

## Numerical Computing Programming Assignment 4

This homework has both written and programming parts. Think about the written questions before beginning the programming.

A tridiagonal system of equations has a matrix where only the subdiagonal, diagonal and superdiagonal have non-zero elements. (For examples see the matrices  $S$  and  $T$  below).

**Written part:** Answer the following questions on paper:

1. Show that without pivoting, the LU factors of a tridiagonal matrix are also tridiagonal.
2. With pivoting, how many non-zero superdiagonals can there be?
3. For a tridiagonal matrix, what is the operation count for LU factorization?
4. For a tridiagonal matrix, what is the operation count for a matrix-vector multiply?
5. For a tridiagonal matrix, what is the operation count for a forward or backward substitution using one of the resulting triangular factors?

**Programming part:**

1. Write a program to solve a *tridiagonal* system of equations. Your program should first factor the matrix, then do forward and backward substitution to solve the system. You can assume that no pivoting is needed and that the factorization is stable without pivoting. You will need to save the multipliers to apply them to many right-hand-sides.

You should modify the LU factorization algorithm so that the loops only go over elements that are not zero. Your program should be written in a modular fashion, so the routines that factor, solve, multiply, copy, initialize, etc. can be easily called in part 2 of this assignment, and can be re-used in later assignments. Test your program by writing a routine that multiplies  $L$  times  $U$  and compares it with  $A$ .

2. Run your program using the tridiagonal matrices  $S$  and  $T$  and vector  $b$  specified below in the following way. You have an initial vector  $b$ . Let  $\mathbf{u}_0 = 0$ . First solve  $S\mathbf{u}_1 = T\mathbf{u}_0 + b$ . Next solve  $S\mathbf{u}_2 = T\mathbf{u}_1 + b$  for  $\mathbf{u}_2$ . Then solve  $S\mathbf{u}_3 = T\mathbf{u}_2 + b$ . Continue in this way until  $\mathbf{u}_{n+1} = \mathbf{u}_n$ . (Note: this last statement should be interpreted as  $\|\mathbf{u}_{n+1} - \mathbf{u}_n\|$  is small, in some norm. Be sure to include in your write-up how you tested for this.) Print the final  $\mathbf{u}$ , and if you can plot it as well in any way that you can, (For example, you can plot it by hand, you can use excel, you can use matlab, you can use gnuplot, there are lots of free plotting programs).

For a given size matrix  $n$ , and choice of number  $r$ , the matrices  $S, T$  and  $b$  are:

$$S = \begin{pmatrix} 1+r & -r/2 & 0 & 0 & \dots & 0 \\ -r/2 & 1+r & -r/2 & 0 & \dots & 0 \\ 0 & -r/2 & 1+r & -r/2 & \dots & 0 \\ 0 & 0 & -r/2 & 1+r & -r/2 & 0 \\ 0 & \dots & 0 & -r/2 & 1+r & -r/2 \\ 0 & \dots & 0 & 0 & -r/2 & 1+r \end{pmatrix}$$

$$T = \begin{pmatrix} 1-r & r/2 & 0 & 0 & \dots & 0 \\ r/2 & 1-r & r/2 & 0 & \dots & 0 \\ 0 & r/2 & 1-r & r/2 & \dots & 0 \\ 0 & 0 & r/2 & 1-r & r/2 & 0 \\ 0 & \dots & 0 & r/2 & 1-r & r/2 \\ 0 & \dots & 0 & 0 & r/2 & 1-r \end{pmatrix}$$

and

$$b = r/(n+1) * \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

where the 1 is in the  $(n+1)/2$  position (the middle of the vector).

Choose a small number, say  $n = 11$ , to debug your program, but then try running it with  $n = 21$  or  $31$ . For this choice of vector  $b$ , nothing should change in the final solution as  $n$  varies. Also try different values for  $r$ , e.g.  $r = 10$ . or  $r = 50$ , and notice how many iterations it takes before the solution stops changing.