

CSL361 Programming assignment 2:

October 9, 2009

1. Develop and test **C/C++** functions (based on *LU decomposition using partial pivoting, forward and backward substitution; column oriented **gaxpy** based, just to be different from what we did in the class*) to solve the system $\mathbf{AX} = \mathbf{B}$ where \mathbf{A} , \mathbf{X} and \mathbf{B} are matrices of dimension $n \times n$, $n \times k$ and $n \times k$, respectively. Use the above function to compute \mathbf{A}^{-1} . Also write a function to estimate the condition number of a matrix. In each case verify the results with standard **matlab** functions. Do not use any standard library.
2. Test the procedure developed above on the symmet-

ric diagonal matrix of order 10:

$$\mathbf{A} = \begin{bmatrix} -2 & 1 & & & & & & & & & \\ & 1 & -2 & 1 & & & & & & & \\ & & 1 & -2 & 1 & & & & & & \\ & & & & \cdots & \cdots & \cdots & & & & \\ & & & & & & & 1 & -2 & 1 & \\ & & & & & & & & 1 & -2 & \end{bmatrix}$$

The inverse of this matrix is known to be

$$(\mathbf{A}^{-1})_{ij} = (\mathbf{A}^{-1})_{ji} = \frac{-i(n+1-j)}{(n+1)} \quad (i \leq j)$$

3. Extend the *LU decomposition* procedure developed above to deal with rectangular matrices ($m \times n$). Compute the space of solutions as a set of basis vectors. Indicate if there is no solution.

[Note: Consider, first the lower triangular case when $m \geq n$, i.e.,

$$\begin{bmatrix} L_{11} \\ L_{21} \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

where $L_{11} \in \mathbb{R}^{n \times n}$, $L_{21} \in \mathbb{R}^{(m-n) \times n}$, $b_1 \in \mathbb{R}^n$ and $b_2 \in \mathbb{R}^{m-n}$. Assume that L_{11} is lower triangular and nonsingular. If we apply forward substitution to $L_{11}x = b_1$, then x solves the system provided

$L_{21}(L_{11}^{-1}b_1) = b_2$. Otherwise, there is no solution and we have to apply least squares optimization.

Now, consider the case when $n > m$. In this case we can apply forward substitution to the square system $L(1 : m, 1 : m)x = b$ and assign an arbitrary value for $x(m + 1 : n)$. You may output the basis.]

4. Test your procedure on randomly generated square and rectangular matrices (choose moderate size matrices $20 \leq n, m \leq 50$). Compute the **null space**, **range space**, **rank** and **nullity**. Verify with standard **matlab** functions. For square matrices compute the inverse (if full rank) and verify whether $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$. Explain any discrepancies that you may observe.
5. Investigate the difficulties in inverting the following matrix:

$$\mathbf{A} = \begin{bmatrix} -0.0001 & 5.096 & 5.101 & 1.853 \\ 0 & 3.737 & 3.740 & 3.392 \\ 0 & 0 & 0.006 & 5.254 \\ 0 & 0 & 0 & 4.567 \end{bmatrix}$$

Estimate the **condition number** of the above matrix.

6. Use the *LU decomposition* procedure developed above

to solve the following matrix equations. In each case explain the difficulties and errors. Estimate the condition number in each case and verify with **matlab**:

- (a) The **Hilbert matrix** of order n is defined by $a_{ij} = (i + j - 1)^{-1}$ for $1 \leq i, j \leq n$. Define $b_i = \sum_{j=1}^n a_{ij}$. Then the solution to the system of equations $\sum_{j=1}^n a_{ij}x_j = b_i$ for $1 \leq i \leq n$ is $\mathbf{x} = [1, 1, \dots, 1]^T$. Verify this. Select some values of n in the range $2 \leq n \leq 15$ and solve using the function developed above. Do the case $n = 2$ by hand to see what difficulties may arise.
- (b) For a moderate value of n (say $5 \leq n \leq 30$) solve the system where $a_{ij} = (i - 1)^{j-1}$ and $b_i = i - 1$. Guess the correct solution by looking at the output and verify.
- (c) For a fixed value of n from 2 to 4, let

$$a_{ij} = (i + j)^2$$

and

$$b_i = ni(i + n + 1) + \frac{1}{6}n(1 + n(2n + 3))$$

the vector $\mathbf{x} = [1, 1, \dots, 1]^T$ is the ideal solution (verify).

7. Fit the function

$$a \ln x + b \cos x + c \exp x$$

to the data

x	0.24	0.65	0.95	1.24	1.73	2.01	2.23	2.52	2.77	2.99
y	0.23	-0.26	-1.10	-0.45	0.27	0.10	-0.29	0.24	0.56	1.00

by least squares (use your own *LU decomposition routines*) and plot the function. Evaluate the residual error and the associated χ^2 probability. Corrupt the data with zero mean Gaussian noise with different values of σ (ranging from 0.01 to 0.1) and repeat. Also show the re-fits after rejecting the outliers.

8. Gaussian elimination with partial pivoting can be organized so that it is rich in matrix multiplications. Such a scheme may be useful when dealing with really large matrices. Following is a block outer product procedure (block dot product and block gaxpy based procedures are also possible).

Assume that $A \in \mathbb{R}^{n \times n}$ and for clarity that $n = rN$. Partition A as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{matrix} r \\ n - r \end{matrix}$$

$r \quad n - r$

- (a) Use scalar Gaussian elimination with partial pivoting (rectangular version) to compute permutations $P_1 \in \mathbb{R}^{n \times n}$, unit lower triangular $L_{11} \in \mathbb{R}^{r \times r}$ and upper triangular $U_{11} \in \mathbb{R}^{r \times r}$ so that

$$P_1 \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} = \begin{bmatrix} L_{11} \\ L_{21} \end{bmatrix} U_{11}$$

- (b) Apply P_1 across the rest of A

$$\begin{bmatrix} \tilde{A}_{12} \\ \tilde{A}_{22} \end{bmatrix} = P_1 \begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix}$$

- (c) Solve the lower triangular multiple right hand side problem

$$L_{11}U_{12} = \tilde{A}_{12}$$

- (d) Perform the level-3 update

$$\tilde{A} = \tilde{A}_{22} - L_{21}U_{12}$$

With these computations we obtain the factorization

$$P_1 A = \begin{bmatrix} L_{11} & 0 \\ L_{21} & I_{n-r} \end{bmatrix} \begin{bmatrix} I_r & 0 \\ 0 & \tilde{A} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & I_{n-r} \end{bmatrix}$$

The process is then repeated on the first r columns of \tilde{A} .

In general, during step k ($1 \leq k \leq N - 1$) of the block algorithm we apply scalar Gaussian elimination to a matrix of size $(n - (k - 1)r)$ -by- r . An r -by- $(n - kr)$ multiple right hand side system is solved and

a level-3 update of size $(n - kr)$ -by- $(n - kr)$ is performed. The level-3 fraction for the overall process is approximately given by $1 - 3/(2N)$. Thus, for large N , the procedure is rich in matrix multiplication.

Implement the block Gaussian elimination outlined above and test it on some of the (large size) examples of earlier problems.

9. Consider the multiple right hand side forward substitution problem (similarly backward substitution) of computing the solution $X \in \mathbb{R}^{n \times q}$ to $LX = B$ where $L \in \mathbb{R}^{n \times n}$ is lower triangular and $B \in \mathbb{R}^{n \times q}$. Here is one way to solve this using block operations rich in matrix multiplication. Partition $LX = B$ as (assuming that the diagonal blocks are square)

$$\begin{bmatrix} L_{11} & 0 & \dots & 0 \\ L_{21} & L_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & \dots & L_{NN} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix}$$

We can solve $L_{11}X_1 = B_{11}$ for X_1 and then remove

it from the block equations

$$\begin{bmatrix} L_{22} & 0 & \dots & 0 \\ L_{32} & L_{33} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{N2} & L_{N3} & \dots & L_{NN} \end{bmatrix} \begin{bmatrix} X_2 \\ X_3 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} B_2 - L_{21}X_1 \\ B_3 - L_{31}X_1 \\ \vdots \\ B_N - L_{N1}X_1 \end{bmatrix}$$

and recurse. Note that the procedure is block saxpy based (of course, you can also do it based on block dot product). Try it out along with the previous problem.