Lab Session 6

MA-423 : Matrix Computations Lab 2018 R. Alam

Stability and accuracy: Consider a linear system $Ax = b$. Then $\text{cond}(A) := \|A\|_2\|A^{-1}\|_2$ is called the condition number of A which is a measure of sensitivity of the linear system to small changes in the linear system. Let $\hat{x}$ be a computed solution of the linear system. Then $\eta(\hat{x}, A) := \frac{\|A\hat{x}-b\|_2}{\|A\|_2\|x\|_2}$ is known as the backward error of $\hat{x}$. It can be shown that

$$\frac{\|x - \hat{x}\|_2}{\|x\|_2} \lesssim \text{cond}(A)\, \eta(\hat{x}, A).$$

If y and $\hat{y}$ are two vectors such that $\|y - \hat{y}\|_2/\|y\|_2 \leq 0.5 \times 10^{-p}$ then $y(i)$ and $\hat{y}(i)$ agree to p significant digits for all indices i which satisfy $|\hat{y}(i)| \approx \|\hat{y}\|_2$. Moreover $|y(j) - \hat{y}(j)|/|y(j)| < 0.5 \times 10^{-p}$ for all $j \neq i$, so that the entries of $\hat{y}$ in these positions agree with corresponding entries of y to more than p significant digits. In summary, if $\|y - \hat{y}\|_2/\|y\|_2 \leq 0.5 \times 10^{-p}$ then y and $\hat{y}$ agree to p significant digits in their entries.

The purpose of the following experiments is to understand ill-conditioning of linear system and stability of an algorithm and their influence on the accuracy of computed solution.

1. 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 experiments.

   (a) Convince yourself that the condition number of H grows quickly with n. Try

   ```
   >> C=[];
   >> N= 2:2:16;
   for n=N
   H=hilb(n); C=[C; cond(H)];
   end
   >> semilogy(N,C)
   ```

   Can you guess an approximate relationship between $\text{cond}(H)$ and n based on this graph? (The MATLAB `cond(H)` computes the 2-norm condition number of H. Type `help cond` for details.)

   (b) The rule-of-thumb of ill-conditioning says that if $\text{cond}(H) = 10^t$ then one should expect to lose t digits in the solution of $Hx = b$. Examine this criterion by solving $Hx = b$. Here is how you can pick up the exact solution. Choose an arbitrary x and set $b := Hx$. Then x is the exact solution of $Hx = b$. The matrix H is SPD (symmetric positive definite). The matlab backslash `A \ b` command uses Cholesky factorization to solve an SPD system. There is also a matlab command `invhilb` which computes H^{-1} in a special way. You can also use GEPP (Gaussian Elimination with Partial Pivoting) to solve $Hx = b$. You may have to use `format long e` to see more digits. Try

   ```
   >> n=8;
   >> H=hilb(n); HI = invhilb(n);
   >> x= rand(n,1);
   >> b =H*x;
   >> x1 = H\ b; % Call this is method1
   >> x2 = HI*b; % Call this is method2
   ```

Compute backward error `eta`, condition number `cond` and the error `err` for method1 and method2 and display the result in the format `[ eta cond err]`.

Repeat for $n = 10$ and $n = 12$.

* List the results corresponding to $n = 8, 10, 12$, and determine correct digits in `x1, x2.`
* How many digits are lost in computing `x1` and `x2`? How does this correlate with the size of the condition number?
* Which is better among `x1` and `x2` or isn't there much of a difference? Is it fair to say that the inaccuracy resulted from a poor algorithm?

(c) 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:

```
>> 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?

(d) 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). 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
```

2. Wilkinson's matrix is defined as follows: 1 on the diagonal, -1 everywhere below the main diagonal, 1 in the last column, and 0 everywhere else. Write a MATLAB function `W = Wilkinson(n)` that generates Wilkinson's matrix W of size n using MATLAB functions `eye, tril` and `ones`.

(a) For $n = 32$, pick a random x and then compute $b := W * x$. Solve $Ax = b$ using MATLAB backslash command and compute the error $\|x - \hat{x}\|_\infty / \|x\|_\infty$ (type `help`

`norm` for more info about computing norm). Does the size of the error confirm that GEPP is unstable for this system? Also compute cond(A). Can the poor answer be attributed to ill-conditioning of the matrix W? Repeat the test for $n = 64$.

(b) Repeat the experiment in part (a) using QR decomposition. It is easy in matlab. The command `[Q,R] =qr(A)` gives unitary Q and upper triangular R such that $A = QR$. Solve $Wx = b$ using QR decomposition and compare the results with those in part(a). Which of the two methods appear to give a better answer?

3. Pivot growth of Gaussian elimination with pivoting (GEPP or GECP) is given by $PG(A) = \max_{ij} |U(i,j)|/\max_{ij} |A(i,j)|$ which influences accuracy of computed solution. Modify your MATLAB function `[L, U, p] = GEPP(A)` for computing LU decomposition of a nonsingular matrix A to write a new function $[\mathtt{L}, \mathtt{U}, \mathtt{p}, \mathtt{rho}] = \mathtt{gepp}(\mathtt{A})$ so that it returns an additional output `rho` which calculates the pivot growth factor, that is, $\mathtt{rho} = PG(A)$.

[**Hint:** Given a matrix X, `max(X)` is a row vector containing the maximum element from each column.]

It is well known that the pivot growth factor for GEPP satisfies $PG(A) \leq 2^{n-1}$ which is attained by the Wilkinson matrix. Verify this graphically by doing the following:

First plot the graph of 2^{n-1} in $\log 10$ scale for $n = 10 : .5 : 505$ by setting `X = 2.^(n-1)` and then typing `semilogy(n,X,'r')`. Hold this plot by typing `hold on` and type the following sequence of commands (which assumes that the Wilkinson matrix of size n is generated by the function `W = Wilkinson(n)`).

```
>> n = 10:20:500;
>> m = length(n); G = zeros(m,1);
>> for i = 1:m
>> W = Wilkinson(n(i)); [L,U,p,rho] = geppg(W); G(i) = rho;
>> end
>> semilogy(n,G,'b*')
```

The second plot should come in the form of blue dots that fall on the red curve produced by the earlier plot.

However, statistics suggest that for most practical examples, $PG(A) \leq n^{2/3}$ for GEPP. Verify this graphically by generating random matrices instead of Wilkinson matrices in the sequence of commands given above.

4. There is no strong correlation between pivot growth and the ill-conditioning of a matrix. This is illustrated by a Golub matrix. A Golub matrix A of size n is an ill-conditioned integer matrix whose LU factorization without pivoting fails to reveal that A is ill-conditioned. The matrix A is given by $A := LU$, where L unit lower triangular with random integer entries and U is unit upper triangular with random integer entries. The function `golub` given below generates a Golub matrix of size n :

```
function A = golub(n)
s = 10;
L = tril(round(s*randn(n)),-1)+eye(n);
U = triu(round(s*randn(n)),1)+eye(n);
A = L*U;
```

Compute LU factorization of A using your function `[L, U] = GENP(A)`. Also, compute the pivot growth $PG(A)$ and the condition number $\text{cond}(A) = \|A\|_2\|A^{-1}\|_2$ using MATLAB command `cond(A)`. If $\text{cond}(A)$ is large then the system $Ax = b$ is ill-conditioned and in such a case A is called ill-conditioned. Does $PG(A)$ reflect the ill-conditioning of A?

**********End**********