62.1 110 161]
x=log(x1)
y=log(y1)
n=length(x)

A=[n sum(x) ; sum(x) sum(x.*x)]
B =[sum(y); sum(x.*y)]

X=inv(A)*B

f=X(1)+X(2)*x
plot(f,x)
a=exp(X(1))
b=X(2)
g=a.*(x.^b)
hold on
plot(g,x)
```

Q4: **X = 3.2818 1.9733**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A8Q4.m
A8Q4.m A7Q2.m A7Q3.m A8Q2.m A8Q3.m

```
clc
clear all

x = [0.1 0.2 0.4 0.5 1 2];
y = [21 11 7 6 5 6];
n = length(x);
plot(x,y,'*');

A = [sum(x) sum(1./sqrt(x));sum(1./sqrt(x)) sum(1./(x.*x))];
B = [sum(y.*sqrt(x));sum(y./x)];
X = inv(A)*B
hold on;
f = X(1)*sqrt(x) + X(2)./x;
plot(x,f)
```

Experiment 9:Numerical Quadrature

1. **Algorithm for Composite Trapezoidal rule:**

```
Step 1: Input function f(x); end points a and b; and N number of subintervals.
Step 2: Set □ = $\frac{b-a}{N}$.
Step 3: Set sum = 0
Step 4: for i = 1 to N-1
Step 5: Set $x = a + □ * i$
Step 6: Set $sum = sum + 2 * f(x)$
          end i
Step 7: Set $sum = sum + f\ \ a + f(b)$
Step 8: Set $ans = sum * \frac{□}{2}$
          STOP
```

2. **Algorithm for Composite Simpson's rule:**

```
Step 1: Input function f(x); end points a and b; and N number of subintervals (even).
Step 2: Set □ = $\frac{b-a}{N}$.
Step 3: Set sum = 0
Step 4: for i = 1 to N-1
Step 5: Set $x = a + □ * i$
Step 6: if rem(i,2) = = 0
              $sum = sum + 2 * f\ \ x$
           else
              $sum = sum + 4 * f\ \ x$
           end if
           end i
Step 7: Set $sum = sum + f\ \ a + f\ \ b$
Step 8: Set $ans = sum * \frac{□}{3}$
            STOP
```

3. Approximate the following integrals using the composite trapezoidal and Simpson rule by taking different subintervals (e.g. 4, 6, 10, 20)

   (a) $I = {}_{-0.2}^{0.25}(cosx)^2dx$

   (b) $I = {}_{e}^{e+1}\frac{1}{xlnx}dx$

   (c) $I = {}^{1}e_{-}^{-x^2}cosx\ dx$

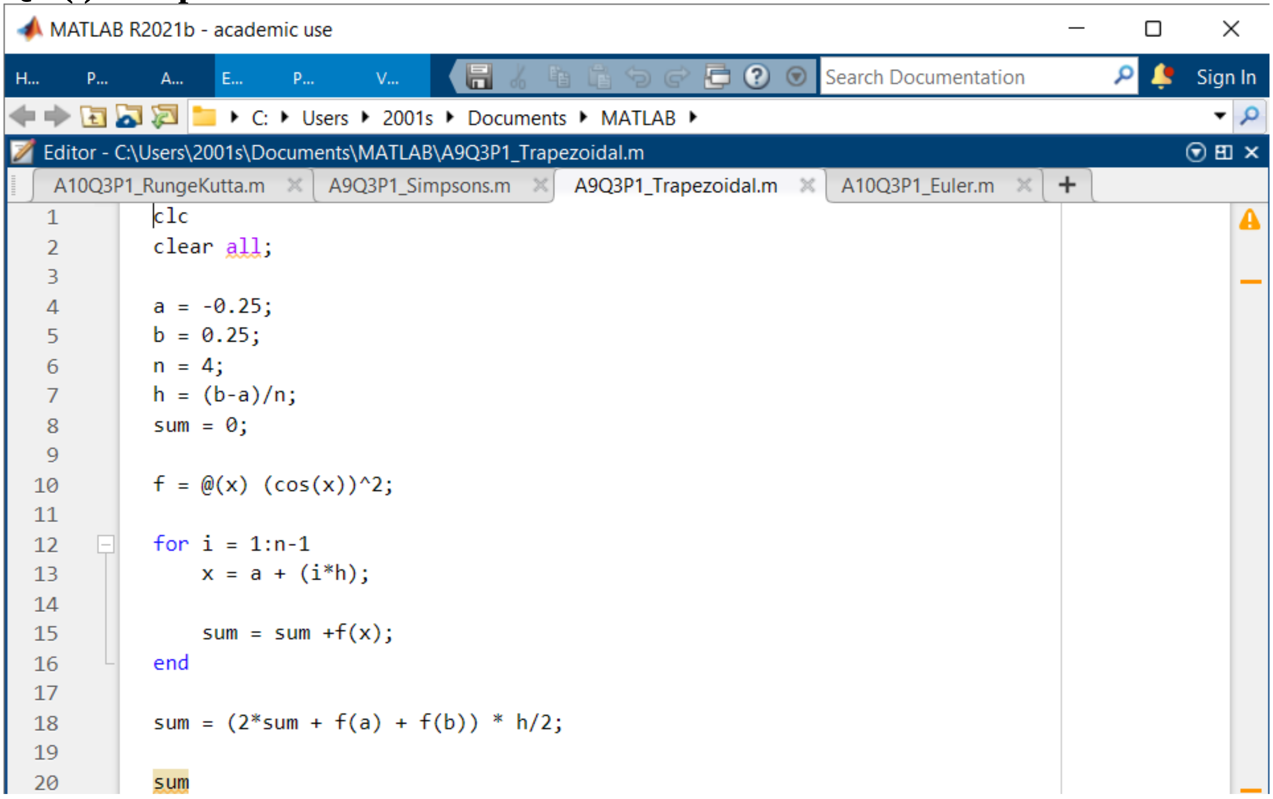
4. The length of the curve represented by a function $y = f(x)$ on an interval $[a, b]$ is given by the integral

$$I = {}_{a}^{b}\ \overline{1 + f'\ \ x\ ^2}dx.$$

Use the trapezoidal rule and Simpson's rule with 4 and 8 subintervals, compute the length of the curve

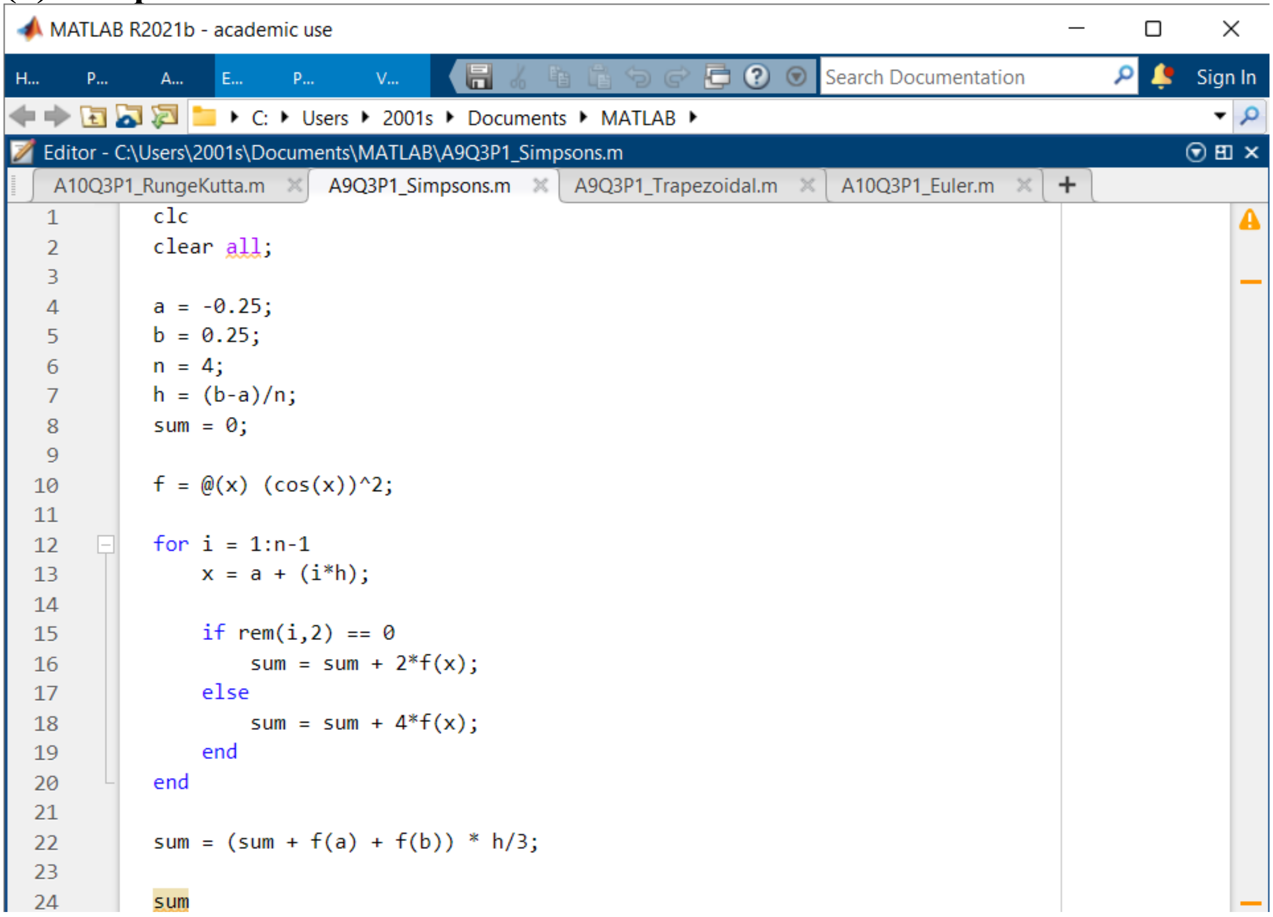
$$y = tan^{-1}\ 1 + x^2\ , 0 \leq x \leq 2.$$

Q3 (i): Trapezoidal **sum = 0.4885**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A9Q3P1_Trapezoidal.m
A10Q3P1_RungeKutta.m
A9Q3P1_Simpsons.m
A9Q3P1_Trapezoidal.m
A10Q3P1_Euler.m

```
clc
clear all;

a = -0.25;
b = 0.25;
n = 4;
h = (b-a)/n;
sum = 0;

f = @(x) (cos(x))^2;

for i = 1:n-1
    x = a + (i*h);

    sum = sum +f(x);
end

sum = (2*sum + f(a) + f(b)) * h/2;

sum
```

(ii): Simpson's **sum = 0.4897**


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A9Q3P1_Simpsons.m
A10Q3P1_RungeKutta.m
A9Q3P1_Simpsons.m
A9Q3P1_Trapezoidal.m
A10Q3P1_Euler.m

```
clc
clear all;

a = -0.25;
b = 0.25;
n = 4;
h = (b-a)/n;
sum = 0;

f = @(x) (cos(x))^2;

for i = 1:n-1
    x = a + (i*h);

    if rem(i,2) == 0
        sum = sum + 2*f(x);
    else
        sum = sum + 4*f(x);
    end
end

sum = (sum + f(a) + f(b)) * h/3;

sum
```

Experiment 10: Solution of Initial Value Problem

1. **Algorithm for Euler Method**

Approximate the solution of the initial value problem $y' = f\ t, y\ a \leq t \leq b, y\ a = \alpha$ in the interval [a, b] with step length h.

Input: function $f\ t, y$; endpoints a, b; step length h; initial condition $t_1 = a$ and $y_1 = \alpha$.

Output: approximation of y in the interval [a, b].

```
Step 1: Evaluate number of sub-intervals $n = (b - a)/\square$.
Step 2: For $i = 1,2, \ldots , n$ do Steps 3 and 4.
Step 3: Evaluate $y_{i+1} = y_i + \square * f\ t_i, y_i$
Step 4: Set $t_{i+1} = t_i + \square$
Step 5: Output $t, y$ .
        STOP
```

2. **Algorithm for Runge-Kutta of fourth-order method:**

Approximate the solution of the initial value problem $y' = f\ t, y\ a \leq t \leq b, y\ a = \alpha$ in the interval [a, b] with step length h.

Input: function $f\ t, y$; endpoints a, b; step length h; initial condition $t_1 = a$ and $y_1 = \alpha$.

Output: approximation of y in the interval [a, b].

```
Step 1: Evaluate number of sub-intervals $n = (b - a)/\square$.
Step 2: For $i = 1,2, \ldots , n$ do Steps 3 to 5.
Step 3: Set $K_1 = \square * f\ t_i , y_i$ ;
            $K_2 = \square * f\ t_i + \frac{\square}{2}, y_i + \frac{K_1}{2};$
            $K_3 = \square * f\ t_i + \frac{\square}{2}, y_i + \frac{K_2}{2};$
            $K_4 = \square * f\ t_i + \square, y_i + K_3$ .
Step 4: Set $y_{i+1} = y_i + \frac{K_1+2K_2+2K_3+K_4}{6}$   $(Compute\ y_{i+1})$
Step 5: Set $t_{i+1} = t_i + \square$. $(Compute\ t_i)$
Step 6: Output $t, y$ .
        STOP
```

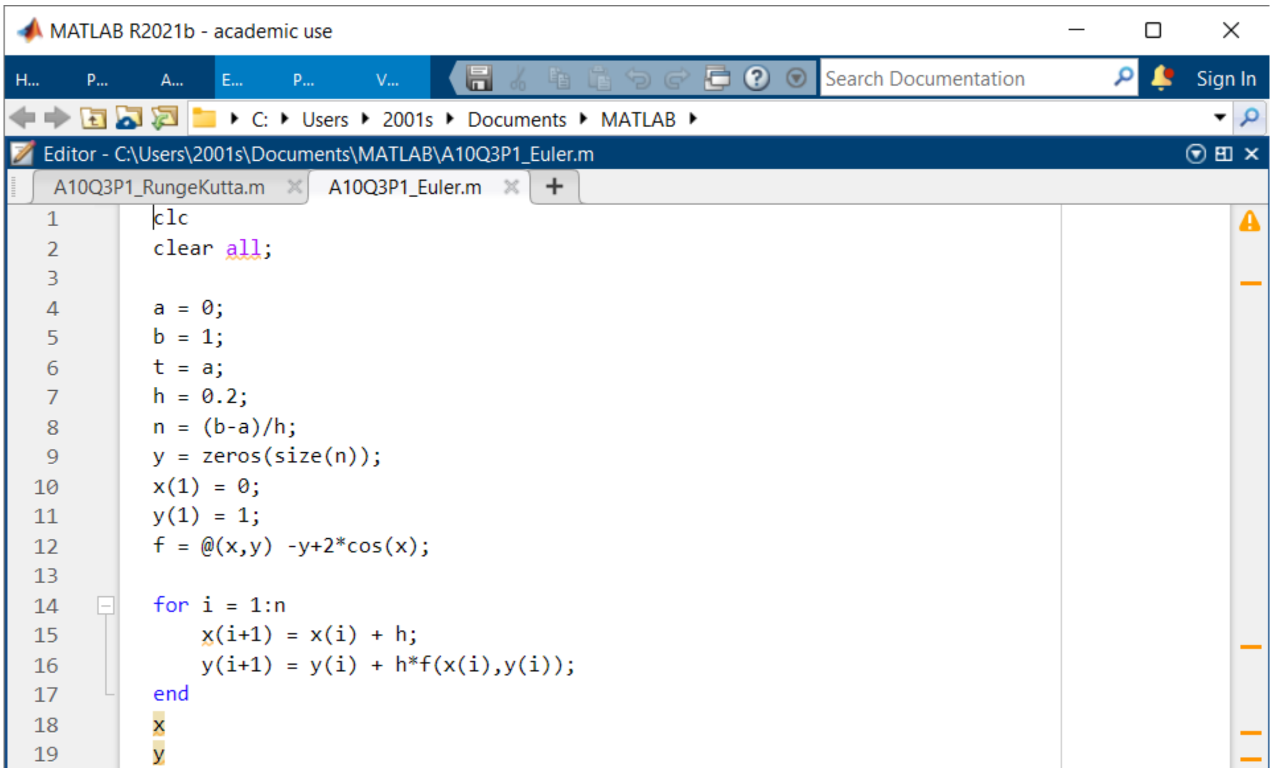
3. Compute solution of the following differential equation by the Euler's method and Runga-Kutta fourth-order method in the interval [0,1] with step length 0.2:

(a) $y' = -y + 2\cos t, y\ 0 = 1$.

(b) $y' = \overline{2 + y}\ , y\ 0 = 0.8$. (c)

$y' = (\cos y)^2 , y\ 0 = 0$.

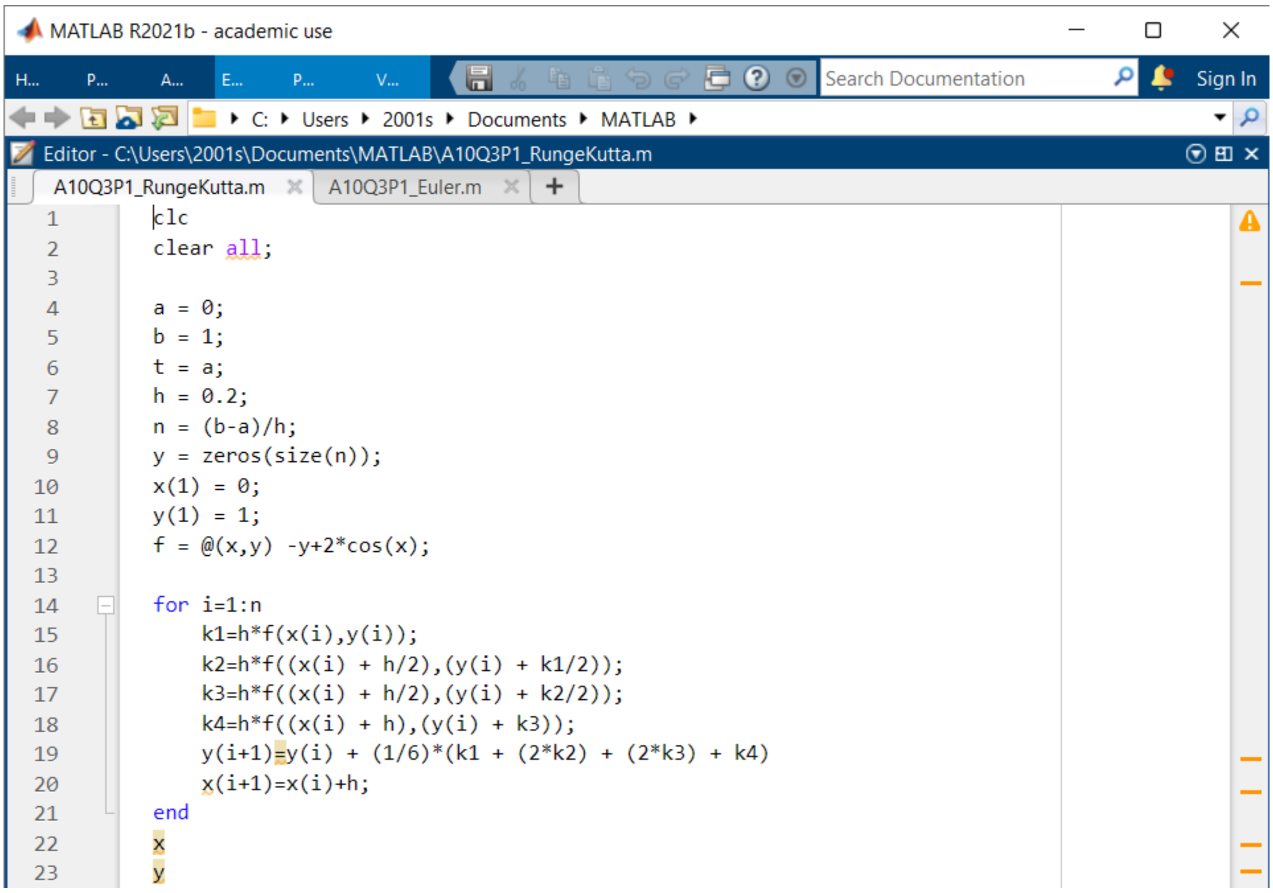
Q3 (i): Euler


MATLAB R2021b - academic use
H... P... A... E... P... V...
Search Documentation
Sign In
C: Users 2001s Documents MATLAB
Editor - C:\Users\2001s\Documents\MATLAB\A10Q3P1_Euler.m
A10Q3P1_RungeKutta.m
A10Q3P1_Euler.m

```
clc
clear all;

a = 0;
b = 1;
t = a;
h = 0.2;
n = (b-a)/h;
y = zeros(size(n));
x(1) = 0;
y(1) = 1;
f = @(x,y) -y+2*cos(x);

for i = 1:n
    x(i+1) = x(i) + h;
    y(i+1) = y(i) + h*f(x(i),y(i));
end
x
y
```

```
x =

         0    0.2000    0.4000    0.6000    0.8000    1.0000


y =

    1.0000    1.2000    1.3520    1.4500    1.4902    1.4708
```

(i): Runge-Kutta


MATLAB R2021b - academic use
Search Documentation
Sign In
C: ▸ Users ▸ 2001s ▸ Documents ▸ MATLAB ▸
Editor - C:\Users\2001s\Documents\MATLAB\A10Q3P1_RungeKutta.m
A10Q3P1_RungeKutta.m
A10Q3P1_Euler.m

```
clc
clear all;

a = 0;
b = 1;
t = a;
h = 0.2;
n = (b-a)/h;
y = zeros(size(n));
x(1) = 0;
y(1) = 1;
f = @(x,y) -y+2*cos(x);

for i=1:n
    k1=h*f(x(i),y(i));
    k2=h*f((x(i) + h/2),(y(i) + k1/2));
    k3=h*f((x(i) + h/2),(y(i) + k2/2));
    k4=h*f((x(i) + h),(y(i) + k3));
    y(i+1)=y(i) + (1/6)*(k1 + (2*k2) + (2*k3) + k4)
    x(i+1)=x(i)+h;
end
x
y
```

```
x =

         0    0.2000    0.4000    0.6000    0.8000    1.0000


y =

    1.0000    1.1787    1.3105    1.3900    1.4141    1.3818
```