Math 504, Fall 2018 - Homework 3

December 4, 2018

1. Let us consider the matrices

$$A_1 = \begin{bmatrix} -1 & 3 & 0 \\ -4 & -1 & 3 \\ 0 & -4 & -1 \end{bmatrix} \text{ and } A_2 = \begin{bmatrix} 1 & 1 & 2 \\ 4 & 4 & 0 \\ 2 & 0 & 1 \end{bmatrix}. \tag{0.1}$$

You are expected to perform all of the computations by hand.

(a) Compute the factorizations of the form $PA = LU$ for each one of A_1 and A_2 by applying the LU factorization algorithm with partial pivoting.

(b) Use the factorization computed in part (a) to solve the linear system $A_2 x = \begin{bmatrix} 3 & -4 & -2 \end{bmatrix}^T$.

2. This question concerns the matrices A_1 and A_2 as in (0.1). In both parts, x denotes an entry that is possibly not zero.

(a) Find a permutation matrix $P \in \mathbb{R}^{3\times 3}$ and a lower triangular matrix $L \in \mathbb{R}^{3\times 3}$ such that

$$L \cdot P \cdot A_2 = \begin{bmatrix} 4 & 4 & 0 \\ 0 & x & x \\ 0 & x & x \end{bmatrix}.$$

(b) Find a lower triangular matrix $L \in \mathbb{R}^{3\times 3}$ and an upper triangular matrix $U \in \mathbb{R}^{3\times 3}$ such that

$$A_1 = U \cdot \begin{bmatrix} -1 & 3 & 0 \\ -4 & x & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot L.$$

3. For a dense matrix $A \in \mathbb{C}^{n\times n}$ ("dense" means many of the entries, for instance more than half of the entries, of A are nonzero) and a vector $b \in \mathbb{C}^n$, the standard approach to solve the linear system $Ax = b$ for $x \in \mathbb{C}^n$ is as follows:

1. Compute an LU factorization $PA = LU$ by partial pivoting;
2. Permute the right-hand side, that is form $\hat{b} = Pb$;
3. Letting $y := Ux$, solve the lower triangular system $Ly = \hat{b}$ for y by forward substitution;
4. Solve the upper triangular system $Ux = y$ for x by back substitution.

Write a Matlab function to solve the system $Ax = b$ following the steps 1-4 described above. A Matlab routine (see `myLU_pivot.m` under the matlab link on the course webpage) for the computation of an LU factorization employing the partial pivoting strategy is provided on the course webpage. You can use this routine in your implementation. You should implement the forward substitution and back substitution on your own.

4. One important application where very large linear systems arises is the numerical solution of partial differential equations. Consider the Poisson equation

$$u_{xx}(x,y) + u_{yy}(x,y) = -(\cos(x+y) + \cos(x-y)), \quad 0 < x < \pi,\ 0 < y < \frac{\pi}{2}$$

with the boundary conditions

$$u(x,0) = \cos\ x,\ \ u\left(x, \frac{\pi}{2}\right) = 0, \quad 0 \le x \le \pi;$$
$$u(0,y) = \cos\ y,\ \ u(\pi,y) = -\cos\ y, \quad 0 \le y \le \frac{\pi}{2}.$$

The solution sought $u(x, y)$ is a twice differentiable function. Furthermore, u_{xx} and u_{yy} denote the partial derivatives of u with respect to x and y twice, respectively.

One can numerically estimate the solution using finite differences. In particular, it turns out that for a given $h \approx 0$, we have

$$u_{xx}(x, y) = \frac{u(x + h, y) - 2u(x, y) + u(x - h, y)}{h^2} + O(h^2),$$
$$u_{yy}(x, y) = \frac{u(x, y + h) - 2u(x, y) + u(x, y - h)}{h^2} + O(h^2).$$

Now let use denote by u_{ij} for $i = 0, \ldots, 2n$ and $j = 0, \ldots, n$ approximate solutions satisfying $u_{ij} \approx u(ih, jh)$ where $h = \frac{\pi/2}{n}$. Notice that $u_{0j}, u_{(2n)j}, u_{i0}, u_{in}$ are exact and can be obtained from the boundary conditions. Employing the finite difference formulas and $u_{ij} \approx u(ih, jh)$, the Poisson equation

$$u_{xx}(ih, jh) + u_{yy}(ih, jh) = -(\cos(ih + jh) + \cos(ih - jh)),$$

at discrete points can be approximated by the linear equations

$$\frac{u_{(i+1)j} - 2u_{ij} + u_{(i-1)j}}{h^2} + \frac{u_{i(j+1)} - 2u_{ij} + u_{i(j-1)}}{h^2} = -(\cos(x_i + y_j) + \cos(x_i - y_j))$$

where $x_i := ih$ and $y_j := jh$.

Choose $h = \frac{\pi/2}{70}$ and set up a linear system with the unknowns $\{u_{ij}\}$ for $i = 1, \ldots, 139$ and $j = 1, \ldots, 69$. Estimate the solution of the Poisson equation by solving the resulting linear system by your linear system solver from question 3.

5. The kth principal submatrix of $A \in \mathbb{C}^{n \times n}$ is the upper left-most $k \times k$ portion of A. Show that A has an LU factorization of the form $A = LU$ where L is unit lower triangular with ones along the diagonal, and U is invertible and upper triangular if and only if all principal submatrices of A are invertible.

6. Show both of the computations below can be performed in a backward stable manner. Assume all of the computations are performed in IEEE floating point arithmetic. For simplicity also assume, in both parts, x and A are representable in IEEE floating point arithmetic.

(a) The summation $f(x) = x_1 + x_2 + x_3$ as a function of $x \in \mathbb{R}^3$.

(b) The product $f(A) = Ax$ as a function of $A \in \mathbb{R}^{n\times n}$ only, and for a fixed $x \in \mathbb{R}^n$.

7. Let $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$ and $y = Ax$. Assume x is representable in IEEE floating point arithmetic. Furthermore, let $\hat{y}$ represent the matrix-vector product Ax computed in IEEE floating point arithmetic. Perform a backward error analysis to deduce an upper bound on the relative error

$$\frac{\|\hat{y} - y\|_2}{\|y\|_2}.$$

8. The purpose of this question is to gain insight into the cause of the numerical error of the solution of a linear system by LU factorization. In theory, when a linear system $Ax = b$ is solved by means of the LU factorization with partial pivoting strategy, the overall procedure is backward stable. The computed solution $\widehat{x}$ satisfies $(A+\delta A)\widehat{x} = b$ for some δA such that

$$\|\delta A\|_1 \leq 3n^3 \rho_{\max} \epsilon_{\text{mach}} \|A\|_1 + o(\epsilon_{\text{mach}}) \tag{0.2}$$

where $\rho_{\max}$ is the growth factor defined by

$$\rho_{\max} \;:=\; \frac{\max_{j,k=1,\ldots,n} |u_{jk}|}{\max_{j,k=1,\ldots,n} |a_{jk}|}.$$

It can be shown that $\rho_{\max} \leq 2^{n-1}$ (but in practice typically $\rho_{\max} \leq \sqrt{n}$). A backward error analysis employing (0.2) yields

$$\frac{\|\widehat{x} - x\|_1}{\|x\|_1} \leq 3n^3 \rho_{\max} \epsilon_{\text{mach}} \|A\|_1 \|A^{-1}\|_1 + o(\epsilon_{\text{mach}}). \tag{0.3}$$

Recall that the quantity $\kappa_1(A) = \|A\|_1\|A^{-1}\|_1$ is called the condition number of the matrix A.

On the course website, the following Matlab m-file is made available:

- `plot_condition_number.m`: to plot the condition number $\kappa_1(A) = \|A\|_1\|A^{-1}\|_1$ with respect to the size of the matrix n for a given class of matrices.

The routine takes three parameters. The first two parameters lb, ub determine the interval $[lb, ub]$ in which the size of the matrix n varies. The third is a string (such as 'lotkin', 'kahan', 'leslie', 'randn') identifying the family of matrices on which you would be experimenting with the rounding errors. There is a built-in Matlab routine `gallery`, which generates matrices from various families (type `help gallery` for more information regarding this routine). You can experiment with most of the families that can be generated by the Matlab command `gallery`. Here you are specifically expected to experiment with 'lotkin', 'ris', 'riemann' and 'chebvand'.

(a) Plot the condition numbers for 'lotkin', 'ris', 'riemann' and 'chebvand' with sizes varying in $[5, 10]$.

(b) Consider the 10×10 linear systems $Ax = b$ with A chosen as one of the 'lotkin', 'ris', 'riemann' and 'chebvand' matrices. For each of these matrices repeat the following: **(1)** set x equal to the vector of ones; **(2)** form $b = Ax$; **(3)** solve the system $A\widehat{x} = b$.

(i) Based on equation (0.3) and your observations in part **(a)**, how many of the decimal digits of x (the exact solution) and $\widehat{x}$ (the computed solution) would you expect to match?

(ii) Perform the sequence of computations **(1)-(3)** in Matlab. You can form a specific matrix by typing

```
>> A = gallery(matname,10);
```

where matname is `'lotkin'`, `'ris'`, `'riemann'`, or `'chebvand'`. Do you observe the accuracy that you expect in **(i)**?

The next question is not the part of the homework. Your solution to this will not be evaluated.

9. We have discussed in class how to compute an LU factorization for a square invertible matrix $A \in \mathbb{C}^{n\times n}$ using the *partial pivoting strategy*. An alternative to the partial pivoting strategy is the *full pivoting strategy*, for which a pseudocode is given below. What kind of factorization does this

---

Input: Invertible $A \in \mathbb{R}^{n\times n}$
Output: Upper triangular $U \in \mathbb{C}^{n\times n}$, a lower triangular $L \in \mathbb{C}^{n\times n}$ and $p, q \in \mathbb{R}^{n-1}$.

$L \leftarrow I_n$
for $j = 1, \ldots, n-1$ **do**
  Let $q, p \in \{j, \ldots, n\}$ be such that $|a_{q,p}| = \max\{|a_{\ell,m}| \mid \ell, m = j, \ldots, n\}$.
  $A(j, j:n) \longleftrightarrow A(q, j:n)$
  $A(1:n, j) \longleftrightarrow A(1:n, p)$
  $L(j, 1:j-1) \longleftrightarrow L(q, 1:j-1)$.
  for $k = j+1, \ldots, n$ **do**
    $L(k, j) \leftarrow A(k, j)/A(j, j)$
    $A(k, j:n) \leftarrow A(k, j:n) - L(k, j)A(j, j:n)$
  end for
end for
$U \leftarrow A$

LU factorization algorithm with full pivoting generate? In particular, how are A and L, U related?