Worksheet: Solving tridiagonal systems

From now on in this course you should remember that, for a general linear system $A\mathbf{x} = \mathbf{b}$ with n equations and n unknowns, the total number of operations is $\frac{2}{3}n^3 + O(n^2)$. This means that, on a modern computer, systems with $n = 10^4$ or more equations are noticably slow because $(10^4)^3 = 10^{12}$ is a trillion, and your laptop takes much more than a second to do a trillion operations. Systems with $n = 10^5$ may take hours on your laptop.

However, systems with $n = 10^7$ and larger are routinely solved on modern computers by exploiting the fact that the matrices A which come from science, engineering, finance, and other applications are *not* full collections of random numbers. It is very common for most of the entries of A to be zero; one says that A is *sparse* in that case.

In particular, *tridiagonal* matrices are common:

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & a_{23} & 0 & \dots & 0 \\ 0 & a_{32} & a_{33} & a_{34} & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \\ \vdots & & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & \dots & & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

The pattern is that the only entries which are nonzero are those on the main diagonal, and just above and below it.

Example. The linear system

$$\begin{aligned} 2x_1 - x_2 \qquad\qquad\qquad\qquad &= 4 \\ -x_1 + 2x_2 - x_3 \qquad\qquad\quad &= -4 \\ -x_2 + 2x_3 - x_4 \qquad\quad &= 4 \\ -x_3 + 2x_4 - x_5 &= -3 \\ -x_4 + 2x_5 &= 2 \end{aligned}$$

might come from approximating a differential equation boundary value problem. It has solution $\mathbf{x} = [2, 0, 2, 0, 1]^\top$.

Part 1. *Check* (verify) the above solution by hand.

Part 2. Start doing Gaussian elimination on a matrix like A above. Note that you only need to process one row per stage, and only a couple of entries of the row will change. Also think about how back-substitution will work; it is also easier than the general case.

Part 3. Write a MATLAB code for a function, as follows, which solves a tridiagonal linear system $A\mathbf{x} = \mathbf{b}$ with n equations and n unknowns. Assume A has the form on the previous page. Your code should not touch (use) the locations of A which are zero. The code should start by extracting n from the input A or b, and it should do some input checking before proceeding.

```
function x = tridiag(A,b)
```

Part 4. How many operations does your code do?