Lab Session 3

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

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

Stability: Let ALG be an algorithm that solves $Ax = b$, that is, $\hat{x} = \text{ALG}(A, b)$ is the computed solution. Then ALG is said to be **backward stable** if $(A+\Delta A)\hat{x} = b+\Delta b$ for some ΔA and Δb such that $\|\Delta A\|/\|A\| = \mathcal{O}(\text{u})$ and $\|\Delta b\|/\|b\| = \mathcal{O}(\text{u})$. Similarly, if $[L, U] = \text{ALG}(A)$ is the computed LU decomposition then ALG is backward stable provided that $A+\Delta A = LU$ for some ΔA such that $\|\Delta A\|/\|A\| = \mathcal{O}(\text{u})$. In this case, $\|\Delta A\|/\|A\| = \|A - LU\|/\|A\|$ is called backward error.

1. Use your MATLAB function `[L,U] = GENP(A)` that computes LU factorization $A = LU$ of an n-by-n matrix A by performing Gaussian Elimination with no pivoting (GENP).

   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 is it from A? 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 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?

Stability and accuracy: Suppose that $\hat{x}$ is a computed solution of $Ax = b$. Then it can be shown that $(A + E)\hat{x} = b$ for some E and the backward error of $\hat{x}$ is given by

$$\eta(\hat{x}, A) := \|E\|_2 = \frac{\|A\hat{x} - b\|_2}{\|A\|_2\|x\|_2}.$$

Further, the sensitivity of the system is measured by $\text{cond}(A) := \|A\|_2\|A^{-1}\|_2$ which is called the condition number of A. It can be shown that

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

Now, suppose that $\|\cdot\|$ is either the 1-norm, ∞-norm or the 2-norm and that x and $\hat{x}$ are two vectors such that $\|x - \hat{x}\|/\|x\| \leq 0.5 \times 10^{-p}$. Then this means that $x(i)$ and $\hat{x}(i)$ agree to p significant digits for all indices i which satisfy $|\hat{x}(i)| \approx \|\hat{x}\|$. Moreover for all $j \neq i$, $|x(j) - \hat{x}(j)|/|x(j)| < 0.5 \times 10^{-p}$ so that the entries of $\hat{x}$ in these positions agree with corresponding entries of x to more than p significant digits. In summary if $\|x - \hat{x}\|/\|x\| \leq 0.5 \times 10^{-p}$, then x and $\hat{x}$ agree to p significant digits in their entries.

The purpose of the following experiment is to understand ill-conditioning and stability and their influence on the accuracy of computed solution.

2. The rule-of-thumb of ill-conditioning is that if $\mathrm{cond}(H) = 10^t$ then one should expect to lose t digits in the solution of $Hx = b$. Examine this by solving $Hx = b$, where H is the infamous Hilbert matrix given by $H(i, j) = 1/(i + j - 1)$. Use MATLAB command H = hilb(n) to generate $n \times n$ Hilbert matrix H.

   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?

3. Use your function `GENP` to compute LU factorization of the Hilbert matrix H for $n = 8, 10, 12$ and check the backward stability of the algorithm GENP.

4. We now look at the growth of the condition number of the Hilbert matrix. 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 $\mathrm{cond}(H)$ and n based on this graph? The MATLAB `cond(H)` computes the 2-norm condition number of H. Theoretically $\mathrm{cond}(H) \approx \left(\dfrac{(1+\sqrt{2})^{4n}}{\sqrt{n}}\right)$. Plot (in a single plot) the theoretical value of cond(H) and cond(H) computed by MATLAB. The condition number computed by MATLAB reaches the maximum when $n = 13$. The computed condition number does not continue to grow when $n > 13$. This can be explained as follows: It is known that $\sigma_{\max}(H) := \|H\|_2 \to \pi$ and $\sigma_{\min}(H) := 1/\|H^{-1}\|_2 \to 0$ as $n \to \infty$. Hence

$$\mathrm{cond}(H) = \frac{\sigma_{\max}(H)}{\sigma_{\min}(H)} \approx \frac{\pi}{\sigma_{\min}(H) + \mathsf{eps}} \approx \frac{\pi}{\mathsf{eps}}.$$

```
%%%%%%     Matlab program that implements growth of cond(H)
% generate Hilbert matrices and compute cond number with 2-norm

N=50; % maximum size of a matrix
condofH = []; % conditional number of Hilbert Matrix
N_it= zeros(1,N);


% compute the cond number of Hn
for n = 1:N
    Hn = hilb(n);
    N_it(n)=n;
    condofH = [condofH cond(Hn,2)];
end

% at this point we have a vector condofH that contains the condition
% number of the Hilber matrices from 1x1 to 50x50.
% plot on the same graph the theoretical growth line.


% Theoretical growth of condofH
x = 1:50;
y = (1+sqrt(2)).^(4*x)./(sqrt(x));

% plot
plot(N_it, log(y));
plot(N_it, log(condofH),'x', N_it,log(y));


% plot labels
plot(N_it, log(condofH),'x', N_it,log(y))
title('Conditional Number growth of Hilbert Matrix: Theoretical vs Matlab')
xlabel('N', 'fontsize', 16)
ylabel('log(cond(Hn))','fontsize', 16)
lgd = legend ('Location', 'northwest')
legend('MatLab', 'Theoretical')
legend('show')

%%%%%%    end of the program
```

(b) 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 $\|r\|_\infty$ imply $\|x - \hat{x}\|_\infty$ small? The answer is NO, in general. Try the following:

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

What is your conclusion? Can you explain your result?

****