Lab Session 1

MA-423 : Matrix Computations July-November 2014 S. Bora

---

1. This is an exercise on handling matrices in MATLAB.

   (a). Generate the following square matrix which is known as the Wilkinson matrix without using any *for loops*.

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

   Here n is the size of the matrix. You may write a function program `W = Wilkinson(n)` which takes the size n of the matrix as input for this.
   Hint: Use the MATLAB commands `eyes`, `ones` and `tril`.

   (b). A real square matrix $H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & -H_{11}^T \end{bmatrix}$ is said to be Hamiltonian if $H_{12}^T = H_{12}$ and $H_{21}^T = H_{21}$. Here T denotes the transpose of a matrix. Use concatenation and the `randn` command to generate a random real Hamiltonian matrix.

2. The following exercise illustrates that addition is not necessarily associative in finite precision environments.

   Given $n \in \mathbb{N}$, the built in function `chop` may be used to round $1/n$ to k significant digits and to simulate the summing of a finite number of terms of the sequence $\{\frac{1}{n}\}$, say the first m, in k-digit arithmetic. (Type `help chop` for details)

   Use the `chop` function to write a MATLAB function program $[\mathtt{s}, \mathtt{scf}, \mathtt{scb}] = \mathtt{sumreciprocal}(\mathtt{m}, \mathtt{k})$ to return the sum of the first m terms of the sequence $\frac{1}{n}, n \in \mathbb{N}$, as `s`, the sum of the first m terms in 'k' digit arithmetic as `scf` and finally the same sum in reverse order, that is, from $\frac{1}{m}$ to 1 in 'k' digit arithmetic as `scb`.

   Now calculate the following:

   (a) Sum up $1/n$ for $n = 1, 2, \ldots, 10^3$.
   (b) Round each number $1/n$ to 5 digits and simulate the summing of the resulting sequence for $n = 1, 2, \ldots, 10^3$ in 5-digit arithmetic.
   (c) Sum up the same chopped (or rounded) numbers in (b) again in 5-digit arithmetic but in reverse order, that is, for $n = 10^3, \ldots, 2, 1$.

   Compare the three computed results. Which among (b) and (c) is closer to (a)?

3. The solution of a system of equations $Ax = b$ can be obtained in MATLAB by setting $\mathtt{x} = \mathtt{A}\backslash\mathtt{b}$. MATLAB uses GEPP (Gaussian Elimination with Partial Pivoting) to find x for this command. [Wait for one more class to know the details!]

   The same may also be found by setting $\mathtt{x} = \mathtt{inv(A)} * \mathtt{b}$. Write an M-file which finds the time taken by both these commands for 20 matrices with sizes increasing from 200 to

1150 in steps of 50 and plots them on a semilog scale on the same graph. Use legends to distinguish between your curves.

4. Write function programs to solve the following systems of equations.

   (a). An upper triangular system $Ux = b$ by column oriented back substitution.

   (b). A lower triangular system $Lx = b$ by column oriented forward substitution.

5. Write a MATLAB function program $[\texttt{L},\texttt{U}] = \texttt{genp}(\texttt{A})$ which finds an LU factorization $A = LU$ of an n-by-n matrix A by performing Gaussian Elimination with no pivoting (GENP) [This is another name for Gaussian Elimination (GE) as you have been just taught in class!]. Use this and the programs written in response to parts (a) and (b) of Question 3 to do the following.

   (a) Find 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 is it from A?

   (b) Solve the system of equations $Ax = b$ where $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. How different is your answer from the correct solution $x \approx [-1, 1]^T$, (which is easily verified by hand calculation)?

   What can you conclude about GENP from the above algorithm? Can you identify the step at which things start to go wrong?