Lab Session 3

MA-423 : Matrix Computations September-November 2020 S. Bora

Instructions: The reports for all experiments are to be written clearly in a single pdf file. This can be a .doc/.odt file converted to pdf format. Additionally, the workspaces of experiments where random matrices and vectors are generated are to be saved separately and the filenames of the same are to be clearly mentioned in the report so that these can be accessed by loading the workspace when checking the work.

The following needs to be carefully noted before performing the experiments.

Note: 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 $Ax = b$ if it has been obtained from a backward stable algorithm.

This is because in such situations the (backward) errors in the computed solutions of $Ax - b$ that are pushed back into the data and indicated by $\frac{\|r\|_2}{\|b\|_2}$ and $\frac{\|r\|_2}{\|x_c\|_2\|A\|_2}$ are of the order of unit roundoff $u \approx 10^{-16}$ in such cases that is, the algorithm finds exact answers of perturbed problems where the perturbations in relative sense are of the same order as those induced by rounding error which is already in the data that is put to the algorithm. Due to this, if error induced by prior calculations/measurement errors is significantly more than a modest multiple of u, then this dominates the error analysis rather than those present due to rounding or backward errors.

1. The condition number of an $n \times n$ invertible matrix A is given by $\kappa(A) := \|A\|\|A^{-1}\|$ where $\|\cdot\|$ is a norm on the space of $n \times n$ real or complex matrices. A matrix is said to be ill conditioned if it has a high condition number. You will learn in your theory class that ill condition matrices are very close to being singular. 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.

2. 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.

   (a) 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.

   (b) 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`? Does the loss of accuracy in each case agree with the value predicted by the *Rule-of-thumb* mentioned earlier? Note that since the experiments are performed on randomly generated data and their is no error other than rounding error, $s \approx 16$ in the *Rule-of-thumb* analysis.

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

3. The purpose of this experiment is to illustrate that a small value of the norm of the residual is not enough to guarantee an accurate answer.

   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 $\|r\|/\|b\|$ imply $\|x - \hat{x}\|/\|x\|$ 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(b) norm(x-xt)/norm(x)])
   ```

   What is your conclusion?

4. 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 (forward) error $\|x - \hat{x}\|_\infty/\|x\|_\infty$. Also compute cond$(A)$ in the infinity norm and $\|r\|_\infty/\|b\|_\infty$. Repeat the test for $n = 64$.

   Further 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 (using `x = colbackward(Q'*b)`).

   Tabulate all the quantities for the different experiments. Noting that $s \approx 16$ in the *rule-of-thumb* analysis, answer the following.

   (a) Which of the two methods appear to give a lower forward error?

   (b) For which of the two methods is the *rule-of-thumb* predicting the correct answer reasonably well?

   (c) Which of the two methods give rise to a lower value of $\|r\|_\infty/\|b\|_\infty$?

   (c) What can you say about the backward stability of GEPP and QR decomposition methods from the experiments?

5. 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?