- Q2.31

Matlab Code

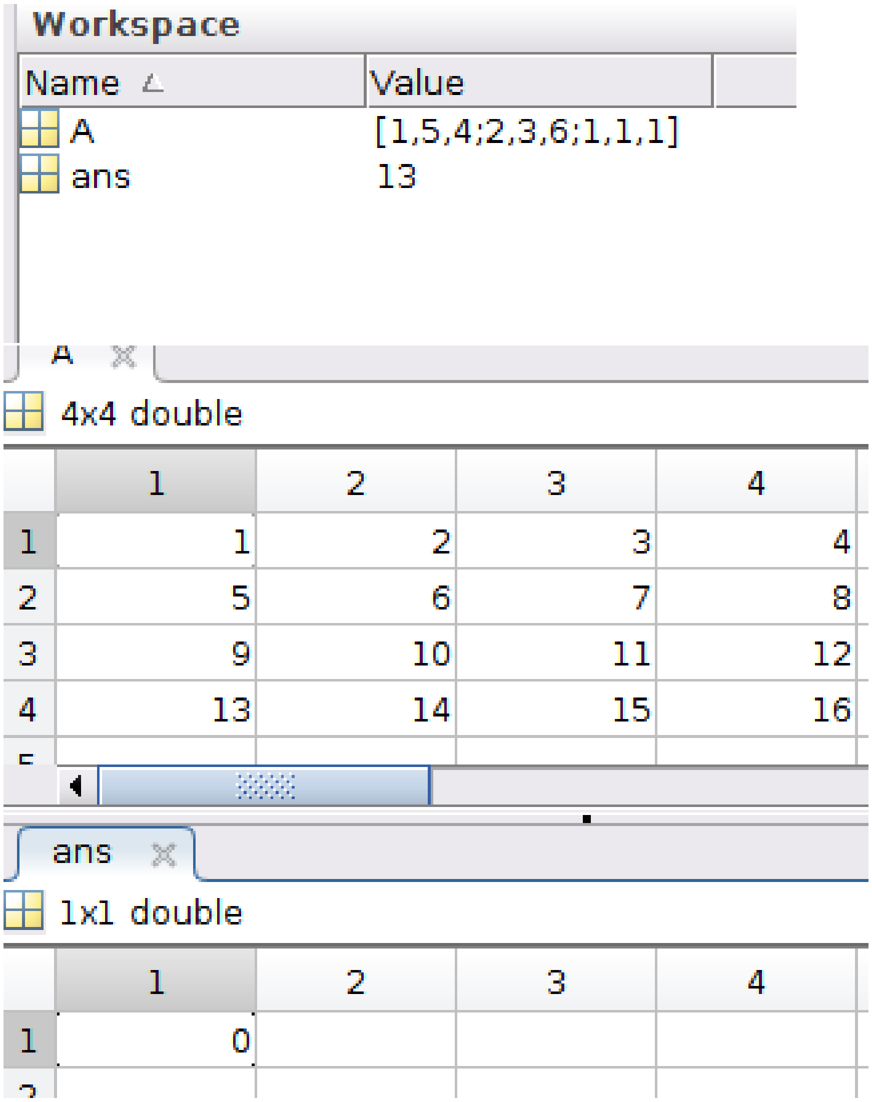
```matlab
function D = Determinant(A)
[n, m]=size(A);  % n= no. of rows in A, m= no. of columns in A

if n ~= m     % check if A is square
    D ='The matrix must be square';

elseif n > 4 % check if bigger than 4x4
    D ='Matrix too big';

elseif n == 2 % 2x2 matrix determinant
        D = A(1,1)*A(2,2) - A(1,2)*A(2,1);

elseif n == 3 % 4x4 matrix
    % 3 new matrixes, ready for recursive call
    M1 = [A(2,2),A(2,3);A(3,2),A(3,3)];
    M2 = [A(2,1),A(2,3);A(3,1),A(3,3)];
    M3 = [A(2,1),A(2,2);A(3,1),A(3,2)];
    % recursive call below to 2x2 matrix
        D = A(1,1)*Determinant(M1) -  A(1,2)*Determinant(M2) +  A(1,3)*Determinant(M3);
elseif n == 4  %4x4 matrix
    % 4 matrixes
    M1 = [A(2,2),A(2,3),A(2,4); A(3,2),A(3,3),A(3,4); A(4,2),A(4,3),A(4,4)];
    M2 = [A(2,1),A(2,3),A(2,4); A(3,1),A(3,3),A(3,4); A(4,1),A(4,3),A(4,4)];
    M3 = [A(2,1),A(2,2),A(2,4); A(3,1),A(3,2),A(3,4); A(4,1),A(4,2),A(4,4)];
    M4 = [A(2,1),A(2,2),A(2,3); A(3,1),A(3,2),A(3,3); A(4,1),A(4,2),A(4,3)];
    % calling recurisvely the function with 4, 3x3 matrixes
        D = A(1,1)*Determinant(M1)-A(1,2)*Determinant(M2) + A(1,3)*Determinant(M3)-A(1,4)*Determinant(M4);
end
```



Workspace

Name	Value
A	[1,5,4;2,3,6;1,1,1]
ans	13

4x4 double

	1	2	3	4
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12
4	13	14	15	16
5				

1x1 double

	1	2	3	4
1	0			
2				

- Q3.2
  root of f(x) = x - $2e^{-x}$

  (a) Bisection method

$$a = 0, b = 1$$

$$f(0) = 0 - 2e^0 = -2$$

$$f(1) = 1 - 2e^{-1} = 0.2642$$

First iteration:

$$x_1 = \frac{a+b}{2} = \frac{0+1}{2} = 0.5$$

$$f(x_1) = 0.5 - 2e^{-0.5} = -0.7$$

as this is a negative number and you need a negative value and positive value between a and b; a now equals c_1
Second iteration:

$$a = 0.5, b = 1$$

$$x_2 = \frac{0.5+1}{2} = 0.75$$

$$f(x_2) = 0.75 - 2e^{-0.75} = -0.1947$$

as this is a negative number, it replaces the previous negative number, a
Third iteration:

$$a = 0.75, b = 1$$

$$x_3 = \frac{0.75+1}{2} = 0.875$$

$$f(x_3) = 0.875 - 2e^{-0.875} = 0.04$$

as this is a positive number, this is the new b
Final iteration:

$$a = 0.75, b = 0.875$$

$$x_4 = \frac{0.75+0.875}{2} = 0.8125$$

(b) Secant method

$$x_1 = 0, x_2 = 1$$

$$f(x_1) = 0 - 2e^0 = -2$$

$$f(x_2) = 1 - 2e^{-1} = 0.2642$$

First iteration

$$x_3 = x_2 - f(x_2) * \frac{x_2 - x_1}{f(x_2) - f(x_1)}$$

$$x_3 = 1 - 0.2642 * \frac{1-0}{0.2642+2} = 0.8516$$

$$f(x_3) = 0.8833 - 2e^{-0.8833} = 0.0565$$

Second iteration

$$x_4 = x_3 - f(x_3) * \frac{x_3 - x_2}{f(x_3) - f(x_2)}$$

$$x_4 = 0.8833 - 0.0565 * \frac{0.8833 - 1}{0.0565 - 0.2642} = 0.8516$$

$$f(x_4) = 0.8516 - 2e^{-0.8516} = -0.0019$$

Third iteration

$$x_5 = x_4 - f(x_4) * \frac{x_4 - x_3}{f(x_4) - f(x_3)}$$

$$x_5 = 0.8516 + 0.0019 * \frac{0.8516 - 0.8833}{-0.019 - 0.0565} = 0.8529$$

$$f(x_5) = 0.8529 - 2e^{-0.8529} = 0.00054$$

Final iteration

$$x_6 = x_5 - f(x_5) * \frac{x_5 - x_4}{f(x_5) - f(x_4)}$$

$$x_6 = 0.8529 - 0.00054 * \frac{0.8529 - 0.8516}{0.00054 + 0.0019} = 0.85261$$

(c) Newton's method

$$f(x) = x - 2e^{-x}, x_1 = 1$$

$$f'(x) = 2e^{-x} + 1$$

$$f(x_1) = 1 - 2e^{-1} = 0.2642$$

$$f'(x_1) = 2e^{-1} + 1 = 1.7358$$

First iteration

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$x_2 = 1 - \frac{0.2642}{1.7358} = 0.8478$$

$$f(x_2) = 0.8478 - 2e^{-0.8478} = -0.0089$$

$$f'(x_2) = 2e^{-0.8478} + 1 = 1.8567$$

Second iteration

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

$$x_3 = 0.8478 - \frac{-0.0089}{1.8467} = 0.8526$$

$$f(x_2) = 0.8526 - 2e^{-0.8526} = -0.0001$$

$$f'(x_2) = 2e^{-0.8526} + 1 = 1.8526$$

Third iteration

$$x_4 = x_3 - \frac{f(x_3)}{f'(x_3)}$$

$$x_4 = 0.8526 - \frac{-0.00001}{1.8526} = 0.8526$$

$$f(x_4) = 0.8526 - 2e^{-0.8526} = -0.0001$$

$$f'(x_4) = 2e^{0.8526} + 1 = 1.8567$$

Final iteration

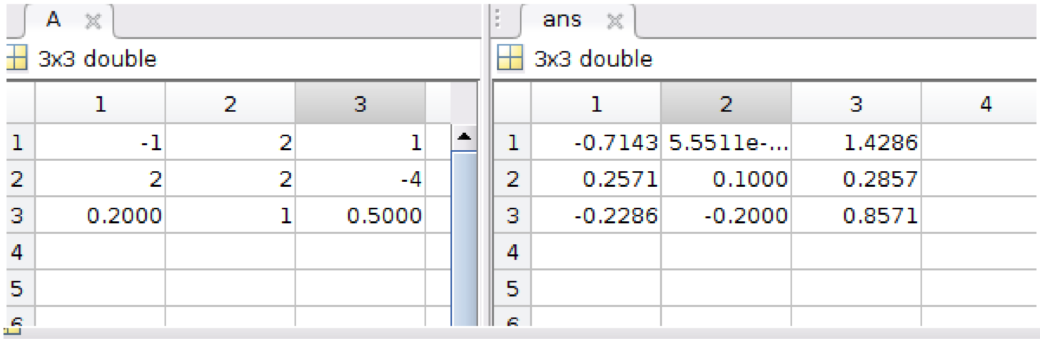
$$x_5 = 0.8526 - \frac{-0.00001}{1.8526} = 0.8256$$

- Q4.24

Matlab Code

```
function Ainv = Inverse (A)
[n, m]=size(A);  % n= no. of rows in A, m= no. of columns in A
if n ~= m     % check if A is a square matrix
    Ainv ='The matrix must be square';
    return
end
if n == 0     % check if A is empty
    Ainv ='Matrix cant be empty';
    return
end

Ainv = eye(n); % set up identity matrix
for r = 1 : n
    for c = r : n
        if A(c,r) ~= 0 % cannot operate a division on a zero
            t = 1/A(r,r); % left val of row being operated on
            for i = 1 : n
                %getting A(r,k) to be 1
                A(r,i) = t * A(r,i);
                %repeat calculation on identity matrix
                Ainv(r,i) = t * Ainv(r,i);
            end
            %looking for not diagonal elements
            for i = 1 : n
                if i ~= r % if count doesnt equal row
                    % for rows that
                    t = -A(i,r);
                    for j = 1 : n
                        % subtracting to get the
                            element to be zero
                        A(i,j) = A(i,j) + t * A(r,j);
                        Ainv(i,j) = Ainv(i,j) + t *
                            Ainv(r,j);
                    end
                end
            end
        end
        break
    end
end
```

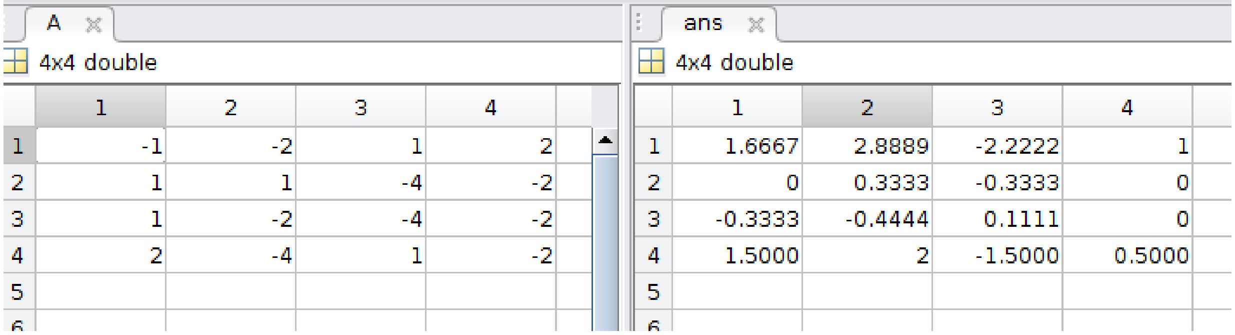


A — 3x3 double

	1	2	3
1	-1	2	1
2	2	2	-4
3	0.2000	1	0.5000
4			
5			

ans — 3x3 double

	1	2	3	4
1	-0.7143	5.5511e-...	1.4286	
2	0.2571	0.1000	0.2857	
3	-0.2286	-0.2000	0.8571	
4				
5				



A — 4x4 double

	1	2	3	4
1	-1	-2	1	2
2	1	1	-4	-2
3	1	-2	-4	-2
4	2	-4	1	-2
5				

ans — 4x4 double

	1	2	3	4
1	1.6667	2.8889	-2.2222	1
2	0	0.3333	-0.3333	0
3	-0.3333	-0.4444	0.1111	0
4	1.5000	2	-1.5000	0.5000
5				