NUMERICAL ANALYSIS LAB MENUAL

Upoma Das
Lecturer,
Department of EEE, BSMRSTU

MATLAB Program No: 01
Program Name: **An M-File to implement the Bisection Method.**

Problem: **Find a real root of the equation $f(x)=x^3-x-1=0$ by using bisection method.**

Main File:

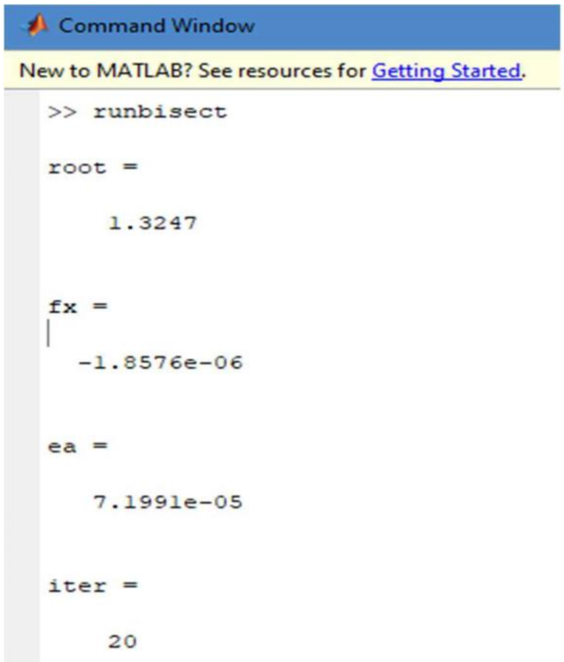
```
function [root,fx,ea,iter]=bisect(func,xl,xu,es,maxit,varargin)
% bisect: root location zeroes
% [root,fx,ea,iter]=bisect(func,xl,xu,es,maxit,p1,p2,...):
% uses bisection method to find the root of func
% input:
% func = name of function
% xl, xu = lower and upper guesses
% es = desired relative error (default = 0.0001%)
% maxit = maximum allowable iterations (default = 50)
% p1,p2,... = additional parameters used by func
% output:
% root = real root
% fx = function value at root
% ea = approximate relative error (%)
% iter = number of iterations
if nargin<3,error('at least 3 input arguments required'),end test =
func(xl,varargin{:})*func(xu,varargin{:});
if test>0,error('no sign change'),end if
nargin<4|isempty(es), es=0.0001;end if
nargin<5|isempty(maxit), maxit=50;end iter = 0; xr
= xl; ea = 100;
while (1)
xrold = xr;
xr = (xl + xu)/2;
iter = iter + 1;
if xr ~= 0,ea = abs((xr - xrold)/xr) * 100;end
test = func(xl,varargin{:})*func(xr,varargin{:});
if test < 0
xu = xr;
elseif test > 0
xl = xr;
else
ea = 0;
end
if ea <= es | iter >= maxit,break,end
end
root = xr; fx = func(xr, varargin{:});
```

Input File:

```
fx=@(x)x^3-x-1;
[root fx ea iter]=bisect(fx,1,2)
```

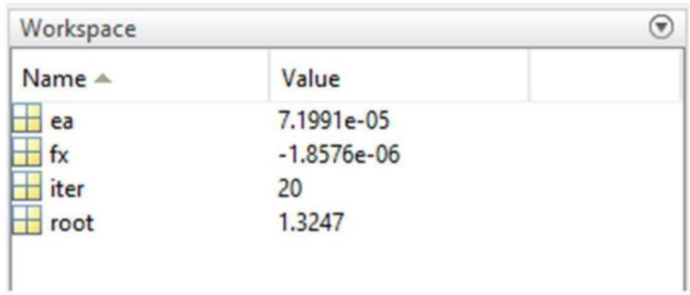
OUTPUT

On Command Window



```
Command Window
New to MATLAB? See resources for Getting Started.
>> runbisect

root =

    1.3247


fx =

  -1.8576e-06


ea =

   7.1991e-05


iter =

    20
```

On Workspace



Workspace

Name	Value
ea	7.1991e-05
fx	-1.8576e-06
iter	20
root	1.3247

Related Questions:

1. **What is Bisection method?**
2. **Verify this program by solving at least three problems.**
3. **Make an m-file to implement False Position Method.**

MATLAB Program No: 02
Program Name: An M-File to implement the Newton Raphson Method.

Problem: **Use the Newton-Raphson method to find a root of the equation x^3-2x-5=0.**

Main File:

```
function
[root,ea,iter]=newtraph(func,dfunc,xr,es,maxit,varargin)
% newtraph: Newton-Raphson root location zeroes
% [root,ea,iter]=newtraph(func,dfunc,xr,es,maxit,p1,p2,...):
% uses Newton-Raphson method to find the root of func
% input:
% func = name of function
% dfunc = name of derivative of function
% xr = initial guess
% es = desired relative error (default = 0.0001%)
% maxit = maximum allowable iterations (default = 50)
% p1,p2,... = additional parameters used by function
% output:
% root = real root
% ea = approximate relative error (%)
% iter = number of iterations
if nargin<3,error('at least 3 input arguments required'),end if
nargin<4|isempty(es),es=0.0001;end
if nargin<5|isempty(maxit),maxit=50;end iter = 0;

while (1)
xrold = xr;
xr = xr - func(xr)/dfunc(xr);
iter = iter + 1;
if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end if ea <= es | iter
>= maxit, break, end end

root = xr;
```

Input File:

```
y=@(x)x^3-2*x-5;
dy=@(x)3*x^2-2;
        [root ea iter]=newtraph(y,dy,2,0.00001)
```

OUTPUT

On Command Window

```
Command Window
New to MATLAB? See resources for Getting Started.

>> runnewtraph

root =

    2.0946


ea =

   7.4418e-09


iter =

     4
```

On Workspace

Name	Value
dy	@(x)3*x^2-2
ea	7.4418e-09
fx	-1.8576e-06
iter	4
root	2.0946
y	@(x)x^3-2*x-5

Related Questions:

1. What is the difference between bracketing method and open method?
2. As an engineer which method will you prefer?
3. Use the above m-file to estimate the root of $f(x) = e^{-x} - x$ employing an initial guess of $x_0 = 0$.
4. $f(x) = x^3 - 6x^2 + 11x - 6.1$. Determine the root with MATLAB.
5. Write a new matlab program to implement Newton Raphson method which doesn't need the derivative of a function.

MATLAB Program No: 03
Program Name: An M-File to implement Linear regression.

Problem: An experiment gave the following table of values for the dependent variable y for a set of known values of x. Obtain an appropriate least squares fit for the data and plot y vs x with MATLAB.

x	1	2	3	4	5	6	7	8	9
y	5.5	7.0	9.6	11.5	12.6	14.4	17.6	19.5	20.5

Main File:

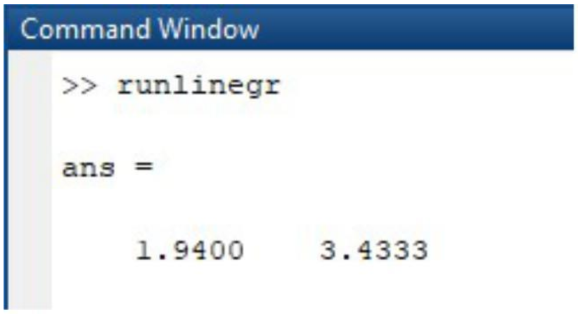
```
function [a, r2] = linregr(x,y)
% linregr: linear regression curve fitting
% [a, r2] = linregr(x,y): Least squares fit of straight
% line to data by solving the normal equations
% input:
% x = independent variable
% y = dependent variable
% output:
% a = vector of slope, a(1), and intercept, a(2)
% r2 = coefficient of determination
n = length(x);
if length(y)~=n, error('x and y must be same length'); end x = x(:); y = y(:);
% convert to column vectors sx = sum(x); sy = sum(y);

sx2 = sum(x.*x); sxy = sum(x.*y); sy2 = sum(y.*y);
a(1) = (n*sxy-sx*sy)/(n*sx2-sx^2);
a(2) = sy/n-a(1)*sx/n;
r2 = ((n*sxy-sx*sy)/sqrt(n*sx2-sx^2)/sqrt(n*sy2-sy^2))^2;
% create plot of data and best fit line xp =
linspace(min(x),max(x),2);
yp = a(1)*xp+a(2);
plot(x,y,'o',xp,yp) grid
on
```

Input File:

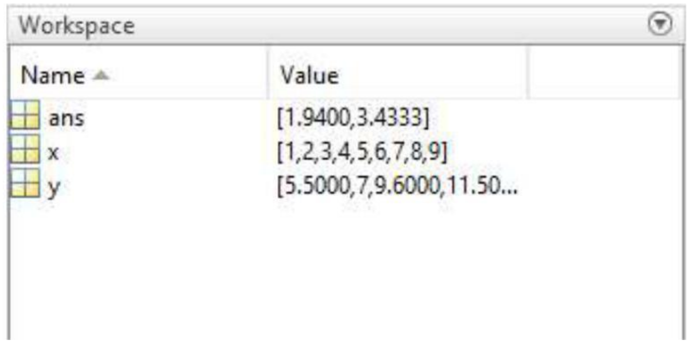
```
x=[1 2 3 4 5 6 7 8 9];
y=[5.5 7 9.6 11.5 12.6 14.4 17.6 19.5 20.5]; linregr(x,y)
```

OUTPUT

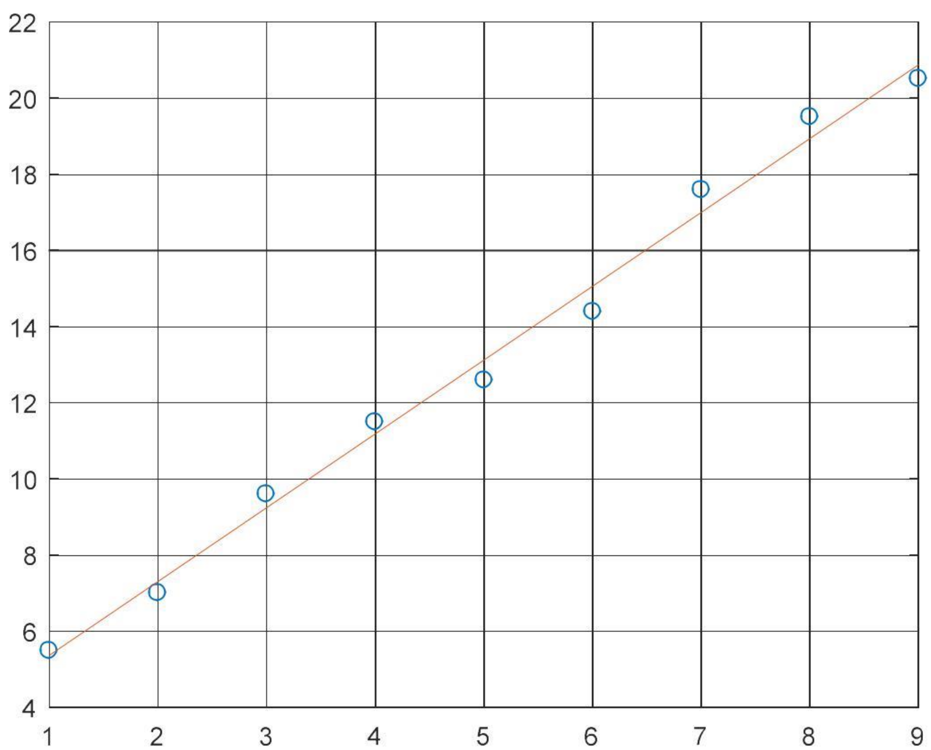
On Command Window



On Workspace



<u>Plot y vs x:</u>



Related Questions:

1. **What is least squares approximation of functions? Explain it.**
2. **Why it is important for an engineer?**

MATLAB Program No: 04
Program Name: An M-File to implement Lagrange Interpolation.

Problem: Given the following tables of values

x	0.4	0.5	0.7	0.8
y	-0.916	-0.693	-0.357	-0.223

Find the value of f(0.6).

Main File:

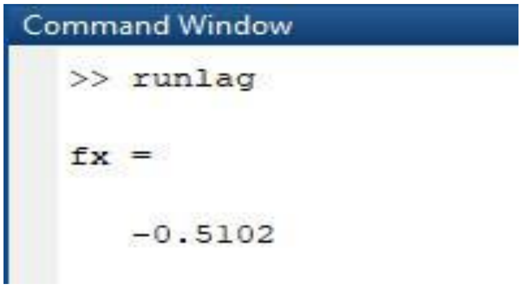
```
function yint = Lagrange(x,y,xx)
% Lagrange: Lagrange interpolating polynomial
% yint = Lagrange(x,y,xx): Uses an (n - 1)-order
% Lagrange interpolating polynomial based on n data points
% to determine a value of the dependent variable (yint) at
% a given value of the independent variable, xx.
% input:
% x = independent variable
% y = dependent variable
% xx = value of independent variable at which the
% interpolation is calculated
% output:
% yint = interpolated value of dependent variable
n = length(x);
if length(y)~=n, error('x and y must be same length'); end s = 0;

for i = 1:n
product = y(i);
for j = 1:n
if i ~= j
product = product*(xx-x(j))/(x(i)-x(j));
end
end
s = s+product;
end
yint = s;
```

Input program :

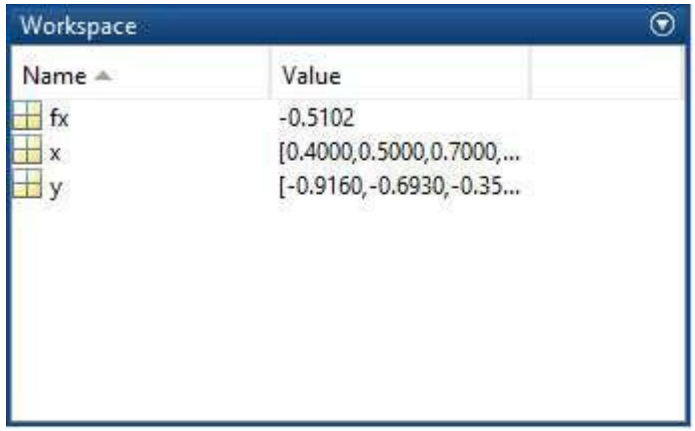
```
x=[.4 .5 .7 .8]; y=[-.916 -.693 -.357 -
.223]; fx=Lagrange(x,y,0.6)
```

OUTPUT

On Command Window



On Workspace



Related Questions:

1. **What is the main difference between interpolation & least squares approximation?**
2. **Make a m-file to implement Inverse Lagrange Interpolation.**
3. **Is it possible to find the value of f(0.9)using the above m-file? Justify your answer.**

MATLAB Program No: 05
Program Name: An M-File to implement Newton interpolation.

Problem: Values of x (in degrees) and sin x are given in the following table:

x (in degrees)	15	20	25	30	35	40
sin x	0.258819	0.3420201	0.4226183	0.5	0.5735764	0.6427876

Determine the value of sin 38°.

Main File:

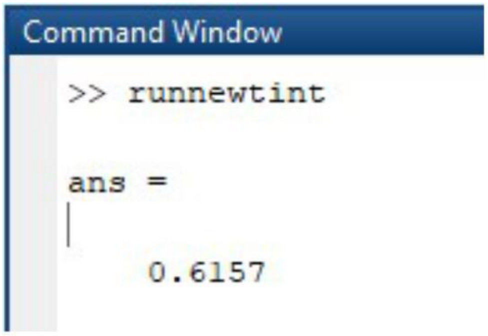
```
function yint = Newtint(x,y,xx)
% Newtint: Newton interpolating polynomial
% yint = Newtint(x,y,xx): Uses an (n - 1)-order Newton
% interpolating polynomial based on n data points (x, y)
% to determine a value of the dependent variable (yint)
% at a given value of the independent variable, xx.
% input:
% x = independent variable
% y = dependent variable
% xx = value of independent variable at which
% interpolation is calculated
% output:
% yint = interpolated value of dependent variable
% compute the finite divided differences in the form of a
% difference table
n = length(x);
if length(y)~=n, error('x and y must be same length'); end b = zeros(n,n);

% assign dependent variables to the first column of b. b(:,1) = y(:); % the
(:) ensures that y is a column vector. for j = 2:n

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

end
% use the finite divided differences to interpolate
xt = 1;
yint = b(1,1);
for j = 1:n-1
xt = xt*(xx-x(j));
yint = yint+b(1,j+1)*xt;
end
```

Input File:

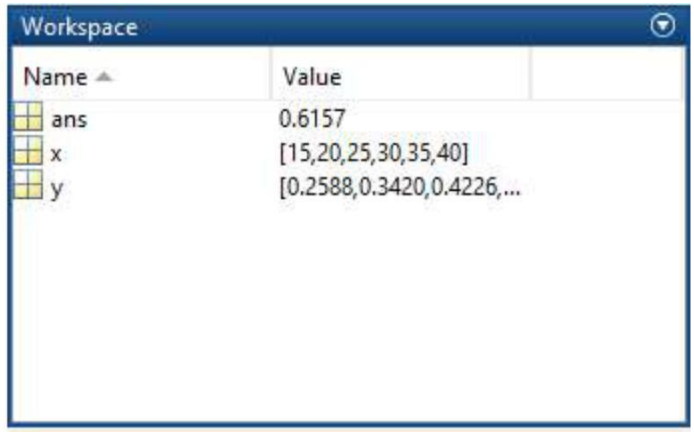
```
x=[15 20 25 30 35 40];
y=[.2588190 .3420201 .4226183 .5 .5735764 .6427876];
Newtint(x,y,38)
```

OUTPUT

On Command Window



On Workspace



Related Questions:

1. **Make a m-file to implement Bessel's Formulae.**
2. **Make a m-file to solve the above problem by using central difference table.**

MATLAB Program No: 06
Program Name: **An M-File to implement the Trapezoidal Rule.**

Problem: **Evaluate** **I =** $\int_0^1 \frac{1}{1+x}\,dx$ correct to three decimal places with h=0.125.

Main File:

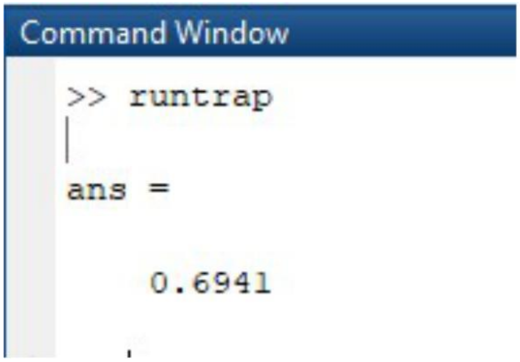
```
function I = trap(func,a,b,n,varargin)
% trap: composite trapezoidal rule quadrature
% I = trap(func,a,b,n,pl,p2,...):
% composite trapezoidal rule
% input:
% func = name of function to be integrated
% a, b = integration limits
% n = number of segments (default = 100)
% pl,p2,... = additional parameters used by func
% output:
% I = integral estimate
if nargin<3,error('at least 3 input arguments required'),end if
~(b>a),error('upper bound must be greater than lower'),end if
nargin<4|isempty(n),n=100;end
x = a; h = (b - a)/n;
s=func(a,varargin{:});
for i = 1 : n-1
x = x + h;
s = s + 2*func(x,varargin{:});
end
s = s + func(b,varargin{:});
I = (b - a) * s/(2*n);
```

Input File:

```
y=@(x)1/(1+x);
trap(y,0,1,8)
```

OUTPUT

On Command Window



On Workspace

Related Questions:

1. **Write a program to implement Simpson's 1/3 rule.**
2. **Write a program to implement Boole's and Weddle's Rule.**

MATLAB Program No: 07
Program Name: An M-File to implement the Trapezoidal Rule for unequally spaced data.

Problem: Use the information in the following Table to integrate the function $f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$.

x	00	0.12	0.22	0.32	0.36	0.40
f(x)	0.200000	1.309729	1.305241	1.743393	2.074903	2.456000

x	0.44	0.54	0.64	0.70	0.80
f(x)	2.842985	3.507297	3.181929	2.363000	0.232000

Main File:

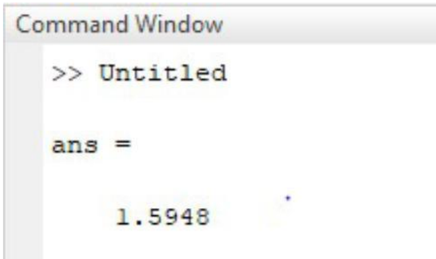
```
function I = trapuneq(x,y)
% trapuneq: unequal spaced trapezoidal rule quadrature
% I = trapuneq(x,y):
% Applies the trapezoidal rule to determine the integral
% for n data points (x, y) where x and y must be of the
% same length and x must be monotonically ascending
% input:
% x = vector of independent variables
% y = vector of dependent variables
% output:
% I = integral estimate
if nargin<2,error('at least 2 input arguments required'),end if
any(diff(x)<0),error('x not monotonically ascending'),end n = length(x);

if length(y)~=n,error('x and y must be same length'); end s = 0;

for k = 1:n-1
s = s + (x(k+l)-x(k))*(y(k)+y(k+l))/2; end

I = s;
```

Input File:

```
x = [0 .12 .22 .32 .36 .4 .44 .54 .64 .7 .8];
y  = 0.2+25*x-200*x.^2+675*x.^3-900*x.^4+400*x.^5; trapz(x,y)
```

OUTPUT

On Command Window



On Workspace

Related Questions:

1. Compare the above program with previous program (Matlab Program No:06)
2. Is it possible to implement Trapezoidal rule for equally spaced data by the above program?

MATLAB Program No: 08
Program Name: An M-File to implement the Gauss Elimination method.

Problem: Solve the following system of equations by Gauss Elimination method.

$$2x_1+3x_2-x_3+2x_4=7$$
$$x_1+x_2+x_3+x_4=2$$
$$x_1+x_2+3x_3-2x_4=-6$$
$$x_1+2x_2+x_3-x_4=-2$$

Main File:

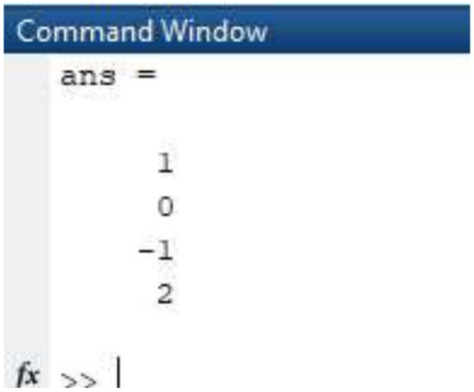
```
function x = GaussNaive(A,b)
% GaussNaive: naive Gauss elimination
% x = GaussNaive(A,b): Gauss elimination without pivoting.
% input:
% A = coefficient matrix
% b = right hand side vector
% output:
% x = solution vector
[m,n] = size(A);
if m~=n, error('Matrix A must be square'); end
nb = n+1;
Aug = [A b];
% forward elimination for
k = 1:n-1
for i = k+1:n
factor = Aug(i,k)/Aug(k,k);
Aug(i,k:nb) = Aug(i,k:nb)-factor*Aug(k,k:nb); end

end
% back substitution
x = zeros(n,1);
x(n) = Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
x(i) = (Aug(i,nb)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,i); end
```

Input File:

```
    A=[2 3 -1 2
         1 1 1 1
      1 1 3 -2
      1 2 1 -1];
b=[ 7 2 -6 -2]';
GaussNaive(A,b)
```

OUTPUT

On Command Window



On Workspace

Related Questions:

1. **Write a program to implement Gauss-Jordan Method.**
2. **Write a program to implement LU Decomposition Method.**

<u>MATLAB Program No:</u> 09
<u>Program Name:</u> An M-File to implement the GaussSeidel iterative method.

Problem: Solve the following system of equations

$$10x+2y+z=9$$
$$2x+20y-2z=-44$$
$$-2x+3y+10z=22$$

By GaussSeidel iterative method.

Main File:

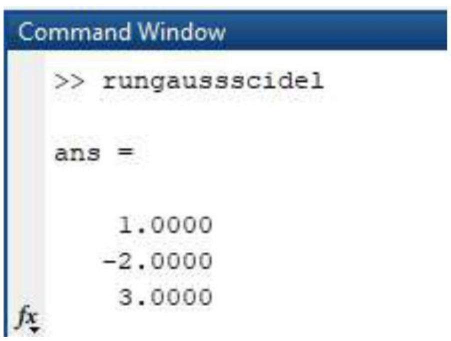
```
function x = GaussSeidel(A,b,es,maxit)
% GaussSeidel: Gauss Seidel method
% x = GaussSeidel(A,b): Gauss Seidel without relaxation
% input:
% A = coefficient matrix
% b = right hand side vector
% es = stop criterion (default = 0.00001%)
% maxit = max iterations (default = 50)
% output:
% x = solution vector
if nargin<2,error('at least 2 input arguments required'),end if
nargin<4|isempty(maxit),maxit=50;end if
nargin<3|isempty(es),es=0.00001;end
[m,n] = size(A);
if m~=n, error('Matrix A must be square'); end C = A;

for i = 1:n
C(i,i) = 0;
x(i) = 0;
end
x = x';
for i = 1:n
C(i,1:n) = C(i,1:n)/A(i,i);
end
for i = 1:n
d(i) = b(i)/A(i,i);
end
iter = 0;
while (1)
xold = x;
for i = 1:n
x(i) = d(i)-C(i,:)*x;
if x(i) ~= 0
ea(i) = abs((x(i) - xold(i))/x(i)) * 100;
end
end
iter = iter+1;
if max(ea)<=es | iter >= maxit, break, end end
```

Input File:

```
A=[10   2 1
  2   20-2
  -2 310];
b=[9 -4422]';
GaussSeidel(A,b)
```

OUTPUT

On Command Window



On Workspace

Related Questions:

1. **Write a program to implement Jacobi's Method.**
2. **Solve the following system of equations**

$$x_1+2x_2=9$$
$$2x_1+2x_3=-44$$
$$-3x_2+10x_3=22$$

By GaussSeidel iterative method.