Computational Mathematics - Assignment 1

Owen Burke, 15316452

February 21, 2019

Question 2.31

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:

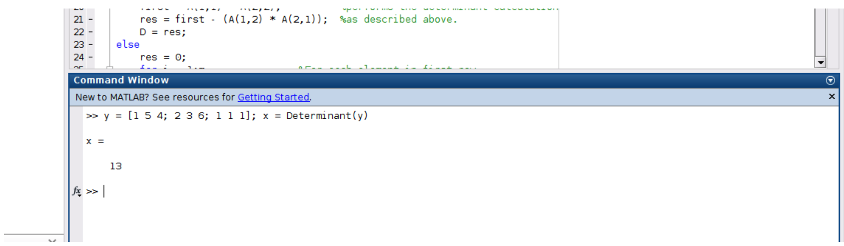

Figure 1: Determinant a)

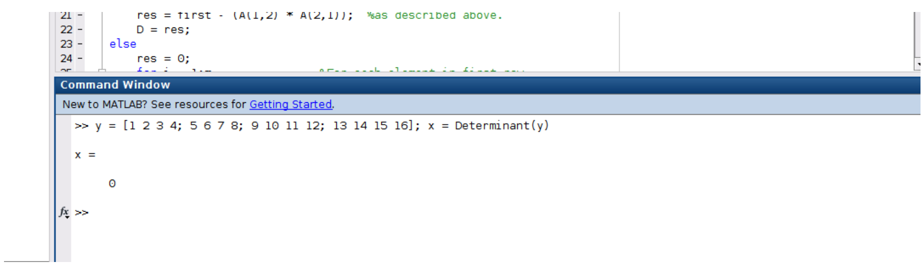

Figure 2: Determinant b)

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

```
function D = Determinant(A)
%Determinant - Returns the determinant of the given matrix
```

```
%   Explanation : This solution is recursive to the point where
%                 you're working with a 2*2 matrix A where the determinant
%                 is simply (A(1,1) * A(2,2)) - (A(1,2) * A(2,1)).
%
%                 For any larger matrix, we take the first row and for each
%                 element in that row we multiply it by the determinant of
%                 the submatrix that is the current matrix with the row and
%                 column of the current element removed. We also multiply
%                 this by -1 if the element coordinates ie. A(1,2), sum up to
%                 be odd (1+2 = 3 which is odd).
%                 We do this for every element in the first row, summing up
%                 the final values for each element which gives us our
%                 determinant
[m,n] = size(A);
if m ~= n                                   %Checks that the matrix is square
    D = "The matrix must be square";   %Returns the error message otherwise
elseif m == 2                               %If matrix is 2*2,
    first = A(1,1) * A(2,2);           %performs the determinant calculation
    res = first - (A(1,2) * A(2,1));   %as described above.
    D = res;
else
    res = 0;
    for i = 1:m                   %For each element in first row
        tempMat = A;              %temp mat forms the submatrix by removing
        tempMat(1,:) = [];        %the row and column of the current element
        tempMat(:,i) = [];
        if mod((1+i),2) == 1      %Find if the coordinates are odd (described above)
```

```
            temp = (A(1,i) * -1) * Determinant(tempMat); % Multiplies the current element by the determinant of the
            res = res + temp;                             % submatrix, recursively, if the coordinates are odd.
        else
            temp = A(1,i) * Determinant(tempMat);    %If they are not odd
            res = res + temp;
        end
    end
    D = res;
end
end
```

Question 3.2

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.

1st iteration:

Given a = 0 and b = 1, find the first numerical solution given by

$$X_{NS1} = \frac{a+b}{2} = \frac{0+1}{2} = 0.5$$

Now, Determine whether the true solution is between a and X_{NS1} or between X_{NS1} and b.
This is done by checking the sign of the product f(a)f(X_{NS1}).
If the product is positive, then the true solution lies between X_{NS1} and b.
Otherwise, it is between a and X_{NS1}.

$$(0\text{-}2e^{-(0)}) * (\tfrac{1}{2}\text{-}2e^{-(\frac{1}{2})}) = 1.426$$

As this result is positive, we know that the true solution lies between X_{NS1} and b, so we make a = X_{NS1} and let b = b and we repeat for the next iteration with the same process.

2nd iteration:

Given a = 0.5, b = 1

$$X_{NS2} = \frac{a+b}{2} = \frac{0.5+1}{2} = 0.75$$

Determine which sub-interval contains the true solution

$$(0.5\text{-}2e^{-(0.5)}) * (0.75\text{-}2e^{-(0.75)}) = 0.139$$

As this result is positive, we know that the true solution lies between X_{NS2} and b, so we make a = X_{NS2} and let b = b and we repeat for the next iteration with the same process.

3rd iteration:

Given a = 0.75, b = 1

$$X_{NS3} = \frac{a+b}{2} = \frac{0.75+1}{2} = \frac{7}{8}$$

Determine which sub-interval contains the true solution

$$(0.75\text{-}2e^{-(0.75)}) * (\tfrac{7}{8}\text{-}2e^{-(\frac{7}{8})}) = \text{-}0.008$$

As this result is negative, we know that the true solution lies between a and X_{NS3}, so we make a = a and let b = X_{NS3} and we get the half way point between these two values as our final numerical solution.

$$X_{FNS} = \frac{a+b}{2} = \frac{0.75+0.875}{2} = 0.8125$$

b) Using the secant method. Start with the two points, x_1 = 0 and x_2 = 1, and carry out the first three iterations.

1st iteration:

Given x_1 = 0 and x_2 = 1,
using the identity (for the slope of the secant containing ($x_1 f(x_1)$), ($x_2 f(x_2)$), ($x_3 f(x_3)$)) :

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

solve for x_3

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

This is given as

$$x_3 = 1 - \frac{(0.264)(1-0)}{(-2)-(0.264)} = 1.117$$

Now we use ($x_2 f(x_2)$), ($x_3 f(x_3)$) as points defining a new secant and repeat step 2.

2^{nd} iteration:

Given $x_2 = 1$ and $x_3 = 1.117$,

solve for x_4

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

This is given as

$$x_4 = 1.117 - \frac{(0.462)(1 - 1.117)}{(0.264) - (0.462)} = 0.844$$

Now we use ($x_3 f(x_3)$), ($x_4 f(x_4)$) as points defining a new secant and repeat step 2.

3^{rd} iteration:

Given $x_3 = 1.117$ and $x_4 = 0.844$,

solve for x_5

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

This is given as

$$x_5 = 0.844 - \frac{(-0.016)(1.117 - 0.844)}{(0.462) - (-0.016)} = 0.853$$

Final numerical solution after three iterations : $x = 0.853$

c) Using Newton's method. Start at $x_1 = 1$ and carry out the first three iterations.

Newton's method approximates the solution initially as the intercept of the tangent to the function, at an initial guess-point, with the x axis. Subsequent iterations approximate the solution as the intercept of the tangent to the function, at the point defined by the previous estimate, with the x axis. Note that it does not necessarily converge.

Given $x_1 = 1$, our starting point is (x_1,$f(x_1)$) = (1, 0.264).
Using the identity for slope of the tangent at (x_1,$f(x_1)$) :

$$f'(x_1) = \frac{f(x_1)-0}{x_1-x_2}$$

Solve for x_2 using :

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

and repeat step 2 iteratively using the formula:

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

To begin we must find f'(x).

$f'(x) = \frac{dy}{dx}$, where y = f(x)

$f'(x) = \frac{d}{dx}\ (x-2e^{-x})$

$f'(x) = \frac{d(x)}{dx} - 2 * \frac{d(e^{-x})}{dx}$

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

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

1^{st} iteration:

$$x_2 = 1 - \frac{0.264}{1+2e^{-(1)}} = 1 - \frac{0.264}{1.736} = \frac{184}{217}$$

2^{nd} iteration:

$$x_3 = \frac{184}{217} - \frac{-0.009}{1.857} = 0.853$$

3^{rd} iteration:

$$x_4 = 0.853 - \frac{0.0007}{1.852} = 0.853$$

Final numerical solution after three iterations : x = 0.853

Question 4.24

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:

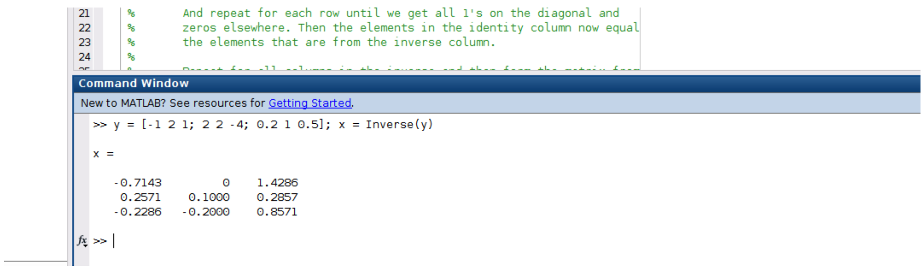

Figure 3: Inverse a)

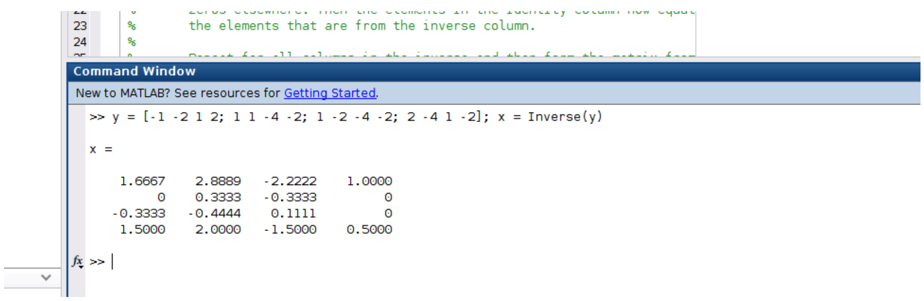

Figure 4: Inverse b)

a) $\begin{bmatrix} -1 & 2 & 1 \\ 2 & 2 & -4 \\ 0.2 & 1 & 0.5 \end{bmatrix}$ **b)** $\begin{bmatrix} -1 & -2 & 1 & 2 \\ 1 & 1 & -4 & -2 \\ 1 & -2 & -4 & -2 \\ 2 & -4 & 1 & -2 \end{bmatrix}$

```
function Ainv = Inverse(A)
%Inverse - Returns the inverse of the given matrix
%  Explanation : This returns the inverse of the given matrix by solving
%          the m equations (for a m*m matrix) where each equation is the matrix A
%          * n-th column of the inverse matrix = the n-th column of the identity matrix
%
%          When each column of the inverse matrix is obtained, they form the
%          inverse of the provided matrix. Each of these equations is solved
```

```
%       using gauss-jordan elimination.
%
%       For each equation, this is done iteratively. For the i-th
%       iteration, we take the element A(i,i). This is our pivot element and
%       we divide each element in the current row (also denoted by the
%       current iteration) by this pivot element, as well as the identity
%       matrix column.
%       Then, for every OTHER row in the matrices (A and the identity), and for each element in
%       those rows, we subtract it from the element in the same column as the pivot in our
%       current row * the element from our current column from the pivot
%       equation.
%
%       And repeat for each row until we get all 1's on the diagonal and
%       zeros elsewhere. Then the elements in the identity column now equal
%       the elements that are from the inverse column.
%
%       Repeat for all columns in the inverse and then form the matrix from
%       the resulting columns



[m,n] = size(A);
if m ~= n
    Ainv = "The matrix must be square";
else
    tempRes = zeros(m,m);
    for i = 1:m
        copy = A;                   %%%%%%%%%%%%Forms identity columns%%%%%%%
        idCol = zeros(m,1);
        idCol(i,1) = 1;
        %disp(idCol);
        for j = 1:m
        %%%%%%%%%%%%%%%%%%%%%%-divide row by pivot-%%%%%%%%%%%%%%%
            pivot = copy(j,j);
```

```
            copy(j,:) = rdivide(copy(j,:),pivot);
            idCol(j,1) = idCol(j,1)/pivot;
        %%%%%%%%%%%%%%%%%%%%%%-eliminate elements from other rows-%%%%%%%%%
            for k = 1:m
                if (k == j)
                    continue; %Skip the row corresponding to the current iteration
                end
                fac = copy(k,j);          %%%%%%Perform the elimination%%%%% Gets the element that will act as the factor
                for c = 1:m
                    first = copy(k,c);    %Gets the element from the current row
                    third = copy(j,c);    %Gets the element from the pivot equation
                    temp = first - (fac*third); %eliminates
                    copy(k,c) = temp;     %write-back
                end
                idCol(k,1) = idCol(k,1) - (fac * idCol(j,1) ); %Perform the operation on the identity column
            end
        end
        tempRes(:,i) = idCol;    %Place the resulting inverse column in the result matrix
    end
    Ainv = tempRes; %return
end
end
```

NAME: Owen Burke

STUDENT NUMBER: 15316452

Please indicate your answers by entering the option ((i), (ii), (iii) or (iv)) where asked. You should append the completed document as a pdf with your type written worked solutions and upload to Blackboard by Friday 22nd of February 2019.

Q 2.31

Part (a):

(i) 4
(ii) 13
(iii) 26
(iv) 18

Your Answer: ii)

Part (b):

(i) 0
(ii) 12
(iii) 7
(iv) 4

Your Answer: i)

Q 3.2

Part (a):

(i) 0.1241
(ii) 0.8125
(iii) 0.074995
(iv) 0.003462

Your Answer: ii)

Part (b):

(i) 0.72481
(ii) 0.85261
(iii) 0.62849
(iv) 0.17238

Your Answer: ii)

Part (c):

(i) 0.65782
(ii) 0.59371
(iii) 0.45802
(iv) 0.85261

Your Answer: iv)

Q 4.24

(i) Inverse(a)=

-0.7143	0.0	1.4286
0.2571	0.1000	0.2857
-0.2286	-0.2000	0.8571

Inverse(b)=

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

(ii)

Inverse(a)=

0.7243	0.0	1.3286
1.2571	0.1000	0.2757
-0.2386	-0.2010	0.9571

Inverse(b)=

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

(iii)

Inverse(a)=

0.7143	0.003	2.3276
1.2671	0.1100	0.3759
-0.2486	-0.2110	0.9771

Inverse(b)=

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

(iv)

Inverse(a)=

0.8343	1.01	1.3336
2.2572	0.1003	0.3857
-0.2486	-0.2110	0.9671

Inverse(b)=

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

Your Answer: i)