**CSU33081 Multiple Choice Answers**

Please enter your answers (A – E) and upload with your type written solutions as a .docx file

Q 1 Answer: E: None of the above

function Xs = SquareRoot(p)

if p <= 0

error('Input must be a positive number');

end

x = p;

max\_iterations = 20;

tolerance = 0.00001;

for iteration = 1:max\_iterations

f\_x = x^2 - p;

f\_prime\_x = 2 \* x;

x\_new = x - (f\_x / f\_prime\_x);

relative\_error = abs((x\_new - x) / x\_new);

x = x\_new;

if relative\_error < tolerance

break;

end

end

Xs = x;

End

When max iterations is changed to 5, and 729 is given as an input, the answer given is 32.5596.

After the full 20 iterations, the answer is D = 27.00

Q2 Answer: B – x1=3, x2=1, x3=4, x4=2

% Gauss Jordan with Pivoting

function x = GaussJordan(a,b)

% Check if the input matrix is square

[n,m] = size(a);

if n~=m

error('Matrix must be square');

end

ab=[a,b]; % Augmented matrix with solution vector

[R,C] = size(ab); % Rows & cols of augmented matrix

for j=1:R % Loop through each pivot element

% Pivoting section starts

if ab(j,j) == 0 % If the pivot element is zero

for k = j + 1:R % Iterate over the matrix

if ab(k,j) ~= 0 % If another element is not zero

abTemp = ab(j,:); % Temp = pivot row

ab(j,:) = ab(k,:); % Pivot row = other row

ab(k,:) = abTemp; % Other row = pivot row

break

end

end

end

% Pivoting section ends

% Normalise the pivot row / make pivot element 1

ab(j,:) = ab(j,:) / ab(j,j); % Divide current row by pivot row

% Eliminate other rows

for i = 1:R

if i ~= j

ab(i,:) = ab(i,:) - ab(i,j) \* ab(j,:); % Eliminate other rows

end

end

end

% Extract solution

x = ab(:,end);

end

Q3 Answer:

Q 4 Answer:

Q 5 Answer: C: 1.900475

% Secant Method

%Init variables

x0 =3;

x1 = 2.5;

f = @(x) 16\*x.^5 - 73\*x.^2 - 133;

error = 0.001;

i = 0;

while abs(x1-x0)>error

    f\_x0 = f(x0);

    f\_x1 = f(x1);

    x\_temp = (x1 - ((f\_x1 \* (x0-x1))/(f\_x0 - f\_x1)));

    x0=x1;

    x1=x\_temp;

    i=i+1;

    % Display the current approximation and error

    fprintf('Iteration %d: x = %.6f, Error = %.6f\n', i, x1, abs(x1 - x0));

end

% Display final result

fprintf('Root found: x = %.6f\n', x1);

Q 6 Answer: E: None of the above

% Newton Raphson

f = @(x) x.^6 - x.^2 - 1;

f\_prime = @(x) 6\*x.^5 - 2\*x;

error = 0.001;

xi = 1.5; % Initial guess

while true

x\_new = xi - f(xi) / f\_prime(xi); % Compute the new x value

if abs(x\_new - xi) < error

break

end

xi = x\_new; % Update xi to the new value

end

% Display final result

fprintf('Root found: x = %.6f\n', xi);

Answer obtained from the above calculations was “Root found: x = 1.151017”

Q 7 Answer:

Q 8 Answer: D – [4 -5 -2; 5 -6 -2; -8 9 3;]

% Gauss Jordan Inversion of a Matrix

a = [0 -3 -2; 1 -4 -2; -3 4 1;]; % Question matrix

b = [1 0 0; 0 1 0; 0 0 1;]; % Diagonal identity matrix

ab=[a,b]; % Augmented matrix with solution vector

[R,C] = size(ab); % Rows & cols of augmented matrix

% GaussJordan Elimination

for i=1:R % Loop through each pivot element

% Pivoting section starts

if ab(i,i) == 0 % If the pivot element is zero

for k = i + 1:R % Iterate over the matrix

if ab(k,i) ~= 0 % If another element is not zero

abTemp = ab(i,:); % Temp = pivot row

ab(i,:) = ab(k,:); % Pivot row = other row

ab(k,:) = abTemp; % Other row = pivot row

break

end

end

end

ab(i, :) = ab(i, :) / ab(i, i); % Divide the pivot row by its pivot element

for j = 1:R % Iterate over the other rows

if j ~= i % If not the pivot row

ab(j, :) = ab(j, :) - ab(j, i) \* ab(i, :); % Eliminate other rows

end

end

end

inversion = ab(:, R+1:end); % Take the 2nd half of the matrix

disp(inversion);

Q 9 Answer: E – None of the above. After 3 iterations: x1 ~ 0.5361, x2 ~ -1.2170, x3 ~

-0.6433

% Jacobi Iteration

% [12 7 3; 1 5 1; 2 7 -11;]:

% [2; -5; 6 ;];

x1=1;

x2=3;

x3=5;

max\_iter = 3;

while max\_iter ~= 0

    x1\_new = (2-((7\*x2)+(3\*x3)))/12;

    x2\_new = (-5-(1\*x1)+(1\*x3))/5;

    x3\_new = (6-((2\*x1)+(7\*x2)))/-11;

    max\_iter = max\_iter -1;

    x1=x1\_new;

    x2=x2\_new;

    x3=x3\_new;

end

disp(x1);

disp(x2);

disp(x3);

Q 10 Answer: