

- Assuming no pivoting is needed (to avoid breakdown or to ensure numerical stability), devise an efficient way to arrange the computations for solving an  $n$ -by- $n$  linear system with non-zero entries in the coefficient matrix only in the first and last rows and columns and also in the two main diagonals. In the case of a  $9 \times 9$  matrix, the non-zero entries would appear as follows:

$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & & & & & & \bullet & \bullet \\ \bullet & & \bullet & & & & & \bullet & \bullet \\ \bullet & & & \bullet & & & & \bullet & \bullet \\ \bullet & & & & \bullet & & & \bullet & \bullet \\ \bullet & & & & & \bullet & & \bullet & \bullet \\ \bullet & \bullet & & & & & & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix} \begin{bmatrix} x \\ x \\ x \\ x \\ x \\ x \\ x \\ x \\ x \end{bmatrix} = \begin{bmatrix} b \\ b \\ b \\ b \\ b \\ b \\ b \\ b \\ b \end{bmatrix},$$

where  $\bullet$ 's represent a non-zero entries and blanks represent zero entries. It is sufficient to illustrate the efficient method for this  $9 \times 9$  case. Note, "efficient" here means your algorithm should require  $O(n)$  operations. Also you should never introduce a non-zero entry in the matrix where originally there was a zero.

- Cholesky decomposition

- Implement the Cholesky algorithm discussed in class for computing the Cholesky decomposition of a symmetric positive definite (SPD) matrix  $A$ . You should create a MATLAB function similar to the `lutx` function from the NCM book. However, it should return two values, the first is the lower triangular matrix  $L$  from the Cholesky decomposition (recall  $A = LL^T$ ), and the second is a scalar value that indicates whether or not the matrix  $A$  is positive definite. The function declaration should be something like

$$[L, q] = \text{cholmy}(A)$$

- Verify that your code works by testing it on the `minij` matrix from the matrix gallery function in MATLAB, which can be obtained with the MATLAB call

$$\gg A = \text{gallery}('minij', n)$$

where  $n$  is the number of rows (and columns) of the matrix. The Cholesky decomposition of this matrix should be a lower triangular matrix where every value in the lower triangular matrix is 1. Report your results for a small value of  $n$ .

- Use your `cholmy` function to determine which of the following family of matrices are positive definite (the statements below are the exact calls you should use in MATLAB to create given matrix):

$$\begin{aligned} M &= \text{magic}(n) \\ H &= \text{hilb}(n) \\ P &= \text{pascal}(n) \\ I &= \text{eye}(n) \\ D &= -\text{diag}(\text{ones}(n-1,1), -1) + 2 * \text{diag}(\text{ones}(n,1), 0) - \text{diag}(\text{ones}(n,1), 1) \end{aligned}$$

- The following MATLAB commands generate a "random" SPD linear system  $Ax = b$ .

```
A = rand(n);
A = A'*A;
b = rand(n,1);
```

Solve this linear system using your `cholmy` function and the `forward` and `backward` functions from the NCM book for forward and backward substitution. Report the time it takes to solve the system using this approach for  $n = 4, 8, 16, 32, 64, 128, 256,$  and  $512$  (look at the `tic` and `toc` commands for doing the timing). Compare these timing results to the timing results for solving the linear system using the `lutx`, `forward`, and `backward` functions from the NCM book for the same values of  $n$ .

Note: To get an accurate idea of the time your code is taking, you should repeat the timing calculation for each  $n$  several times and take the average.

3. For this problem you will use the Cholesky function built into MATLAB (`chol`). This function is more sophisticated than your `cholmy` function in that it efficiently handles the decomposition of a sparse symmetric positive definite matrix by exploiting any structure formed by the non-zero entries in the matrix.

- (a) Download the file “`bcsstk38.mat`” from the course web page and save it to your working directory in MATLAB. This file contains a sparse matrix that arises from a structural analysis of an engine compartment of one of Boeing’s airplanes. Load the matrix into matlab using the following series of commands:

```
>>load bcsstk38;
>>A = problem.A;
```

Use the `spy` command in matlab to generate a plot showing the sparsity pattern of the matrix `A`. Compute the sparsity ratio of the `A` matrix by comparing the number of non-zero entries in `A` (use the `nnz` command) to the total number of entries in `A`. Report this number along with some descriptive text in the title of the `spy` plot of `A`.

- (b) Compute the Cholesky decomposition of the matrix from part (a). Plot the sparsity pattern of the upper triangular matrix from the decomposition. Compute the amount of “fill-in” from the Cholesky decomposition. The “fill-in” can be defined as the ratio of the number of non-zero entries in the Cholesky decomposition to the number of non-zero entries in the original matrix. Report this number along with some descriptive text in the title of the `spy` plot of the upper triangular matrix.
  - (c) What you would like to happen is for the “fill-in” from the Cholesky decomposition to be as small as possible since this translates directly into a more computationally efficient method for solving the underlying linear system involving the matrix `A`. For any given sparse symmetric positive definite matrix, the “fill-in” that occurs from the Cholesky decomposition depends on how the rows and columns of the matrix are ordered. Several algorithms exist for permuting the rows and columns to try and minimize the “fill-in”. All these algorithms are based on results from Graph Theory. One of the most popular, and in some cases best, algorithms is called the Reverse Cuthill-McKee method. MATLAB contains a function for this method called `symrcm`. Your goal is to use this function on the original matrix `A` from part (a) to compare the “fill-in” from the Cholesky decomposition of the permuted `A` matrix to the “fill-in” from the original one. Create plots for the sparsity pattern of the permuted `A` matrix and its Cholesky decomposition. In the latter include the “fill-in”.
4. Let  $T$  be a diagonally dominant tridiagonal matrix,  $A$  be a symmetric positive definite matrix, and  $B$  and  $C$  be full nonsingular matrices. Assume all of these matrices are of size  $n$ -by- $n$ . Let  $f(\mathbf{x})$  be defined as follows

$$f(\mathbf{x}) = \mathbf{x}^T B^{-1} C T^{-1} A^{-1} \mathbf{x} + \mathbf{b}^T B^{-T} \mathbf{x},$$

where  $\mathbf{x}$  and  $\mathbf{b}$  are column vectors of size  $n$ . Using your `cholmy` function and the `lutx`, `forward`, `backward`, and `tridisolve` functions from the NCM book, show how you would efficiently evaluate the function  $f(\mathbf{x})$  for many  $\mathbf{x}$  in MATLAB. Write a MATLAB routine showing exactly the sequence of steps you would use. Test your code using the following definitions for  $T$ ,  $A$ ,  $B$ ,  $C$ ,  $\mathbf{b}$ , and  $\mathbf{x}$ :

```
>>T = diag(ones(n-1,1),-1) - 2*diag(ones(n,1),0) + diag(ones(n,1),1);
>>A = tril(rand(n)); >>A = A'*A; >>B = rand(n); >>C = rand(n); >>b = rand(n,1); >>x = ones(n,1);
```

Note: no where in your code should you actually compute the inverse of a matrix.

5. Consider the linear system of equation  $Ax = b$ , where

$$A = \begin{bmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \end{bmatrix}, \quad b = \begin{bmatrix} 0.882 \\ 0.744 \\ 0.618 \\ 0.521 \\ 0.447 \end{bmatrix}.$$

Suppose we in some way have obtained the approximate solution vector

$$\bar{x} = \begin{bmatrix} -2.1333 \\ 0.6258 \\ 17.4552 \\ -11.8692 \\ -1.4994 \end{bmatrix}.$$

It is then easy to show that the residual becomes *exactly*

$$A\bar{x} - b = \begin{bmatrix} 0.00001 \\ -0.00001 \\ 0.00001 \\ -0.00001 \\ 0.00001 \end{bmatrix}.$$

- Does this imply that  $\bar{x}$  is close to the exact solution  $x$ ?
- Use MATLAB to obtain an accurate solution to the given system.
- Use MATLAB again to obtain a condition number for  $A$ . Use the appropriate result on perturbations of the right hand side (RHS) of a linear system to confirm that this very small residual indeed is big enough to allow for the solution to be as far away from the correct one as occurs in this example.

Hint: The  $A$  matrix can be constructed easily in MATLAB using the function `hankel` (use MATLAB `help` to see why). The following code constructs  $A$ :

```
A = 1./hankel(2:6,6:10).
```

The condition number (with respect to the two-norm) can be computed using the MATLAB function `cond`:

```
cond(A)
```