Lab Session 2

MA-571 : Numerical Linear Algebra Lab 2022 R. Alam

MATLAB Scripts, functions and plotting

1. Write a MATLAB function that implements GENP. Your function should look like the following.

```
    function [L, U] = GENP(A);
    % [L U] = GENP(A) produces a unit lower triangular matrix L and an upper
    % triangular matrix U so that A= LU.

    [n, n] = size(A);
    for k = 1:n-1
       % compute multipliers for k-th step
       A(k+1:n,k) = A(k+1:n,k)/A(k,k);
       % update A(k+1:n,k+1:n)
       j = k+1:n;
       A(j,j) = A(j,j)-A(j,k)*A(k,j);
    end
    % strict lower triangle of A, plus I
    L = eye(n,n)+ tril(A,-1);
    U = triu(A); % upper triangle of A
```

   Next, write a MATLAB function that implements GEPP. Your function should look like the following.

```
function [L, U, p] = GEPP(A);
% [L U, p] = GEPP(A) produces a unit lower triangular matrix L, an upper
% triangular matrix U and a permutation vector p, so that A(p,:)= LU.

[n, n] = size(A);
p = (1:n)';
for k = 1:n-1

    % find largest element in A(k:n,k)
    [r, m] = max( abs( A(k:n,k) ) );
    m = m+k-1;
    if (m ~=k) % swap row
       A([k m],:)  = A([m k],:);
       p([k m]) = p([m k]);
    end
       if (A(k,k) ~= 0)
```

```
            % compute multipliers for k-th step
            A(k+1:n,k) = A(k+1:n,k)/A(k,k);
            % update A(k+1:n,k+1:n)
            j = k+1:n;
            A(j,j) = A(j,j)-A(j,k)*A(k,j);
        end
    end
    % strict lower triangle of A, plus I
    L = eye(n,n)+ tril(A,-1);
    U = triu(A); % upper triangle of A
```

Now modify GEPP to write a new function GEPP2 that also computes determinant of A.

```
function [L, U, p, d] = GEPP2(A);
% [L, U, p, d] = GEPP(A) produces a unit lower triangular matrix L, an upper
% triangular matrix U and a permutation vector p, so that A(p,:)= LU
% and d = det(A).
```

Finally write a MATLAB function that implements GECP

```
function [L, U, p, q] = GECP(A);
% [L, U, p, q] = GECP(A) produces a unit lower triangular matrix L,
% an upper triangular matrix U and two permutation vectors p and q,
% so that A(p,q)= LU.
```

Consider $A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}$. Compute `[L, U] = GENP(A)`and define `E = L*U-A`. Also compute `[L, U, p] = GEPP(A)` and compute `F = LU - A(p,:)`. Now compute the norms of `E` and `F`. What is conclusion about GENP and GEPP from this experiment?

2. In the 1999 movie Office Space, a character creates a program that takes fractions of cents that are truncated in a bank's transactions and deposits them to his own account. This is not a new idea, and hackers who have actually attempted it have been arrested. In this exercise we will simulate the program to determine how long it would take to become a millionaire this way.

   Assume that we have access to 50,000 bank accounts. Initially we can take the account balances to be uniformly distributed between, say, \$100 and \$100,000. The annual interest rate on the accounts is 5%, and interest is compounded daily and added to the accounts, except that fractions of a cent are truncated. These will be deposited to an illegal account that initially has balance \$0.

   Write a MATLAB program that simulates the Office Space scenario. You can set up the initial accounts with the commands

```
accounts = 100 + (100000-100) * rand(50000,1);
% Sets up 50,000 accounts with balances between $100 and $100000.
accounts = floor(100 * accounts)/100;
% Deletes fractions of a cent from initial balances.
```

   (a) Write a MATLAB program that increases the accounts by (5/365)% interest each day, truncating each account to the nearest penny and placing the truncated amount into an account, which we will call the illegal account. Assume that the illegal account can hold fractional amounts (i.e., do not truncate this account's values) and let the illegal account also accrue daily interest. Run your code to determine how many days it would take to become a millionaire assuming the illegal account begins with a balance of zero.

(b) Without running your MATLAB code, answer the following questions: On average, about how much money would you expect to be added to the illegal account each day due to the embezzlement? Suppose you had access to 100,000 accounts, each initially with a balance of, say, \$5000. About how much money would be added to the illegal account each day in this case? Explain your answers.

Note that this type of rounding corresponds to fixed-point truncation rather than floating-point, since only two places are allowed to the right of the decimal point, regardless of how many or few decimal digits appear to the left of the decimal point.

3. The roots of a quadratic polynomial $p(x) := ax^2+bx+c$ is given by $(-b\pm\sqrt{b^2-4ac})/2a$. Write a MATLAB function that implements the above formula to compute the roots. Your function will look like this:

```
function [x1, x2] = quadroot1( a, b, c)
d = sqrt( b^2 - 4 * a* c );
x1 = (-b + d) / (2*a);
x2 = (-b - d) / (2*a);
```

The largest (in magnitude) root of p can be computed as $x_1 = (-b-\text{sign}(b)\sqrt{b^2-4ac})/2a$ and the second root x_2 can be computed from the identity $x_1x_2 = c/a$. Write a MATLAB function that implements the modified method to compute the roots. Your function will look like this:

```
function [x1, x2] = quadroot2(a, b, c)
d = sqrt( b^2 - 4 * a* c );
x1 = (-b - sign(b) * d) / (2*a);
x2 = c/( a * x1 );
```

Find the roots of $x^2-(10^7+10^{-7})x+1)$ using `quadroot1` and `quadroot2`. Do you observe any difference? Which method is better and why?

4. The machine epsilon `eps` of a floating point system is the distance from 1 to the next floating number bigger than 1 and `u = eps/2` is the unit roundoff (default). You can compute `eps` and `u` in MATLAB by writing a small script. What is it that the following MATLAB script computes?

```
x = 2;
while x > 1
x = x/2
end
```

On the other hand, if the condition $x > 1$ in the above script is replaced with $1 + x > 1$ then what will be the output?

5. This exercise illustrates the difficulty in handling polynomials in finite precision computation. Consider the polynomial

$$\begin{aligned} p(x) = (x-2)^9 \quad &= \quad x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 \\ &\quad\; -4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512 \end{aligned}$$

Write a MATLAB script to evaluate p at 151 equidistant points (use `linspace` command) in the interval [1.95, 2.05] using two methods:

   (a) Apply Horner's method, or call MATLAB function `polyval` (Type `help polyval` for more info).

   (b) Calculate $p(x) = (x-2)^9$ directly.

   Plot these results in two separate figures. For example, if x is a row vector of points in the given interval then the commands

```
>> y = p(x); plot(x, y)
```

   will do the job. Do the plots differ from one another? If yes, can you think of possible reasons?

*** End ***