CS3081 Computational Mathematics

Séamus Woods
15317173

19/02/2019

0.1 Question 2.31

Question: Write a user-defined MATLAB function that calculates the determinant of a square (n x n) matrix, where n can be 2, 3, or 4. For function name and arguments, use D = Determinant (A). The input argument A is the matrix whose determinant is calculated. The function Determinant should first check if the matrix is square. If it is not, the output D should be the message "The matrix must be square." Use Determinant to calculate the determinant of the following two matrices:

$$\overset{\text{(a)}}{\begin{bmatrix} 1 & 5 & 4 \\ 2 & 3 & 6 \\ 1 & 1 & 1 \end{bmatrix}} \quad \overset{\text{(b)}}{\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}}$$

Part (a):

(i) 4

(ii) 13

(iii) 26

(iv) 18

Your Answer:
The answer I got for part (a) was (ii)..13.

Part (b):

(i) 0

(ii) 12

(iii) 7

(iv) 4

Your Answer:
The answer I got for part (b) was (i)..0.

My MATLAB code for calculating these answers can be seen below.

```
% Creating function Determinant()..
function D = Determinant(A)
% Getting dimensions of matrix..
[m, n] = size(A);
% If matrix is not square, output error message and return..
if (m ~= n)
    disp("The matrix must be square");
    return;
end
% If 2x2 matrix, get determinant
if (m == 2)
    % For a matrix [a b; c d] the determinant is ad - cb.
    part1 = A(1,1) * A(2,2);
    part2 = A(1,2) * A(2,1);
    D = part1 - part2;

% If 3x3 matrix, get determinant (We can use recursion here)
elseif (m == 3)
    % Part 1, 2 and 3 consists of make 2x2 matrices for recursion.
    part1 = [A(2,2) A(2,3);
             A(3,2) A(3,3)];

    part2 = [A(2,1) A(2,3);
             A(3,1) A(3,3)];

    part3 = [A(2,1) A(2,2);
             A(3,1) A(3,2)];
    % Caclulating determinant...
    part4 = A(1,1) * Determinant(part1);
    part5 = A(1,2) * Determinant(part2);
    part6 = A(1,3) * Determinant(part3);
    D = part4 - part5 + part6;

% If 4x4 matrix, get determinant.. same principle as step above.
elseif (m == 4)
    % Creating 3x3 matrices for recursion..
    part1 = [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)];

    part2 = [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)];

    part3 = [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)];


    part4 = [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)];
    % Calculating determinant.
    part5 = A(1,1) * Determinant(part1);
    part6 = A(1,2) * Determinant(part2);
    part7 = A(1,3) * Determinant(part3);
    part8 = A(1,4) * Determinant(part4);
    D = part5 - part6 + part7 - part8;
end

% Part(a) = 13
% Part(b) = 0
```

0.2 Question 3.2

Question: Determine the root of $f(x) = x - 2e^{-x}$ by:

(a) Using the bisection method. Start with a= 0 and b = 1, and carry out the first three iterations.

(b) Using the secant method. Start with the two points, x1 = 0 and x2 = 1, and carry out the first three iterations.

(c) Using Newton's method. Start at x1 = 1 and carry out the first three iterations.

Part (a):

(i) 0.1241

(ii) 0.08294

(iii) 0.074995

(iv) 0.003462

Your Answer:
Bisection Method: is a bracketing method for finding a numerical solution of an equation of the form $f(x) = 0$ when it is known that withing a given interval $[a, b], f(x)$ is continuous and the equation has a solution.
The algorithm for the bisection method is as follows:

1. Choose first interval by finding points a and b such that a solution exists between them (a and b should have different signs). For us, a and b have been given to us as 0 and 1 respectively.

2. Calculate the first estimate of the numerical solution x_{NS1} by:

$$x_{NS1} = \frac{(a+b)}{2}$$

3. Determine if the solution is between a and x_{NS1} or b and x_{NS1}. This is done by checking the sign of the product $f(a) * f(x_{NS1})$. If the result of this is less than 0, the solution is between a and x_{NS1}, else if the solution is greater than 0, the solution is between x_{NS1} and b.

4. Select the subinterval that contains the true solution and go back to step 2. Step 2 through 4 are repeated until error bound is attained.

Since we have step 1 already done for us we will begin with step 2.

Iteration 0: $x_{NS1} = \frac{(0+1)}{2} = 0.5$. This is our first estimate of our numerical solution. $f(0)*f(0.5) = ((0)-2e^{-(0)})*((0.5)-2e^{-(0.5)}) = -2*-0.7130 = 1.426$. Since this is greater than 0, we know our solution is in between x_{NS1} and b.

Iteration 1: $x_{NS1} = \frac{(0.5+1)}{2} = 0.75$. This is our second estimate of our numerical solution. $f(0.5) * f(0.75) = ((0.5) - 2e^{-(0.5)}) * ((0.75) - 2e^{-(0.75)}) = -0.7130 * -0.1947 = 0.1388$. Since this is greater than 0, we know our solution is in between x_{NS1} and b.

Iteration 2: $x_{NS1} = \frac{(0.75+1)}{2} = 0.875$. This is our second estimate of our numerical solution. $f(0.75) * f(0.875) = ((0.75) - 2e^{-(0.75)}) * ((0.875) - 2e^{-(0.875)}) = -0.1947 * 0.04127 = -0.0080$. Since this is less than 0, we know our solution is in between a and x_{NS1}.

Iteration 3: $x_{NS1} = \frac{(0.75+0.875)}{2} = 0.8125$. This is our second estimate of our numerical solution. $f(0.75) * f(0.8125) = ((0.75) - 2e^{-(0.75)}) * ((0.8125) - 2e^{-(0.8125)}) = -0.1947 * -0.07499 = -0.0146$. Since this is less than 0, we know our solution is in between x_{NS1} and a.

The answer we end up with is 0.8125.. or 0.875 for the iteration before.

Part (b):

(i) 0.72481

(ii) 0.86261

(iii) 0.62849

(iv) 0.17238

Your Answer:
Secant Method: is a scheme for finding a numerical solution of an equation of the form $f(x) = 0$. The method uses two points in the neighborhood of the solution to determine a new estimate for the solution. Two points are used to define a straight line, and the point where the line intersects the x-axis is the new estimate for the solution.
The equation can be generalized to an iteration formula in which a new estimate of the solution x_{i+1} is determined from the previous two solutions x_i and x_{i-1}

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1}-x_i)}{f(x_{i-1})-f(x_i)}$$

Iteration 1: Let $x_i = b..(1)$ and $x_{i-1} = a..(0)$. We first find our next estimate of the solution by subbing into our formula.. $x_{i+1} = 1 - \frac{f(1)(0-1)}{f(0)-f(1)}$, giving us $x_{i+1} = 0.88339$. $f(0.88339) = 0.05663$.

Iteration 2: We now repeat the process for our new estimate of the solution. $x_{i+1} = 0.88339 - \frac{f(0.88339)(1-0.88339)}{f(1)-f(0.88339)}$, giving us $x_{i+1} = 0.85154$. $f(0.85154) = -0.00197$.

Iteration 3: And again.. $x_{i+1} = 0.85154 - \frac{f(0.85154)(0.88339-0.85154)}{f(0.88339)-f(0.85154)}$, giving us $x_{i+1} = 0.85261$. $f(0.85261) = 0.00000833298$.

So our answer is 0.85261 or (ii).. probably some inaccuracies due to rounding.

Part (c):

(i) 0.65782

(ii) 0.59371

(iii) 0.45802

(iv) 0.85261

Your Answer:
Newton's method is a scheme for finding a numerical solution of an equation of the form $f(x) = 0$ where $f(x)$ if continuous and differentiable and the equation is known to have a solution near a given point. The equation can be generalized for determining the "next" solution x_{i+1} from the present solution x_i:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Iteration 1: First easiest to find out what $f'(x)$ is.. $f'(x) = 2e^{-x} + 1$. We know that $x_i = 1$, so we just need to plug it into our formula to get the next solution. $x_{i+1} = 1 - \frac{f(1)}{f'(1)} = 0.848$.

Iteration 2: $x_{i+1} = 0.848 - \frac{f(0.848)}{f'(0.848)} = 0.8433$.

Iteration 3: $x_{i+1} = 0.8433 - \frac{f(0.833)}{f'(0.833)} = 0.852$. $f(0.852) = -0.0011$.

So our answer is 0.852 or (iv).

0.3 Question 4.24

Question: Write a user-defined MATLAB function that determines the inverse of a matrix using the Gauss-Jordan method. For the function name and arguments use Ainv =Inverse (A), where A is the matrix to be inverted, and Ainv is the inverse of the matrix. Use the Inverse function to calculate the inverse of:

$$\underset{\text{The Matrix A}}{\begin{bmatrix} -1 & 2 & 1 \\ 2 & 2 & -4 \\ 0.2 & 1 & 0.5 \end{bmatrix}} \quad \underset{\text{The Matrix B}}{\begin{bmatrix} -1 & -2 & 1 & 2 \\ 1 & 1 & -4 & -2 \\ 1 & -2 & -4 & -2 \\ 2 & -4 & 1 & -2 \end{bmatrix}}$$

(i)

$$\underset{\text{Inverse(a)}}{\begin{bmatrix} -0.7143 & 0.0 & 1.4286 \\ 0.2571 & 0.1000 & 0.2857 \\ -0.2286 & -0.2000 & 0.8571 \end{bmatrix}} \quad \underset{\text{Inverse(b)}}{\begin{bmatrix} 1.6667 & 2.8889 & -2.2222 & 1.0000 \\ 0.0 & 0.3333 & -0.3333 & 0.0 \\ -0.3333 & -0.4444 & 0.1111 & 0.0 \\ 1.5000 & 2.0000 & -1.5000 & 0.5000 \end{bmatrix}}$$

(ii)

$$\underset{\text{Inverse(a)}}{\begin{bmatrix} 0.7243 & 0.0 & 1.3286 \\ 1.2571 & 0.1000 & 0.2757 \\ -0.2386 & -0.2010 & 0.9571 \end{bmatrix}} \quad \underset{\text{Inverse(b)}}{\begin{bmatrix} 1.6677 & 2.9889 & 3.2222 & 1.01700 \\ 0.3433 & -0.3433 & 0.3333 & 0.00371 \\ -0.3433 & -0.2879 & 0.2111 & 0.0 \\ 1.2400 & 2.0120 & -1.5783 & 0.5600 \end{bmatrix}}$$

(iii)

$$\underset{\text{Inverse(a)}}{\begin{bmatrix} 0.7143 & 0.003 & 2.3276 \\ 1.2671 & 0.1100 & 0.3759 \\ -0.2486 & -0.2110 & 0.9771 \end{bmatrix}} \quad \underset{\text{Inverse(b)}}{\begin{bmatrix} 1.6877 & 3.9789 & 3.2002 & 2.01800 \\ 0.3533 & -0.4433 & 0.3333 & 0.02371 \\ -0.3443 & -0.2999 & 0.3121 & 0.0382 \\ 1.2420 & 3.0130 & -1.5733 & 0.5610 \end{bmatrix}}$$

(iv)

$$\underset{\text{Inverse(a)}}{\begin{bmatrix} 0.8343 & 1.01 & 1.3336 \\ 2.2572 & 0.1003 & 0.3857 \\ -0.2486 & -0.2110 & 0.9671 \end{bmatrix}} \quad \underset{\text{Inverse(b)}}{\begin{bmatrix} 1.6777 & 4.9889 & 3.2232 & 1.11700 \\ 0.3443 & -0.3443 & 0.3233 & 0.07371 \\ -0.3443 & -0.2979 & 0.3211 & 0.07800 \\ 1.2480 & 2.1220 & -1.5883 & 0.5621 \end{bmatrix}}$$

Your Answer:
The answer I got was (a)

$$\underset{\text{Inverse(a)}}{\begin{bmatrix} -0.7143 & 0.0 & 1.4286 \\ 0.2571 & 0.1000 & 0.2857 \\ -0.2286 & -0.2000 & 0.8571 \end{bmatrix}} \quad \underset{\text{Inverse(b)}}{\begin{bmatrix} 1.6667 & 2.8889 & -2.2222 & 1.0000 \\ 0.0 & 0.3333 & -0.3333 & 0.0 \\ -0.3333 & -0.4444 & 0.1111 & 0.0 \\ 1.5000 & 2.0000 & -1.5000 & 0.5000 \end{bmatrix}}$$

My MATLAB code for calculating this can be seen below.

```
% Creating function Inverse()
function Ainv = Inverse(A)
% Getting dimensions of matrix
[m, n] = size(A);

% Making sure matrix is square..
if (m ~= n)
    disp("Square matrices only");
end

% Creating empty matrix of same size..
Inv = zeros(m);
% Formatting empty matrix to identity matrix..
for i = 1:m
    Inv(i,i) = 1
end

% Just need 1 for loop for the diagonal..
for i = 1:m
```

```
    % If the diagonal isn't 1
    if A(i,i) ~= 1
        % Divide that row so it is..
        temp = A(i,i);
        for k = 1:m
            A(i,k) = A(i,k) / temp;
            Inv(i,k) = Inv(i,k) / temp;
        end
    end

    % For the all column cells of the diagonal..
    for j = 1:n
       % Make them 0..
       if i ~= j && A(j,i) ~= 0
           temp = A(j,i);
           for k = 1:m
              A(j,k) = A(j,k) - (temp*A(i,k));
              Inv(j,k) = Inv(j,k) - (temp*Inv(i,k));
           end
       end
    end
end
% Rinse and repeat

disp(A);
disp(Inv);
```