Computer Problems

2.1. (*a*) Show that the matrix

$$A = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \end{bmatrix}$$

is singular. Describe the set of solutions to the system $Ax = b$ if

$$b = \begin{bmatrix} 0.1 \\ 0.3 \\ 0.5 \end{bmatrix}.$$

(*b*) If we were to use Gaussian elimination with partial pivoting to solve this system using exact arithmetic, at what point would the process fail?

(*c*) Because some of the entries of A are not exactly representable in a binary floating-point system, the matrix is no longer exactly singular when entered into a computer; thus, solving the system by Gaussian elimination will not necessarily fail. Solve this system on a computer using a library routine for Gaussian elimination. Compare the computed solution with your description of the solution set in part *a*. If your software includes a condition estimator, what is the estimated value for cond(A)? How many digits of accuracy in the solution would this lead you to expect?

2.2. (*a*) Use a library routine for Gaussian elimination to solve the system $Ax = b$, where

$$A = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -1 & 7 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix}.$$

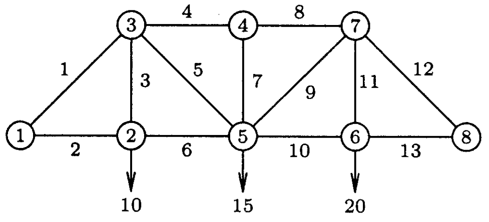
(*b*) Use the LU factorization of A already computed to solve the system $Ay = c$, where

$$c = \begin{bmatrix} 4 \\ 8 \\ -6 \end{bmatrix},$$

without refactoring the matrix.

(*c*) If the matrix A changes so that $a_{1,2} = 2$, use the Sherman-Morrison updating technique to compute the new solution x without refactoring the matrix, using the original right-hand-side vector b.

2.3. The following diagram depicts a plane truss having 13 members (the numbered lines) connected by 10 joints (the numbered circles). The indicated loads, in tons, are applied at joints 2, 5, and 6, and we wish to determine the resulting force on each member of the truss.



For the truss to be in static equilibrium, there must be no net force, horizontally or vertically, at any joint. Thus, we can determine the member forces by equating the horizontal forces to the left and right at each joint, and similarly equating the vertical forces upward and downward at each joint. For the eight joints, this would give 16 equations, which is more than the 13 unknown forces to be determined. For the truss to be statically determinate, that is, for there to be a unique solution, we assume that joint 1 is rigidly fixed both horizontally and vertically, and that joint 8 is fixed vertically. Resolving the member forces into horizontal and vertical components and defining $\alpha = \sqrt{2}/2$, we obtain the following system of equations for the member forces f_i:

$$\begin{aligned}
\text{Joint 2}: &\begin{cases} f_2 = f_6 \\ f_3 = 10 \end{cases} \\
\text{Joint 3}: &\begin{cases} \alpha f_1 = f_4 + \alpha f_5 \\ \alpha f_1 + f_3 + \alpha f_5 = 0 \end{cases} \\
\text{Joint 4}: &\begin{cases} f_4 = f_8 \\ f_7 = 0 \end{cases} \\
\text{Joint 5}: &\begin{cases} \alpha f_5 + f_6 = \alpha f_9 + f_{10} \\ \alpha f_5 + f_7 + \alpha f_9 = 15 \end{cases} \\
\text{Joint 6}: &\begin{cases} f_{10} = f_{13} \\ f_{11} = 20 \end{cases} \\
\text{Joint 7}: &\begin{cases} f_8 + \alpha f_9 = \alpha f_{12} \\ \alpha f_9 + f_{11} + \alpha f_{12} = 0 \end{cases} \\
\text{Joint 8}: &\begin{cases} f_{13} + \alpha f_{12} = 0 \end{cases}
\end{aligned}$$

Use a library routine to solve this system of linear equations for the vector f of member forces. Note that the matrix of this system is quite sparse, so 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 $\boldsymbol{A}$. You may use either the 1-norm or the ∞-norm (or try both and compare the results). You will need to compute $\|\boldsymbol{A}\|$, which is easy, and estimate $\|\boldsymbol{A}^{-1}\|$, which is more challenging. As discussed in Section 2.3.3, one way to estimate $\|\boldsymbol{A}^{-1}\|$ is to choose a vector $\boldsymbol{y}$ such that the ratio $\|\boldsymbol{z}\|/\|\boldsymbol{y}\|$ is large, where $\boldsymbol{z}$ is the solution to $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{y}$. Try two different approaches to choosing $\boldsymbol{y}$:

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

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

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

$$\boldsymbol{A}_1 = \begin{bmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{bmatrix},$$

$$\boldsymbol{A}_2 = \begin{bmatrix} -73 & 78 & 24 \\ 92 & 66 & 25 \\ -80 & 37 & 10 \end{bmatrix}.$$

How do the results using these two methods compare? To check the quality of your estimates, compute $\boldsymbol{A}^{-1}$ 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 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where

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

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

(*b*) Compute the residual $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ using double-precision arithmetic, if available (but storing the final result in a single-precision vector $\boldsymbol{r}$). Note that the solution routine may destroy the array containing $\boldsymbol{A}$, 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 $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{r}$ to obtain the "improved" solution $\boldsymbol{x} + \boldsymbol{z}$. Note that $\boldsymbol{A}$ need not be refactored.

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

2.6. An $n \times n$ *Hilbert* matrix $\boldsymbol{H}$ 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 $\boldsymbol{b} = \boldsymbol{H}\boldsymbol{x}$, where $\boldsymbol{x}$ 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) to solve the resulting linear system $\boldsymbol{Hx} = \boldsymbol{b}$, obtaining an approximate solution $\hat{\boldsymbol{x}}$. Compute the ∞-norm of the residual $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{H}\hat{\boldsymbol{x}}$ and of the error $\Delta\boldsymbol{x} = \hat{\boldsymbol{x}} - \boldsymbol{x}$, where $\boldsymbol{x}$ is the vector of all ones. How large can you take n before the error is 100 percent (i.e., there are no significant digits in the solution)? Also use a condition estimator to obtain cond($\boldsymbol{H}$) for each value of n. Try to characterize the condition number as a function of n. As n varies, how does the number of correct digits in the components of the computed solution relate to the condition number of the matrix?

2.7. (*a*) What happens when Gaussian elimination with partial pivoting is used on a matrix of the following form?

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

Do the entries of the transformed matrix grow? What happens if complete pivoting is used instead? (Note that part *a* does not require a computer.)

(*b*) Use a library routine for Gaussian elimination with partial pivoting to solve various sizes of linear systems of this form, using right-hand-side vectors chosen so that the solution is known. How do the error, residual, and condition number behave as the systems become larger? This artificially contrived system illustrates the worst-case growth factor cited in Section 2.3.5 and is not indicative of the usual behavior of Gaussian elimination with partial pivoting.

2.8. Multiplying both sides of a linear system $\boldsymbol{Ax} = \boldsymbol{b}$ by a nonsingular diagonal matrix $\boldsymbol{D}$ to obtain a new system $\boldsymbol{DAx} = \boldsymbol{Db}$ simply rescales the rows of the system and in theory does not change the solution. Such scaling does affect the condition number of the matrix and the choice of pivots in Gaussian elimination, however, so it may affect the accuracy of the solution in finite-precision arithmetic. Note that scaling can introduce some rounding error in the matrix unless the entries of $\boldsymbol{D}$ are powers of the base of the floating-point arithmetic system being used (why?).

Using a linear system with randomly chosen matrix $\boldsymbol{A}$, and right-hand-side vector $\boldsymbol{b}$ chosen so that the solution is known, experiment with various scaling matrices $\boldsymbol{D}$ to see what effect they have on the condition number of the matrix $\boldsymbol{DA}$ and the solution given by a library routine for solving the linear system $\boldsymbol{DAx} = \boldsymbol{Db}$. Be sure to try some fairly skewed scalings, where the magnitudes of the diagonal entries of $\boldsymbol{D}$ vary widely (the purpose is to simulate a system with badly chosen units). Compare both the relative residuals and the error given by the various scalings. Can you find a scaling that gives very poor accuracy? Is the residual still small in this case?

2.9. (*a*) Use Gaussian elimination *without* pivoting to solve the linear system

$$\begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1+\epsilon \\ 2 \end{bmatrix}$$

for $\epsilon = 10^{-2k}$, $k = 1, \ldots, 10$. The exact solution is $x = [1 \quad 1]^T$, independent of the value of ϵ. How does the accuracy of the computed solution behave as the value of ϵ decreases?

(*b*) Repeat part *a*, still using Gaussian elimination without pivoting, but this time use one iteration of iterative refinement to improve the solution, computing the residual in the same precision as the rest of the computations. Now how does the accuracy of the computed solution behave as the value of ϵ decreases?

2.10. Consider the linear system

$$\begin{bmatrix} 1 & 1+\epsilon \\ 1-\epsilon & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1+(1+\epsilon)\epsilon \\ 1 \end{bmatrix},$$

where ϵ is a small parameter to be specified. The exact solution is obviously

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ \epsilon \end{bmatrix}$$

for any value of ϵ.

Use a library routine based on Gaussian elimination to solve this system. Experiment with various values for ϵ, especially values near $\sqrt{\epsilon_{\text{mach}}}$ for your computer. For each value of ϵ you try, compute an estimate of the condition number of the matrix and the relative error in each component of the solution. How accurately is each component determined? How does the accuracy attained for each component compare with expectations based on the condition number of the matrix and the error bounds given in Section 2.3.4? What conclusions can you draw from this experiment?