you may wish to experiment with a banded system solver or more general sparse solver, although this particular problem instance is too small for these to offer significant advantage over a general solver.

2.4. Write a routine for estimating the condition number of a matrix A. You may use either the 1-norm or the ∞-norm (or try both and compare the results). You will need to compute $\|A\|$, which is easy, and estimate $\|A^{-1}\|$, which is more challenging. As discussed in Section 2.3.3, one way to estimate $\|A^{-1}\|$ is to choose a vector y such that the ratio $\|z\|/\|y\|$ is large, where z is the solution to $Az = y$. Try two different approaches to choosing y:

(a) Choose y as the solution to the system $A^T y = c$, where c is a vector each of whose components is ± 1, with the sign for each component chosen by the following heuristic. Using the factorization $A = LU$, the system $A^T y = c$ is solved in two stages, successively solving the triangular systems $U^T v = c$ and $L^T y = v$. At each step of the first triangular solution, choose the corresponding component of c to be 1 or -1, depending on which will make the resulting component of v larger in magnitude. (You will need to write a custom triangular solution routine to implement this.) Then solve the second triangular system in the usual way for y. The idea here is that any ill-conditioning in A will be reflected in U, resulting in a relatively large v. The relatively well-conditioned unit triangular matrix L will then preserve this relationship, resulting in a relatively large y.

(b) Choose some small number, say, five, different vectors y randomly and use the one producing the largest ratio $\|z\|/\|y\|$. (For this you can use an ordinary triangular solution routine.)

You may use a library routine to obtain the necessary LU factorization of A. Test both of the approaches on each of the following matrices:

$$A_1 = \begin{bmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} -73 & 78 & 24 \\ 92 & 66 & 25 \\ -80 & 37 & 10 \end{bmatrix}.$$

How do the results using these two methods compare? To check the quality of your estimates, compute A^{-1} explicitly to determine its true norm (this computation can also make use of the LU factorization already computed). If you have access to linear equations software that already includes a condition estimator, how do your results compare with its?

2.5. (a) Use a single-precision routine for Gaussian elimination to solve the system $Ax = b$, where

$$A = \begin{bmatrix} 21.0 & 67.0 & 88.0 & 73.0 \\ 76.0 & 63.0 & 7.0 & 20.0 \\ 0.0 & 85.0 & 56.0 & 54.0 \\ 19.3 & 43.0 & 30.2 & 29.4 \end{bmatrix},$$

$$b = \begin{bmatrix} 141.0 \\ 109.0 \\ 218.0 \\ 93.7 \end{bmatrix}.$$

(b) Compute the residual $r = b - Ax$ using double-precision arithmetic, if available (but storing the final result in a single-precision vector r). Note that the solution routine may destroy the array containing A, so you may need to save a separate copy for computing the residual. (If only one precision is available in the computing environment you use, then do all of this problem in that precision.)

(c) Solve the linear system $Az = r$ to obtain the "improved" solution $x + z$. Note that A need not be refactored.

(d) Repeat steps b and c until no further improvement is observed.

2.6. An $n \times n$ *Hilbert* matrix H has entries $h_{ij} = 1/(i + j - 1)$, so it has the form

$$\begin{bmatrix} 1 & 1/2 & 1/3 & \cdots \\ 1/2 & 1/3 & 1/4 & \cdots \\ 1/3 & 1/4 & 1/5 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

For $n = 2, 3, \ldots$, generate the Hilbert matrix of order n, and also generate the n-vector $b = Hx$, where x is the n-vector with all of its components equal to 1. Use a library routine for Gaussian elimination (or Cholesky factorization, since the Hilbert matrix is symmetric and positive definite)