BLG 202E – Numerical Methods in CE

Spring 2017

Assignment – 2

Due: 09.04.2017 23:59

Question 1 – Gaussian Elimination and Backward Substitution

$$x_1 - x_2 + 3x_3 = 2$$

$$x_1 + x_2 = 4$$

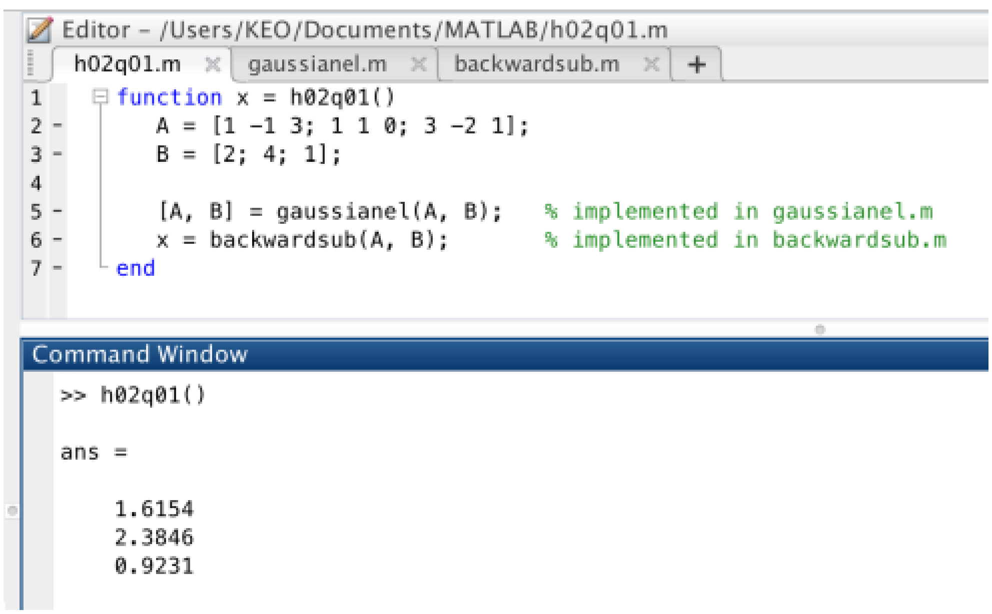
$$3x_1 - 2x_2 + x_3 = 1$$

$$(A \mid b) \Rightarrow \left(\begin{array}{ccc|c} 1 & -1 & 3 & 2 \\ 1 & 1 & 0 & 4 \\ 3 & -2 & 1 & 1 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|c} 1 & -1 & 3 & 2 \\ 0 & 2 & -3 & 2 \\ 0 & 1 & -8 & -5 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|c} 1 & -1 & 3 & 2 \\ 0 & 1 & -8 & -5 \\ 0 & 2 & -3 & 2 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|c} 1 & -1 & 3 & 2 \\ 0 & 1 & -8 & -5 \\ 0 & 0 & 13 & 12 \end{array}\right)$$

(Gaussian elimination)

$$x_3 = \frac{12}{13}, \qquad x_2 = \left(-5 + 8 * \frac{12}{13}\right) = \frac{31}{13}, \qquad x_1 = \left(2 + \frac{31}{13} - 3 * \frac{12}{13}\right) = \frac{21}{13}$$

(Backward substitution)



(Figure 1: Implementation of Gaussian elimination and backward substitution)

The results calculated manually and calculated with matlab are exactly the same. The matlab scripts are attached to zip file (h02q01.m, gaussianel.m, and backwardsub.m)

Question 2 – LU Decomposition

$$Ax = b$$

$$LUx = b$$

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 8 & 14 \\ 2 & 6 & 13 \end{bmatrix}$$

a) $l_{21} = \frac{3}{1} = 3,\ l_{31} = \frac{2}{1} = 2 \ \Rightarrow \ M^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix},\ A^{(1)} = M^{(1)} * A = \begin{pmatrix} 1 & 2 & 4 \\ 0 & 2 & 2 \\ 0 & 2 & 5 \end{pmatrix}$

$$l_{32} = \frac{2}{2} = 1 \ \Rightarrow \ M^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}, \qquad U = A^{(2)} = M^{(2)} * A^{(1)} = \begin{pmatrix} 1 & 2 & 4 \\ 0 & 2 & 2 \\ 0 & 0 & 3 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 2 & 1 & 1 \end{pmatrix}$$

Assume that we have many right-hand side vectors to find to solutions. We have to apply Gaussian elimination method for every right-hand side vector separately which costs $\mathcal{O}(n^3)$ flops and backward substitution which costs $\mathcal{O}(n^2)$. But at LU decomposition method, we just have to apply LU decomposition which costs $\mathcal{O}(n^3)$ flops for ones, and then for each right-hand side vector we apply backward substitution ($\mathcal{O}(n^2)$).

b) The implementation of LU decomposition in matlab is attached to zip file and used in h02q02.m

```
Editor - /Users/KEO/Documents/MATLAB/h02q02.m
h02q02.m  forwardsub.m  backwardsub.m

function  h02q02()
    % (part a): calling mylu function implemented for lu decomposition
    A = [1 2 4; 3 8 14; 2 6 13];
    b1 = [3; 13; 4];
    b2 = [6; 24; 15];
    b3 = [-1; -5; -4];

    % (part b): calculating x vectors with given b vectors
    [l, u] = mylu(A);        % LU decomposition (calculated for ones)

    y1 = forwardsub(l, b1); % implemented in forwardsub.m
    x1 = backwardsub(u, y1) % implemented in backwardsub.m

    y2 = forwardsub(l, b2);
    x2 = backwardsub(u, y2)

    y3 = forwardsub(l, b3);
    x3 = backwardsub(u, y3)
end
```

```
>> h02q02()

x1 =

     3
     4
    -2


x2 =

     2
     4
    -1


x3 =

     1
    -1
     0
```

(Figure 2.1, 2.2: Implementation of LU decomposition and x vectors for given b vectors)

Question 3 – Pivoting

$$A = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -2 \end{bmatrix}$$

a) $PA = LU, \quad U = A^{(3)} = M^{(3)} * P^{(3)} * M^{(2)} * P^{(2)} * M^{(1)} * P^{(1)} * A$

$P^{(1)} = P^{(2)} = M^{(1)} = M^{(2)} = I$ $(identity\ matrix)$, so we do not have to apply these steps.

So we obtain this equation: $U = A^{(3)} = M^{(3)} * P^{(3)} * A$

$$P^{(3)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad P^{(3)} * A = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And at this stage it can be observed $M^{(3)}$ is also identity matrix, so we can delete this from equation.

$$U = A^{(3)} = P^{(3)} * A = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

b)

$$\text{b} = \begin{bmatrix} 26 \\ 9 \\ 1 \\ -3 \end{bmatrix}, \quad Ax = b$$

$$(A \mid b) \Rightarrow \left(\begin{array}{cccc|c} 5 & 6 & 7 & 8 & 26 \\ 0 & 4 & 3 & 2 & 9 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & -2 & -3 \end{array}\right) \Rightarrow \left(\begin{array}{cccc|c} 5 & 6 & 7 & 8 & 26 \\ 0 & 4 & 3 & 2 & 9 \\ 0 & 0 & -1 & -2 & -3 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

(Gaussian elimination: A_{43} is chosen as pivot and exchange rows 4 and 3)

$$x_4 = 1\,, \qquad x_3 = \frac{(-3 - (-2) * 1\,)}{-1} = 1\,,$$

$$x_2 = \frac{9 - 2 * 1 - 3 * 1}{4} = 1, \qquad x_1 = \frac{26 - 8 * 1 - 7 * 1 - 6 * 1}{5} = 1$$

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Question 4 – Data Compression and Truncated SVD

In this question, I wrote a matlab code (h02q04.m) to compute truncated SVD of the image with various ranks. I can be observed that the cuteness of image and r values increase proportionally. Images for all r values were saved in Images folder.

```
function h02q04()
    X = imread('cute.jpg');
    G = rgb2gray(X);
    D = im2double(G);

    [U,S,V] = svd(D);

    for r = 10 : 5 : 30
       reduced = U(:,1:r)*S(1:r,1:r)*V(:,1:r)';
       figure, imshow(reduced); % Copied in Images folder
    end
end
```

Kadir Emre Oto

150140032

otok@itu.edu.tr