Lab Session 9

MA-581 : Numerical Computations Lab R. Alam Date: October 11, 2021

While writing program you should try to avoid the use of *for loops* and use MATLAB vector notations wherever possible.

1. 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.

2. Let A be a random matrix generated by `rand(8)`. Find the maximum values (a) in each column, (b) in each row, and (c) overall. Also use `find` to find the row and column indices of all elements that are larger than 0.25.

3. A *magic square* is an n-by-n matrix in which each integer $1, 2, \ldots, n^2$ appears once and for which all the row, column, and diagonal sums are identical. MATLAB has a command `magic` that returns magic squares. Check its output for a few values of n and use MATLAB to verify the summation property. (The antidiagonal sum will be the trickiest. Look for help on how to "flip" a matrix.)

4. Consider the magic square `A = magic(n)` for `n = 3, 4, or 5`. What does

   ```
   p = randperm(n); q = randperm(n); A = A(p,q);
   ```

do to

```
sum(A), sum(A')', sum(diag(A)), sum(diag(flipud(A))), rank(A)
```

The magic square `A = magic(4)` is singular. What do

```
null(A), null(A,'r'), null(sym(A)), and rref(A)
```

tell you about linear dependence of columns of `A`?

5. Are the following true or false? Assume A is a generic n-by-n matrix.
(a) `A^(-1)` equals `1/A` (b) `A.^(-1)` equals `1./A`

6. Do not use `for loop` for this example.

   (a) Use the MATLAB commands `eye`, `ones` and `tril` to generate the Wilkinson matrix W which is defined as follows:

$$W_{ij} = \begin{cases} -1 & \text{if } i > j \\ 1 & \text{if } i = j \text{ or } j = n \\ 0 & \text{otherwise} \end{cases}$$

   (b) A real square matrix $H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & -H_{11}^\top \end{bmatrix}$ is said to be Hamiltonian if $H_{12}^\top = H_{12}$ and $H_{21}^\top = H_{21}$. Generate an $n \times n$ Hamiltonian matrix H for $n = 10, 20$.

7. (a) Look up `diag` in the online help and use it (more than once) to build the 16-by-16 matrix

$$D = \begin{bmatrix} -2 & 1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & -2 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}$$

   (b) Now read about `toeplitz` and use it to build D.

   (c) Use toeplitz and whatever else you need to build

$$\begin{bmatrix} 1 & 2 & 3 & \cdots & 8 \\ 0 & 1 & 2 & \cdots & 7 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 2 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{8} \\ \frac{1}{2} & 1 & \frac{1}{2} & \cdots & \frac{1}{7} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \frac{1}{7} & \frac{1}{6} & \ddots & 1 & \frac{1}{2} \\ \frac{1}{8} & \frac{1}{7} & \cdots & \frac{1}{2} & 1 \end{bmatrix}$$

   The second case looks best in format `rat`.

8. Suppose `A` is any matrix. What does the following statement do?
`A( 1:size(A,1)+1:end )`

9. If $\hat{x}$ is the computed solution of $Ax = b$ then $r := A\hat{x} - b$ is called the **residual**. Of course $r = 0$ if and only if $x = \hat{x}$. But usually $r \neq 0$. Does a small $\|r\|_2$ imply $\|x - \hat{x}\|_2$ small? Try the following experiment.

   Consider the Hilbert matrix H, where $H(i,j) := 1/(i+j-1)$ (the MATLAB command `H = hilb(n)` generates H) and perform the following computations:

```
>> n=10;
>> H=hilb(n); x = randn(n,1);
>> b = H*x;
>> x1= H \ b;
>> r = H*x1-b;
>> disp( [norm(r) norm(x-x1)])
```

   What is your conclusion?

   Since H is positive definite, LU factorization of H exists. Compute LU factorization of the Hilbert matrix H for $n = 8, 10, 12$ and check the backward stability of the algorithm GENP (GE with no pivoting). Recall that if `[L, U]= GENP(A)` then `GENP` is backward stable if `A+E = LU` for some `E` such that `norm(E)/norm(A)` $= \mathcal{O}(\mathbf{u})$.

   A naive MATLAB code that implements GENP is given below.

```
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
```

10. Compute the LU decomposition of $A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}$. What is the matrix that you get upon forming the product LU with the matrices L and U obtained as outputs of `GENP`? How different it is from A? Compute `norm(A-L*U)` and check backward stability of `GENP`. Now solve the linear system $Ax = b$ by using the computed LU factors and your programs for upper and lower triangular systems, where $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Again solve $Ax = b$ using the MATLAB command `x = A\b` (which uses Gaussian elimination with partial pivoting (GEPP)). What can you conclude about GENP and GEPP from the above experiment? Can you identify the step at which things start to go wrong?

*** End ***