

# CAAM 353: Computational Numerical Analysis

## Homework 2, Jan. 17, 2008

**Due: Jan. 24, 2008**

*Note: All MATLAB functions mentioned in this homework assignment can be found on the CAAM353 homepage, or come with MATLAB. Turn in all MATLAB code that you have written and turn in all output generated by your MATLAB functions/scripts. MATLAB functions/scripts must be commented and output must be formatted nicely.*

**Problem 1 (20 points)** i. (20points) Write a MATLAB function `getLUP.m` that takes as inputs the  $n \times n$  real array  $A$  and the integer array  $ipivt$  and returns a unit lower triangular matrix  $L \in \mathbb{R}^{n \times n}$ , an upper triangular matrix  $U \in \mathbb{R}^{n \times n}$ , and a permutation matrix  $P \in \mathbb{R}^{n \times n}$ , such that

$$PA = LU.$$

For the three matrices specified in `testlu_pp.m`, compare the output of your function `getLUP.m` with that of the MATLAB function `lu.m`. (Keep in mind that the MATLAB function `lu.m` expects the original matrix  $A$  as input and that `lu_pp.m` overwrites the array  $A$ !)

ii. (extra credit problem) Write a MATLAB function `getLUPQ.m` that takes as inputs the  $n \times n$  real array  $A$  the integer arrays  $ipivtr$ ,  $ipivtc$  and returns a unit lower triangular matrix  $L \in \mathbb{R}^{n \times n}$ , an upper triangular matrix  $U \in \mathbb{R}^{n \times n}$ , and permutation matrices  $P, Q \in \mathbb{R}^{n \times n}$ , such that

$$PAQ = LU.$$

For the three examples specified in `testlu_cp.m`, use the output of your function `getLUPQ.m` to solve the linear system  $Ax = b$ . Report the matrices  $P, Q, L, U$ , the computed solution  $x$ , and the error between computed and exact solution.

**Problem 2 (20 points)** We want to solve

$$A^T y = d$$

using the LU-decomposition of  $A$ . The MATLAB program `lu_pp.m` computes Gauss-transformations  $M_k, k = 1, \dots, n-1$ , permutation matrices  $P_k, k = 1, \dots, n-1$ , and an upper triangular matrix  $U$  such that

$$U = M_{n-1}P_{n-1}M_{n-2}P_{n-2} \cdots M_2P_2M_1P_1A. \quad (1)$$

Information about these matrices is returned by `lu_pp.m` in the  $n \times n$  real array  $A$  and the integer array  $ipivt$ .

In class we have discussed how (1) can be used to solve  $A^T y = d$ . Write a MATLAB function `lu_pp_sl_trans` that takes as its input the output of the MATLAB function `lu_pp.m` and the right hand side vector, and returns the solution of  $A^T y = d$ . Your function `lu_pp_sl_trans` should take as much advantage of the special structure of  $M_k$  and  $P_k$ ,  $k = 1, \dots, n-1$ , as possible!

Test your MATLAB function using the data

$$A = \begin{pmatrix} -2 & 3 & 0 & 2 \\ 2 & 0 & 4 & 5 \\ 4 & 1 & 2 & 3 \\ 2 & -1 & 6 & 5 \end{pmatrix}, \quad d = \begin{pmatrix} 6 \\ 3 \\ 12 \\ 15 \end{pmatrix}.$$

Check your computed solution against  $y = A' \backslash d$ .

**Problem 3 (30 points)** This problem is motivated by *substructuring techniques* or *domain decomposition techniques* used for the parallel solution of large scale system of linear equations arising in the simulation of mechanical structures or in the solution of (partial) differential equations.

Let  $A_{kk} \in \mathbb{R}^{m_k \times m_k}$ ,  $A_{kn} \in \mathbb{R}^{m_k \times m_n}$ ,  $A_{nk} \in \mathbb{R}^{m_n \times m_k}$ ,  $k = 1, \dots, n$  be given matrices and let  $b_k \in \mathbb{R}^{m_k}$ ,  $k = 1, \dots, n$  be given vectors. Consider the linear system

$$\begin{pmatrix} A_{11} & 0 & 0 & \cdots & A_{1n} \\ 0 & A_{22} & 0 & \cdots & A_{2n} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ A_{n1} & A_{n2} & & \cdots & A_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{pmatrix} \quad (2)$$

in the unknowns vectors  $x_k \in \mathbb{R}^{m_k}$ ,  $k = 1, \dots, n$ .

- i. (10points) Assume that  $A_{kk} \in \mathbb{R}^{m_k \times m_k}$ ,  $k = 1, \dots, n-1$ , are invertible and use the first  $n-1$  equations in (2) to express  $x_k$  in terms of  $x_n$ . Insert these expressions into the last equation in (2) to obtain an  $m_n \times m_n$  system of linear equations in  $x_n$ .
- ii. (20 points) We now apply the approach in part i to a linear system that arises from the discretization of a boundary value problem in which the interval  $[0, 1]$  is divided into  $n-1$  subintervals of equal size and each of these  $n-1$  subinterval is subdivided into  $m_k = ((N-1) - (n-2))/(n-1)$  subintervals,  $k = 1, \dots, n-1$ . In this case we set  $N = 128$  and  $n = 3$ ,  $n = 5$ , or  $n = 9$ .

Let  $m_k = ((N - 1) - (n - 2))/(n - 1)$ ,  $k = 1, \dots, n - 1$ , and  $m_n = n - 2$ .

$$\begin{aligned}
 A_{kk} &= \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{m_k \times m_k}, \quad k = 1, \dots, n - 1, \\
 A_{nn} &= \begin{pmatrix} 2 & & & \\ & 2 & & \\ & & \ddots & \\ & & & 2 \end{pmatrix} \in \mathbb{R}^{m_n \times m_n}, \\
 A_{1n} = A_{n1}^T &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{m_1 \times m_n}, \\
 A_{n-1,n} = A_{n,n-1}^T &= \begin{pmatrix} 0 & 0 & 0 & -1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{m_{n-1} \times m_n}, \\
 A_{k,n} = A_{n,k}^T &= \begin{pmatrix} 0 & -1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \in \mathbb{R}^{m_k \times m_n}, \quad k = 2, \dots, n - 2,
 \end{aligned}$$

( $A_{kn}$ ,  $k = 2, \dots, n - 2$ , has one entry equal to  $-1$  in the first row of column  $k - 1$  and one entry equal to  $-1$  in the last row of column  $k$ ; all other entries are equal to 0) and

$$b_k = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} \in \mathbb{R}^{m_k}, k = 1, \dots, n - 1, \quad b_n = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} \in \mathbb{R}^{m_n},$$

Implement your solution strategy developed in part i to solve (2) with the matrices specified above. Choose  $N = 128$  and apply your implementation for  $n = 2, 4, 8$ .

Rather than printing the solution vector, use the following MATLAB code. In the following MATLAB code I assume that cell arrays are used to store the solution vector  $x$ . In particular,  $x\{k\}$  is the subvector  $x_k$  in (2).

```

xx = x{1};
for k = 2:n-1
    xx = [ xx; x{n}(k-1); x{k}];
end
plot(xx)
title(['Solution for N=128 and n = ', int2str(n)])

```

**Note:** In your implementation you may use MATLAB's `spdiags` and `sparse` to generate the matrices. For example the matrices  $A_{kk}$ ,  $k = 1, \dots, n-1$ , can be generated using

```

mk          = ((N-1)-(n-2))/(n-1);
e           = ones(mk,1);
A{k,k}      = spdiags([-e 2*e -e], -1:1, mk, mk);
b{k}        = e;

```

The matrices  $A_{kn}$ ,  $k = 2, \dots, n-2$ , can be generated using

```

mk = ((N-1)-(n-2))/(n-1);
A{k,n} = sparse(mk,n-2);
A{k,n}(1, k-1) = -1;
A{k,n}(mk, k) = -1;
A{n,k} = A{k,n}';

```

To compute matrices like  $A^{-1}B$  you may use the MATLAB 'backslash' command `A\B`.