Homework 5

Math 3607, Autumn 2021

Marco LoPiccolo

Table of Contents

Problem 1.

Part a.)

1. (Improved triangular substitutions; adapted from **FNC** 2.3.5) If $B \in \mathbb{R}^{n\times p}$ has columns $\mathbf{b}_1, \dots, \mathbf{b}_p$, then we can pose p linear systems at once by writing $AX = B$, where $X \in \mathbb{R}^{n\times p}$ whose jth column $\mathbf{x}_j$ solves $A\mathbf{x}_j = \mathbf{b}_j$ for $j = 1, \dots, p$:

$$A \underbrace{\left[\begin{array}{c|c|c|c} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_p \end{array}\right]}_{=X} = \underbrace{\left[\begin{array}{c|c|c|c} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_p \end{array}\right]}_{=B}.$$

(a) Modify `backsub.m` and `forelim.m` from lecture[1] so that they solve the case where the second input is an $n \times p$ matrix, for $p \geq 1$. Include the programs at the end of your live script.

```
%Programs included at end of livescript
```

Part b.)

(b) If $AX = I$, then $X = A^{-1}$. Use this fact to write a MATLAB function `ltinverse` that uses your modified `forelim` to compute the inverse of a lower triangular matrix. Include the program at the end of your live script. Then test your function using the following matrices, that is, compare your numerical solutions against the given exact solutions.

$$L_1 = \begin{bmatrix} 2 & 0 & 0 \\ 8 & -7 & 0 \\ 4 & 9 & -27 \end{bmatrix}, \qquad L_1^{-1} = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ \frac{4}{7} & -\frac{1}{7} & 0 \\ \frac{50}{189} & -\frac{1}{21} & -\frac{1}{27} \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{3} & 1 & 0 & 0 \\ 0 & \frac{1}{3} & 1 & 0 \\ 0 & 0 & \frac{1}{3} & 1 \end{bmatrix}, \qquad L_2^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 & 0 \\ \frac{1}{9} & -\frac{1}{3} & 1 & 0 \\ -\frac{1}{27} & \frac{1}{9} & -\frac{1}{3} & 1 \end{bmatrix}$$

```
format rat
L1 = [2 0 0; 8 -7 0; 4 9 -27]
```

```
L1 =
       2              0              0
       8             -7              0
       4              9            -27
```

```
X = ltInverse(L1)
```

```
X =
       1/2            0              0
       4/7           -1/7            0
      50/189         -1/21          -1/27
```

```
L2 = [1 0 0 0; (1/3) 1 0 0; 0 (1/3) 1 0; 0 0 (1/3) 1]
```

```
L2 =
       1              0              0              0
       1/3            1              0              0
       0              1/3            1              0
       0              0              1/3            1
```

```
X = ltInverse(L2)
```

```
X =
       1              0              0              0
      -1/3            1              0              0
       1/9           -1/3            1              0
      -1/27           1/9           -1/3            1
```

Problem 2.

Part a.)

2. (Triangular substitution and stability; **FNC** 2.3.6) Consider the following linear system $A\mathbf{x} = \mathbf{b}$:

$$\underbrace{\begin{bmatrix} 1 & -1 & 0 & \alpha-\beta & \beta \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \alpha \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{\mathbf{b}}.$$

(a) Show that $\mathbf{x} = (1, 1, 1, 1, 1)^{\mathrm{T}}$ is the solution for any α and β.

a) $$\underbrace{\begin{bmatrix} 1 & -1 & 0 & \alpha-\beta & \beta \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} \alpha \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{b}$$

$x = (1,1,1,1,1)^T$

$1 \cdot x_5 = 1$

$x_5 = 1/1 \qquad x_5 = 1$

$1 \cdot x_4 + -1 \cdot x_5 = 0$

$x_4 = x_5 \qquad x_5 = 1$

$1 = 1 \qquad x_4 = 1$

$1 \cdot x_3 + -1 \cdot x_4 + 0 \cdot x_5 = 0$

$1 \cdot x_3 = 1 \cdot x_4 \qquad x_3 = 1 \qquad x_4 = 1$

$1 = 1$

$1 \cdot x_2 + -1 \cdot x_3 + 0 \cdot x_4 + 0 \cdot x_5 = 0$

$x_2 - x_3 = 0$

$x_2 = x_3 \qquad x_2 = 1 \qquad x_3 = 1$

$1 = 1$

$1 \cdot x_1 + -1 \cdot x_2 + 0 \cdot x_3 + (\alpha-\beta)x_4 + \beta \cdot x_5 = \alpha$

$x_1 = x_2 + -(\alpha-\beta) \cdot x_4 + -\beta \cdot x_5 + \alpha$

$x_1 = x_2 + (-\alpha+\beta) \cdot x_4 + -\beta \cdot x_5 + \alpha \qquad x_2 = 1 \quad x_4 = 1$

$x_1 = 1 + \cancel{-\alpha} + \cancel{\beta} \cancel{-\beta} + \cancel{\alpha} \qquad x_5 = 1$

$x_1 = 1 \qquad x_1 = 1$

$1 = 1$

Part b.)

(b) Using MATLAB, solve the system with $\alpha = 0.1$ and $\beta = 10, 100, \ldots, 10^{12}$, making a table of the values of β and $|x_1 - 1|$. Write down your observation.

```
format long g
alpha = 0.1;
beta = 10 .^ [1:12];
b = [alpha 0 0 0 1]';
fprintf('Beta                  |x1 - 1|')
```

```
Beta                      |x1 - 1|
```

```
fprintf('-------------------------------')
```

```
-------------------------------
```

```
for i = 1:12
    A = [1 -1 0 alpha-beta(:,i) beta(:,i);
         0 1 -1 0 0; 0 0 1 -1 0;
         0 0 0 1 -1;
         0 0 0 0 1];
    x = A \ b;
    fprintf('%13d   %12d\n',B(:,i), abs(x(1)-1))
end
```

```
           10              0
          100              0
         1000              0
        10000              0
       100000              0
      1000000              0
     10000000              0
    100000000              0
   1000000000              0
  10000000000              0
 100000000000              0
1000000000000              0
```

```
for i = 1:12
    A = [1 -1 0 alpha-beta(:,i) beta(:,i);
         0 1 -1 0 0; 0 0 1 -1 0;
         0 0 0 1 -1;
         0 0 0 0 1];
    x = backsub(A, b);
    fprintf('%13d   %12.12f\n',B(:,i), abs(x(1)-1))
end
```

```
           10   0.000000000000
          100   0.000000000000
         1000   0.000000000000
        10000   0.000000000000
       100000   0.000000000006
      1000000   0.000000000023
     10000000   0.000000000373
    100000000   0.000000005960
   1000000000   0.000000023842
  10000000000   0.000000381470
 100000000000   0.000006103516
1000000000000   0.000024414063
```

As we see here in the two loops the basic \ loop gives us zeroes back while the second does not, this is because catastrophic cancellation is occurring in the first loop which leads to some of the values after the decimal being lost and it just gives us zero back versus the second using backsub which gives us more decimal values back

Problem 3.

Part a.)

3. (Vectorizing `mylu.m`; **FNC** 2.4.7) Below is an instructional version of LU factorization code presented in lecture.

```
function [L,U] = mylu(A)
% MYLU   LU factorization (demo only--not stable!).
% Input:
%    A     square matrix
% Output:
%    L,U  unit lower triangular and upper triangular such that LU=A
  n = length(A);
  L = eye(n);    % ones on diagonal
  % Gaussian elimination
  for j = 1:n-1
    for i = j+1:n
      L(i,j) = A(i,j) / A(j,j);    % row multiplier
      A(i,j:n) = A(i,j:n) - L(i,j)*A(j,j:n);
    end
  end
  U = triu(A);
end
```

Consider the innermost loop. Since the different iterations in i are all independent, it is possible to *vectorize* this group of operations, that is, rewrite it without a loop. In fact, the necessary changes are to delete the keyword `for` in the inner loop, and delete the following `end` line. (You should also put a semicolon at the end of `i = j+1:n` to suppress extra output.)

(a) Make the changes as directed and verify that the function works properly.

```
A = [2 2 1; -4 6 1; 5 -5 3];
%Check to make sure function works properly
[L, U] = mylu(A)
```

```
L = 3×3
     1     0     0
    -2     1     0
   2.5    -1     1
U = 3×3
     2     2     1
     0    10     3
     0     0   3.5
```

```
[L, U] = myluMod(A)
```

```
L = 3×3
     1     0     0
```

```
     -2      1      0
    2.5     -1      1
U = 3×3
      2      2      1
      0     10      3
      0      0    3.5
```

```
%Functions found at end of livescript
```

Part b.)

(b) Write out symbolically (*i.e.*, using ordinary elementwise vector and matrix notation) what the new version of the function does in the case $n = 5$ for the iteration with $j = 3$.

b.) $n=5 \qquad j=3$

<u>Code</u>

$i = 4:5$

$L(i,3) = A(i,3) / A(3,3)$

$A(i,3:5) = A(i,3:5) - L(i,3) * A(3,3:5)$

<u>Math version</u>

$$L_{4:5,3} = \begin{bmatrix} L_{4,3} \\ L_{5,3} \end{bmatrix} \qquad A_{4:5,3} = \begin{bmatrix} A_{4,3} \\ A_{5,3} \end{bmatrix}$$

$$L_{4:5,3} = \begin{bmatrix} L_{4,3} / A_{33} \\ L_{5,3} / A_{33} \end{bmatrix}$$

$$A_{4:5,3:5} = \begin{bmatrix} A_{4,3} & A_{4,4} & A_{4,5} \\ A_{5,3} & A_{5,4} & A_{55} \end{bmatrix}$$

$$L_{4:5,3} = \begin{bmatrix} L_{4,3} \\ L_{5,3} \end{bmatrix} \quad A_{3,3:5} = \begin{bmatrix} A_{3,3} & A_{3,4} & A_{3,5} \end{bmatrix}$$

$$A_{4:5,3:5} = \begin{bmatrix} A_{4,3} & A_{4,4} & A_{4,5} \\ A_{5,3} & A_{5,4} & A_{55} \end{bmatrix} - \begin{bmatrix} L_{4,3} \\ L_{5,3} \end{bmatrix} \cdot \begin{bmatrix} A_{3,3} & A_{3,4} & A_{3,5} \end{bmatrix}$$

$$= \begin{bmatrix} A_{4,3} & A_{4,4} & A_{4,5} \\ A_{5,3} & A_{5,4} & A_{55} \end{bmatrix} - \begin{bmatrix} L_{4,3} \cdot A_{3,3} & L_{4,3} \cdot A_{3,4} & L_{4,3} \cdot A_{3,5} \\ L_{5,3} \cdot A_{33} & L_{5,3} \cdot A_{3,4} & L_{5,3} \cdot A_{3,5} \end{bmatrix}$$

$$= \begin{bmatrix} A_{4,3} - (L_{2,3} \cdot A_{33}) & A_{4,4} - (L_{4,3} \cdot A_{3,4}) & A_{4,5} - (L_{4,3} \cdot A_{3,5}) \\ A_{5,3} - (L_{5,3} \cdot A_{33}) & A_{5,3} - (L_{5,3} \cdot A_{3,4}) & A_{5,3} - (L_{5,3} \cdot A_{3,5}) \end{bmatrix}$$

Problem 4.

Part a.)

4. (Application of LU factorization: **FNC** 2.4.6) When computing the determinant of a matrix by hand, it is common to use cofactor expansion and apply the definition recursively. But this is terribly inefficient as a function of the matrix size.

   (a) ✏ Explain why, if $A = LU$ is an LU factorization,

$$\det(A) = u_{11}u_{22}\cdots u_{nn} = \prod_{i=1}^{n} u_{ii}.$$

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ u_{21} & u_{22} & u_{23} & \cdots & u_{2n} \\ u_{31} & u_{32} & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & & \\ u_{n1} & u_{n2} & u_{n3} & \cdots & u_{nn} \end{bmatrix}$$

$$\det(LU) = \det(L) \cdot \det(U)$$

Lower triangular

upper triangular

Go to LU factorization

L is a specific lower triangular matrix

$A = LU$

$\det(A) = \det(L)\det(U)$

unit lower triangular matrix

$\det(L) = 1 \cdot 1 \cdots 1 = 1$

$\det(A) = \det(U)$

$\det(T) = t_{11} t_{22} \cdots t_{nn} = \prod_{i=1}^{n} t_{ii}$ where $T \in \mathbb{R}^{n \times n}$ is a triangular matrix so

$$\det(U) = u_{11} \cdot u_{22} \cdots\cdots u_{nn} = \det(A)$$

so

if $A = LU$ is an LU factorization

with

$$\det(A) = u_{11} u_{22} \cdots u_{nn} = \prod_{i=1}^{n} u_{ii}$$

Part b.)

(b) Using the result of part (a), write a MATLAB function `determinant` that computes the determinant of a given matrix `A` using `mylu` from lecture. Include the function at the end of your live script. Use your function and the built-in `det` on the matrices `magic(n)` for $n = 3, 4, \ldots, 7$, and make a table (using `disp` or `fprintf`) showing n, the value from your function, and the relative error when compared to `det`.

```
format long g
fprintf('n               det                          determinant                relative error')
```

```
n              det                          determinant              relative error
```

```
for n = 3:7
    y = magic(n);
    x = det(y);
    z = determinant(y);
    relerr = (z - x) / x;
    fprintf('%1d    %26.12f    %26.12f   %15.12f\n', n, x, z, relerr)
end
```

```
3            -360.000000000000            -360.000000000000   -0.000000000000
4               0.000000000001               0.000000000000   -0.294117647059
5         5069999.999999999069         5069999.999999997206   -0.000000000000
6               0.000000001150               0.000000000000   -1.000000000000
7     -348052801599.999938964844    -348052801600.000122070312    0.000000000000
```

Problem 5.

5. (Proper usage of `lu`; **FNC** 2.6.1) Suppose that $A \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$. On the left is correct MATLAB code to solve $A\mathbf{x} = \mathbf{b}$; on the right is similar but incorrect code. Explain using mathematical notation exactly what vector is found by the right-hand version.

```
[L,U] = lu(A);
x = U \ ( L \ b );
```

```
[L,U] = lu(A);
x = U \ L \ b;
```

[L, U] = lu(A);
x = U \ (L \ b);

This is finding

$L^{-1}b$

before finding

$U^{-1}(L^{-1}b)$

[L, U] = lu(A)
x = U \ L \ b;

This is equal to

(U \ L) \ b

which is first finding

$U^{-1}L$

and then it finds

$(U^{-1}L)^{-1}b$

Which means these are not equal

$$\boxed{U^{-1}(L^{-1}b) \neq (U^{-1}L)^{-1}b}$$

Problem 6.

6. (FLOP Counting) Do **LM** 10.1–12(a,b,d). Justify your calculation of p and c for each part.

LM 10.1-12 (a, b, d) justify calculation of p and c for each part.

a.) $x = ABCDb$

$ABC(Db)$

Db = matrix vector $\sim 2n^2$ flops $= \vec{u}$

$AB(C\vec{u})$

$C\vec{u} = 2n^2$ flops $= \vec{v}$

$(B\vec{v}) = 2n^2$ flops $= \vec{w}$

$(A\vec{w}) = 2n^2$ flops

$$\boxed{\begin{array}{c} 8n^2 \text{ flops} \\ x = (A(B(C\cdot(D\cdot b)))) \end{array}}$$

b.) $x = BA^{-1}b$

$\vec{v} = A^{-1}\cdot b = \frac{2}{3}n^3$ flops

$\Rightarrow B\cdot\vec{v} = 2n$ flops

$$\boxed{\begin{array}{c} \frac{2}{3}n^3 + 2n \text{ flops} \\ x = B\cdot(A\backslash b) \end{array}}$$

c.) $x = B(C+A)^{-1}b$

1 $C+A$

$n\times n + n\times n$

n^2 flops

② $C+A = M$

$\vec{v} = M^{-1}\vec{b}$: $\sim \frac{2}{3}n^3$ flops

③ $\vec{x} = B\cdot\vec{v}$: $\sim 2n^2$ flops

$$\boxed{\begin{array}{c} \frac{2}{3}n^3 + 3n^2 \text{ flops} \\ x = B\cdot\left((C+A)\backslash b\right) \end{array}}$$

d.) $x = B^{-1}(C+A)\cdot b$

① $C+A = M = n^2$ flops

② $\vec{v} = M\cdot b = 2n^2$ flops

③ $B^{-1}\cdot\vec{v} = \frac{2}{3}n^3$ flops

$$\boxed{\begin{array}{c} \frac{2}{3}n^3 + 3n^2 \text{ flops} \\ x = B\backslash\left((C+A)\cdot b\right) \end{array}}$$

e.) $x = B^{-1}(C+A)^{-1}b$

① $M = C+A = n^2$ flops

② $\vec{v} = M^{-1}\cdot b = \frac{2}{3}n^3$ flops

③ $B^{-1}\cdot\vec{v} = \frac{2}{3}n^3$ flops

$$\boxed{\begin{array}{c} \frac{4}{3}n^3 + n^2 \text{ flops} \\ x = B\backslash\left((C+A)\backslash b\right) \end{array}}$$

f.) $x = B^{-1}(2A^{-1} + I)(C^{-1} + A)^{-1} b$

① $2 \cdot A^{-1} = M = n^2$ flops

② $K = M + I = n^2$ flops

③ $N = C^{-1} + A = n^2$ flops

④ $\vec{v} = N^{-1} \cdot b = \frac{2}{3} n^3$ flops

⑤ $\vec{u} = K \cdot b = 2n^2$ flops

⑥ $B^{-1} \cdot \vec{u} = \frac{2}{3} n^3$ flops

$$\boxed{\begin{array}{c} \frac{4}{3} n^3 + 5n^2 \text{ flops} \\ x = B \backslash \left((2 \cdot A^{-1} + I) \left((C^{-1} + A) \backslash b \right) \right) \end{array}}$$

Problem 7.

Part a.)

7. (Matrix norms; Sp20 midterm) Let

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}.$$

(a) ✏ Calculate $\|A\|_1$, $\|A\|_2$, $\|A\|_\infty$, and $\|A\|_F$ all by hand.

a.) $A = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}$

$\|A\|_1$, $\|A\|_2$, $\|A\|_\infty$, $\|A\|_F$

- $\|A\|_1$ = max column $\underset{(abs)}{\text{sum}}$ $\Rightarrow$

  $\Rightarrow$ col. 1 sum = 1

  $\Rightarrow$ col. 2 sum = 5

  $\Rightarrow \max\{1, 5\} = 5$

  $\boxed{\|A\|_1 = 5}$

- $\|A\|_\infty$ = max row $\underset{(abs)}{\text{sum}}$ $\Rightarrow$

  $\Rightarrow$ row 1 sum = 3

  $\Rightarrow$ row 2 sum = 3

  $\Rightarrow \max\{3, 3\} = 3$

  $\boxed{\|A\|_\infty = 3}$

- $\|A\|_2$ = sqrt. of max eigenvalue of $A^T A$

  $A^T = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}$

  $A^T \cdot A = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 + 0 \cdot 0 & 1 \cdot 2 + 0 \cdot 3 \\ 2 \cdot 1 + 3 \cdot 0 & 2 \cdot 2 + 3 \cdot 3 \end{bmatrix}$

  $B = \begin{bmatrix} 1 & 2 \\ 2 & 13 \end{bmatrix}$

Eigenvalues of $B = A^T \cdot A$
(characteristic equation of B)

$$= \det(\lambda I - B)$$

$$= \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 2 & 13 \end{bmatrix}$$

$$\Rightarrow \det \begin{bmatrix} \lambda - 1 & -2 \\ -2 & \lambda - 13 \end{bmatrix} = (\lambda-1)(\lambda-13) - (-2 \cdot -2)$$

$$= \lambda^2 - 13\lambda - \lambda + 13 - 4$$

$$= \lambda^2 - 14\lambda + 9$$

$$\lambda = \frac{-(-b) \pm \sqrt{b^2 - 4ac}}{2a}$$

$$= \frac{14 \pm \sqrt{196 - 4(1)(9)}}{2}$$

$$\text{Max}(\lambda) = \max\left(\frac{14 \pm \sqrt{160}}{2}\right)$$

$$\max(\lambda) = \frac{14 + \sqrt{160}}{2}$$

$$\boxed{\|A\|_2 = \sqrt{\frac{14 + \sqrt{160}}{2}}}$$

$$\|A\|_F = \sqrt{\sum a_{ij}^2} = 1^2 + 2^2 + 0^2 + 3^2 = \boxed{\sqrt{14}}$$

Part b.)

(b) Imagine that MATLAB does not offer `norm` function and you are writing one for others to use, which begins with

```
function MatrixNorm(A, j)
% MatrixNorm   computes matrix norms
% Usage:
%   mat_norm(A, 1) returns the 1-norm of A
%   mat_norm(A, 2) is the same as mat_norm(A)
%   mat_norm(A, 'inf') returns the infinity-norm of A
%   mat_norm(A, 'fro') returns the Frobenius norm of A
```

Complete the program. (*Hint*: To handle the second input argument properly which can be a number or a character, use `ischaracter` and/or `strcmp`.)

```
A = [1 2;
     0 3]
```

```
A = 2×2
     1     2
     0     3
```

```
A1 = mat_norm(A, 1)
```

```
A1 =
     5
```

```
B1 = mat_norm(A, 2)
```

```
B1 =
          3.65028153987288
```

```
C1 = mat_norm(A, 'inf')
```

```
C1 =
     3
```

```
D1 = mat_norm(A, 'Afro')
```

```
D1 =
          3.74165738677394
```

```
A2 = norm(A, 1)
```

```
A2 =
     5
```

```
B2 = norm(A, 2)
```

```
B2 =
          3.65028153987288
```

```
C2 = norm(A, Inf)
```

```
C2 =
```

```
3
```

```
D2 = norm(A, 'fro')
```

```
D2 =
       3.74165738677394
```

Function backsub

```
function X = backsub(U,B)
% BACKSUB x = backsub(U,B)
% Solve an upper triangular linear system.
% Input:
% U upper triangular square matrix (n by n)
% B right-hand side vectors concatenated into an (n-by-p) matrix
% Output:
% X solution of UX = B (n-by-p)
[n,p] = size(B); % grab dimensions
X = zeros(n,p); % preallocate output
for j = 1:p
    for i = n:-1:1
        X(i,j)= ( B(i,j) - U(i,i+1:n)*X(i+1:n,j))/ U(i,i);
    end
end
end
```

Function forelim

```
function X = forelim(L,B)
% % FORELIM x = forelim(L,B)
% % Solve a lower triangular linear system.
% % Input:
% % L lower triangular square matrix (n by n)
% % B right-hand side vector (n by p)
% % Output:
% % X solution of Lx=b (n by p matrix)
[n, p] = size(B);
X = zeros(n,p);
for j = 1:p
    for i = 1:n
        X(i,j) = (B(i,j) - L(i, 1:i-1) * X(1:i-1,j)) / L(i,i);
    end
end
end
```

Function ltinverse

```
function X = ltInverse(L)
% % FORELIM X = ltInverse(L)
```

```
% % Solve creates the inverse of a lower triangular matrix
% % Input:
% % L lower triangular square matrix (n by n)
% % Output:
% % X solution of X = A^-1
I = eye(length(L));
X = forelim(L, I);
end
```

Function myLU (from Lecture and is used to verify modified function)

```
function [L,U] = mylu(A)
% MYLU LU factorization (demo only--not stable!).
% Input:
% A square matrix
% Output:
% L,U unit lower triangular and upper triangular such that LU=A
n = length(A);
L = eye(n); % ones on diagonal
% Gaussian elimination
for j = 1:n-1
    for i = j+1:n
        L(i,j) = A(i,j) / A(j,j); % row multiplier
        A(i,j:n) = A(i,j:n) - L(i,j)*A(j,j:n);
    end
end
U = triu(A);
end
```

Function myLUmodified

```
function [L,U] = myluMod(A)
% MYLU LU factorization (demo only--not stable!).
% Input:
% A square matrix
% Output:
% L,U unit lower triangular and upper triangular such that LU=A
n = length(A);
L = eye(n); % ones on diagonal
% Gaussian elimination
for j = 1:n-1
    i = j+1:n;
    L(i,j) = A(i,j) / A(j,j);  % row multiplier
    A(i,j:n) = A(i,j:n) - L(i,j)*A(j,j:n);
end
U = triu(A);
end
```

Function determinant

```
function x = determinant(A)
% determinant computes the determinant of a matrix A = LU of an LU
% factorization by first using myLU to find the upper triangular matrix
```

```
% input:
% A square matrix
% output:
% determinant of the matrix A
[~, U] = mylu(A); %gives us the upper triangular matrix of matrix A
x = prod(diag(U));

end
```

Function Matrix Norm

```
function maxVal = mat_norm(A, j)
% % MatrixNorm computes matrix norms
% % Usage:
% % mat_norm(A, 1) returns the 1-norm of A
% % mat_norm(A, 2) is the same as mat_norm(A)
% % mat_norm(A, 'inf') returns the infinity-norm of A
% % mat_norm(A, 'Afro') returns the Frobenius norm of A

if ischar(j) == 1
    if strcmp(j,'inf') == 1
        maxVal = max(sum(abs(A), 2));
    end
    if strcmp(j,'Afro') == 1
        maxVal = sqrt(A(:)'*A(:));
    end
else
    if j == 1
        maxVal = 0;
        for i = 1:length(A)
            x = sum(A(:,i));
            if x > maxVal
                maxVal = x;
            end
        end
    end
    if j == 2
        maxVal = max(sqrt(eig(A'*A)));
    end
end

end
```