Lab Session 2

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

---

1. Write a function program `[L,U,p] = gepp(A)` to find a unit lower triangular matrix `L`, an upper triangular matrix `U` and a column vector `p` satisfying `A(p,:) = LU` via Gaussian Elimination with Partial Pivoting (GEPP) as explained in the theory class.

2. Write a function program `x = geppsolve(A,b)` to solve a system $Ax = b$ via GEPP. Your program should call the program `[L,U,p] = gepp(A)` for the main step. Compare your answers with that of the MATLAB command $A \setminus b$.

3. Given $A \in \mathbb{R}^{n\times n}$, if P be a permutation matrix such that $PA = LU$ is an LU factorization of PA, then the determinant of A satisfies $\det(A) = \sigma\Pi_{i=1}^{n} u_{ii}$ where $u_{ii}, i = 1, 2, \ldots, n$ are the diagonal entries of U and $\sigma = 1$ or $\sigma = -1$ depending upon whether P is a product of an even or odd number of transpositions respectively.

   Use this fact to write a function program `d = mydet(A)` to compute the determinant of A. Verify that your program costs $\frac{2}{3}n^3$ flops for $A \in \mathbb{R}^{n\times n}$ as against at least $n!$ flops via the conventional formula for the $\det(A)$.

   [Hint: You may slightly modify `[L,U,p] = gepp(A)` to obtain `[L,U,p,sig] = gepp(A)` where `sig` plays the role of σ and then call this program within `d = mydet(A)`.]

4. If $A \in \mathbb{R}^{n\times n}$ is invertible, then A^{-1} satisfies the equation $AX = I_n$ where I_n is the $n \times n$ identity matrix. Finding X is equivalent to solving n linear systems

$$AX(:, i) = I_n(:, i),\ i = 1, 2, \ldots, n.$$

   Using the fact that each of these systems involve the same coefficient matrix A, write a function program `B = myinv(A)` to compute the inverse of A that costs $\mathcal{O}(n^3)$ flops.

   [Hint: Call `[L,U,p] = gepp(A)` once and `x = geppsolve(A,b)` n times within a *for loop*.]

5. The purpose of this experiment is to observe ill-conditioning. For illustration, we consider the infamous Hilbert matrix H given by $H(i, j) := 1/(i + j - 1)$.

   Origin: Suppose a function f is approximated by a polynomial p of degree $n - 1$, that is, $p(x) = a_0 + a_1 t + \cdots + a_{n-1}t^{n-1}$ on $[0, 1]$. The method of least squares (more on LSP, later in the course) determines a_j such that the error

$$E := \|f - p\|_2^2 = \int_0^1 (f(t) - p(t))^2 dt$$

   is minimized. Differentiating E w.r.t a_k and setting the partial derivative to 0 (necessary condition for minima), we have

$$\sum_{j=1}^{n-1} a_j \int_0^1 t^{j+k} dt = \int_0^1 t^k f(t) dt,\ k = 0 : n - 1$$

   which can be written in the matrix form $Ha = b$.

   There is a built-in function in MATLAB for generating Hilbert matrix. Type `help hilb` and `help invhilb` for details.

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)
```

Based on this graph formulate a conjecture as to the relationship between cond(H) and n. [Note: The MATLAB `cond(H)` computes the 2-norm condition number of H. Type `help cond` for details.]

Modify the above code and compute the condition number in 1-norm and ∞-norm.

6. The *Rule-of-thumb* of ill-conditioning says that if $\text{cond}(A) = 10^t$, and the entries of A and b are correct to s decimal places, then one should expect an agreement of at least $s - t$ significant digits in the corresponding entries of the exact solution and computed solution of $Ax = b$ obtained from GEPP or from Cholesky decomposition (which is applied if A is positive definite).

   Given a Hilbert matrix H, the aim of the next exercise is to verify the above rule of thumb for solutions of the system $Hx = b$ obtained via a number of different solution methods. Since computations are done in double precision by default in Matlab, (this means using 64 bits to represent any number in binary form which roughly translates to 16 digits after the decimal) we will have $s = 16$. In practice, the exact solution is unavailable. But for the experiments we can use the following trick to gain access to it:

   Choose x first and set $b := Hx$ and then solve $Hx = b$ by setting $x1 = H\backslash b$. Then x is the exact solution of $Hx = b$ and $x1$ is the computed solution!

   The following points will also help in your verification.

   * The number of significant digits in a number is the first nonzero digits and the number of all subsequent digits. For example 0.0123 has 3 significant digits while 1.0123 has 5 such digits.
   * If the norm is the 1, ∞ or 2 norm and 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 at least p significant digits in their entries.

   The system $Hx = b$ may be solved by using the `invhilb` command that generates the inverse of a Hilbert matrix. Moreover you can also use your `gepp` function formulated in the first problem to solve $Hx = b$. You may have to use `format long e` to see more digits. Note that $H \setminus b$ will NOT find a solution via GEPP in this case as H is positive definite. Instead it will use Cholesky method to find the solution. Now execute the following steps:

```
>> n=8;
>> H=hilb(n); HI = invhilb(n);
>> x= rand(n,1);
>> b =H*x;
>> x1 = H\ b; x2 = HI*b;
```

```
% obtain x3 by solving Hx=b using GEPP.
>> [x x1 x2 x3]
>> [cond(H) norm(x-x1)/norm(x) norm(x-x2)/norm(x) norm(x-x3)/norm(x)]
```

Repeat for $n = 10$ and $n = 12$ and list the results corresponding to $n = 8, 10, 12$, and determine correct digits in `x1, x2, x3.` Now answer the following questions:

How many digits are lost in computing `x1, x2` and `x3`? How does this correlate with the size of the condition number?

Which is better among `x1, x2` and `x3` or isn't there much of a difference?

7. 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 $\|r\|$ imply $\|x - \hat{x}\|$ small? Try the following:

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

What is your conclusion?

8. The next exercise shows why small pivots should be avoided and also demonstrates the better numerical properties of Gaussian Elimination with partial pivoting over no pivoting in dealing with matrices with small entries in pivotal positions. Generate random matrices of different sizes (for example, $n = 20, 40, 80, 100$ etc.) with small entries in $(1, 1)$ positions. To do this just type
`>> A = rand(n); A(1,1) = 50*eps*A(1,1);`
Find L and U by using `genp` code in the previous assignment and compute their norms as well as the norm of $LU - A$. (Type `help norm` for this.)

   Repeat the process with the L and U generated from the `lu` command of MATLAB. The only difference is that this time you will have to check the norm of $LU - PA$ instead of $LU - A$.

   Do this for each of the different sizes. Which of the two sets of LU decompositions produce the larger norms?

9. Recall the Wilkinson's matrix that you had generated in one of your previous classes.

   For $n = 32$, pick a random x and then compute $b := W * x$. Store the solution obtained by using GEPP in $\hat{x}$ and compute the error $\|x - \hat{x}\|_\infty / \|x\|_\infty$. Also compute cond$(A)$. As per the *Rule-of-thumb* (as given in [6]) can the poor answer be attributed to ill-conditioning of the matrix W? Repeat the test for $n = 64$.

10. Repeat the above experiment using QR decomposition (More about this later but finding it is easy in MATLAB. Just typing `[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. Which of the two methods appear to be give a better answer?