2.8 Effect of Roundoff Errors

The rounding errors introduced during the solution of a linear system of equations almost always cause the computed solution—which we now denote by x_*—to differ somewhat from the theoretical solution $x = A^{-1}b$. In fact, if the elements of x are not floating-point numbers, then x_* cannot equal x. There are two common measures of the discrepancy in x_*: the *error*,

$$e = x - x_*,$$

and the *residual*,

$$r = b - Ax_*.$$

Matrix theory tells us that, because A is nonsingular, if one of these is zero, the other must also be zero. But they are not necessarily both "small" at the same time. Consider the following example:

$$\begin{pmatrix} 0.780 & 0.563 \\ 0.913 & 0.659 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.217 \\ 0.254 \end{pmatrix}.$$

What happens if we carry out Gaussian elimination with partial pivoting on a hypothetical three-digit decimal computer? First, the two rows (equations) are interchanged so that 0.913 becomes the pivot. Then the multiplier

$$\frac{0.780}{0.913} = 0.854 \quad \text{(to three places)}$$

is computed. Next, 0.854 times the new first row is subtracted from the new second row to produce the system

$$\begin{pmatrix} 0.913 & 0.659 \\ 0 & 0.001 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.254 \\ 0.001 \end{pmatrix}.$$

Finally, the back substitution is carried out:

$$\begin{aligned} x_2 &= \frac{0.001}{0.001} = 1.00 \quad \text{(exactly)}, \\ x_1 &= \frac{0.254 - 0.659x_2}{0.913} \\ &= -0.443 \quad \text{(to three places)}. \end{aligned}$$

Thus the computed solution is

$$x_* = \begin{pmatrix} -0.443 \\ 1.000 \end{pmatrix}.$$

To assess the accuracy without knowing the exact answer, we compute the residuals (exactly):

$$\begin{aligned} r = b - Ax_* &= \begin{pmatrix} 0.217 - ((0.780)(-0.443) + (0.563)(1.00)) \\ 0.254 - ((0.913)(-0.443) + (0.659)(1.00)) \end{pmatrix} \\ &= \begin{pmatrix} -0.000460 \\ -0.000541 \end{pmatrix}. \end{aligned}$$

The *residuals* are less than 10^{-3}. We could hardly expect better on a three-digit machine. However, it is easy to see that the exact solution to this system is

$$x = \begin{pmatrix} 1.000 \\ -1.000 \end{pmatrix}.$$

So the components of our computed solution actually have the wrong signs; the *error* is larger than the solution itself.

Were the small residuals just a lucky fluke? You should realize that this example is highly contrived. The matrix is very close to being singular and is *not* typical of most problems encountered in practice. Nevertheless, let us track down the reason for the small residuals.

If Gaussian elimination with partial pivoting is carried out for this example on a computer with six or more digits, the forward elimination will produce a system something like

$$\begin{pmatrix} 0.913000 & 0.659000 \\ 0 & -0.000001 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.254000 \\ 0.000001 \end{pmatrix}.$$

Notice that the sign of $U_{2,2}$ differs from that obtained with three-digit computation. Now the back substitution produces

$$\begin{aligned} x_2 &= \frac{0.000001}{-0.000001} = -1.00000, \\ x_1 &= \frac{0.254 - 0.659x_2}{0.913} \\ &= 1.00000, \end{aligned}$$

the exact answer. On our three-digit machine, x_2 was computed by dividing two quantities, both of which were on the order of rounding errors and one of which did not even have the correct sign. Hence x_2 can turn out to be almost anything. Then this arbitrary value of x_2 was substituted into the first equation to obtain x_1.

We can reasonably expect the residual from the first equation to be small—x_1 was computed in such a way as to make this certain. Now comes a subtle but crucial point. We can also expect the residual from the second equation to be small, *precisely because the matrix is so close to being singular.* The two equations are very nearly multiples of one another, so any pair (x_1, x_2) that nearly satisfies the first equation will also nearly satisfy the second. If the matrix were known to be exactly singular, we would not need the second equation at all—any solution of the first would automatically satisfy the second.

In Figure 2.1, the exact solution is marked with a circle and the computed solution with an asterisk. Even though the computed solution is far from the exact intersection, it is close to both lines because they are nearly parallel.

Although this example is contrived and atypical, the conclusion we reached is not. It is probably the single most important fact that we have learned about matrix computation since the invention of the digital computer:

> *Gaussian elimination with partial pivoting is guaranteed to produce small residuals.*

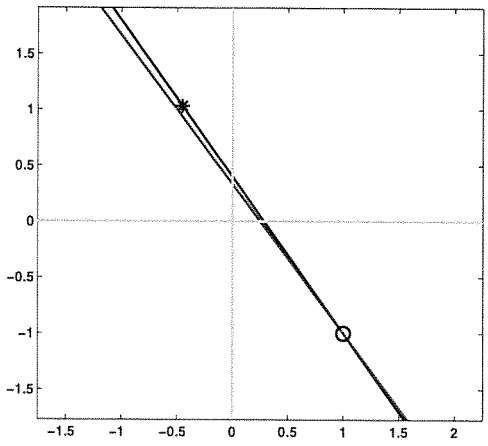

Figure 2.1. *The computed solution, marked by an asterisk, shows a large error, but a small residual.*

Now that we have stated it so strongly, we must make a couple of qualifying remarks. By "guaranteed" we mean it is possible to prove a precise theorem that assumes certain technical details about how the floating-point arithmetic system works and that establishes certain inequalities that the components of the residual must satisfy. If the arithmetic units work some other way or if there is a bug in the particular program, then the "guarantee" is void. Furthermore, by "small" we mean on the order of roundoff error *relative to* three quantities: the size of the elements of the original coefficient matrix, the size of the elements of the coefficient matrix at intermediate steps of the elimination process, and the size of the elements of the computed solution. If any of these are "large," then the residual will not necessarily be small in an absolute sense. Finally, even if the residual is small, we have made no claims that the error will be small. The relationship between the size of the residual and the size of the error is determined in part by a quantity known as the *condition number* of the matrix, which is the subject of the next section.

2.9 Norms and Condition Numbers

The coefficients in the matrix and right-hand side of a system of simultaneous linear equations are rarely known exactly. Some systems arise from experiments, and so the coefficients are subject to observational errors. Other systems have coefficients given by formulas that involve roundoff error in their evaluation. Even if the system can be stored exactly in the computer, it is almost inevitable that roundoff errors will be introduced during its solution. It can be shown that roundoff errors in Gaussian elimination have the same effect on the answer as errors in the original

coefficients.

Consequently, we are led to a fundamental question. If perturbations are made in the coefficients of a system of linear equations, how much is the solution altered? In other words, if $Ax = b$, how can we measure the sensitivity of x to changes in A and b?

The answer to this question lies in making the idea of *nearly singular* precise. If A is a singular matrix, then for some b's a solution x will not exist, while for others it will not be unique. So if A is nearly singular, we can expect small changes in A and b to cause very large changes in x. On the other hand, if A is the identity matrix, then b and x are the same vector. So if A is nearly the identity, small changes in A and b should result in correspondingly small changes in x.

At first glance, it might appear that there is some connection between the size of the pivots encountered in Gaussian elimination with partial pivoting and *nearness to singularity*, because if the arithmetic could be done exactly, all the pivots would be nonzero if and only if the matrix is nonsingular. To some extent, it is also true that if the pivots are small, then the matrix is *close to singular*. However, when roundoff errors are encountered, the converse is no longer true—a matrix might be close to singular even though none of the pivots are small.

To get a more precise, and reliable, measure of nearness to singularity than the size of the pivots, we need to introduce the concept of a *norm* of a vector. This is a single number that measures the general size of the elements of the vector. The family of vector norms known as l_p depends on a parameter p in the range $1 \leq p \leq \infty$:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

We almost always use $p = 1$, $p = 2$, or $\lim p \to \infty$:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|,$$

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2},$$

$$\|x\|_\infty = \max_i |x_i|.$$

The l_1-norm is also known as the *Manhattan* norm because it corresponds to the distance traveled on a grid of city streets. The l_2-norm is the familiar Euclidean distance. The l_∞-norm is also known as the *Chebyshev* norm.

The particular value of p is often unimportant and we simply use $\|x\|$. All vector norms have the following basic properties associated with the notion of distance:

$$\|x\| > 0 \quad \text{if } x \neq 0,$$
$$\|0\| = 0,$$

$$\|cx\| = |c|\|x\| \text{ for all scalars } c,$$
$$\|x + y\| \le \|x\| + \|y\| \text{ (the } \textit{triangle inequality}).$$

In MATLAB, $\|x\|_p$ is computed by `norm(x,p)`, and `norm(x)` is the same as `norm(x,2)`. For example,

```
x = (1:4)/5
norm1 = norm(x,1)
norm2 = norm(x)
norminf = norm(x,inf)
```

produces

```
x =
    0.2000    0.4000    0.6000    0.8000

norm1 =
     2.0000

norm2 =
     1.0954

norminf =
     0.8000
```

Multiplication of a vector x by a matrix A results in a new vector Ax that can have a very different norm from x. This change in norm is directly related to the sensitivity we want to measure. The range of the possible change can be expressed by two numbers:

$$M = \max \frac{\|Ax\|}{\|x\|},$$
$$m = \min \frac{\|Ax\|}{\|x\|}.$$

The max and min are taken over all nonzero vectors x. Note that if A is singular, then $m = 0$. The ratio M/m is called the *condition number* of A:

$$\kappa(A) = \frac{\max \frac{\|Ax\|}{\|x\|}}{\min \frac{\|Ax\|}{\|x\|}}.$$

The actual numerical value of $\kappa(A)$ depends on the vector norm being used, but we are usually only interested in order of magnitude estimates of the condition number, so the particular norm is usually not very important.

Consider a system of equations

$$Ax = b$$

and a second system obtained by altering the right-hand side:

$$A(x + \delta x) = b + \delta b.$$

We think of δb as being the error in b and δx as being the resulting error in x, although we need not make any assumptions that the errors are small. Because $A(\delta x) = \delta b$, the definitions of M and m immediately lead to

$$\|b\| \leq M\|x\|$$

and

$$\|\delta b\| \geq m\|\delta x\|.$$

Consequently, if $m \neq 0$,

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A)\frac{\|\delta b\|}{\|b\|}.$$

The quantity $\|\delta b\|/\|b\|$ is the *relative* change in the right-hand side, and the quantity $\|\delta x\|/\|x\|$ is the *relative* error caused by this change. The advantage of using relative changes is that they are *dimensionless*, that is, they are not affected by overall scale factors.

This shows that the condition number is a relative error magnification factor. Changes in the right-hand side can cause changes $\kappa(A)$ times as large in the solution. It turns out that the same is true of changes in the coefficient matrix itself.

The condition number is also a measure of nearness to singularity. Although we have not yet developed the mathematical tools necessary to make the idea precise, the condition number can be thought of as the reciprocal of the relative distance from the matrix to the set of singular matrices. So, if $\kappa(A)$ is large, A is close to singular.

Some of the basic properties of the condition number are easily derived. Clearly, $M \geq m$, and so

$$\kappa(A) \geq 1.$$

If P is a permutation matrix, then the components of Px are simply a rearrangement of the components of x. It follows that $\|Px\| = \|x\|$ for all x, and so

$$\kappa(P) = 1.$$

In particular, $\kappa(I) = 1$. If A is multiplied by a scalar c, then M and m are both multiplied by the same scalar, and so

$$\kappa(cA) = \kappa(A).$$

If D is a diagonal matrix, then

$$\kappa(D) = \frac{\max|d_{ii}|}{\min|d_{ii}|}.$$

These last two properties are two of the reasons that $\kappa(A)$ is a better measure of nearness to singularity than the determinant of A. As an extreme example, consider

a 100-by-100 diagonal matrix with 0.1 on the diagonal. Then $\det(A) = 10^{-100}$, which is usually regarded as a small number. But $\kappa(A) = 1$, and the components of Ax are simply 0.1 times the corresponding components of x. For linear systems of equations, such a matrix behaves more like the identity than like a singular matrix.

The following example uses the l_1-norm:

$$A = \begin{pmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{pmatrix},$$
$$b = \begin{pmatrix} 4.1 \\ 9.7 \end{pmatrix},$$
$$x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Clearly, $Ax = b$, and

$$\|b\| = 13.8, \quad \|x\| = 1.$$

If the right-hand side is changed to

$$\tilde{b} = \begin{pmatrix} 4.11 \\ 9.70 \end{pmatrix},$$

the solution becomes

$$\tilde{x} = \begin{pmatrix} 0.34 \\ 0.97 \end{pmatrix}.$$

Let $\delta b = b - \tilde{b}$ and $\delta x = x - \tilde{x}$. Then

$$\|\delta b\| = 0.01,$$
$$\|\delta x\| = 1.63.$$

We have made a fairly small perturbation in b that completely changes x. In fact, the relative changes are

$$\frac{\|\delta b\|}{\|b\|} = 0.0007246,$$
$$\frac{\|\delta x\|}{\|x\|} = 1.63.$$

Because $\kappa(A)$ is the maximum magnification factor,

$$\kappa(A) \geq \frac{1.63}{0.0007246} = 2249.4.$$

We have actually chosen the b and δb that give the maximum, and so, for this example with the l_1-norm,

$$\kappa(A) = 2249.4.$$

It is important to realize that this example is concerned with the *exact* solutions to two slightly different systems of equations and that the method used to

obtain the solutions is irrelevant. The example is constructed to have a fairly large condition number so that the effect of changes in b is quite pronounced, but similar behavior can be expected in any problem with a large condition number.

The condition number also plays a fundamental role in the analysis of the roundoff errors introduced during the solution by Gaussian elimination. Let us assume that A and b have elements that are exact floating-point numbers, and let x_* be the vector of floating-point numbers obtained from a linear equation solver such as the function we shall present in the next section. We also assume that exact singularity is not detected and that there are no underflows or overflows. Then it is possible to establish the following inequalities:

$$\frac{\|b - Ax_*\|}{\|A\|\|x_*\|} \leq \rho\epsilon,$$
$$\frac{\|x - x_*\|}{\|x_*\|} \leq \rho\kappa(A)\epsilon.$$

Here ϵ is the relative machine precision `eps` and ρ is defined more carefully later, but it usually has a value no larger than about 10.

The first inequality says that the *relative residual* can usually be expected to be about the size of roundoff error, no matter how badly conditioned the matrix is. This was illustrated by the example in the previous section. The second inequality requires that A be nonsingular and involves the exact solution x. It follows directly from the first inequality and the definition of $\kappa(A)$ and says that the *relative error* will also be small if $\kappa(A)$ is small but might be quite large if the matrix is nearly singular. In the extreme case where A is singular but the singularity is not detected, the first inequality still holds but the second has no meaning.

To be more precise about the quantity ρ, it is necessary to introduce the idea of a *matrix norm* and establish some further inequalities. Readers who are not interested in such details can skip the remainder of this section. The quantity M defined earlier is known as the norm of the matrix. The notation for the matrix norm is the same as for the vector norm:

$$\|A\| = \max \frac{\|Ax\|}{\|x\|}.$$

It is not hard to see that $\|A^{-1}\| = 1/m$, so an equivalent definition of the condition number is

$$\kappa(A) = \|A\|\|A^{-1}\|.$$

Again, the actual numerical values of the matrix norm and condition number depend on the underlying vector norm. It is easy to compute the matrix norms corresponding to the l_1 and l_∞ vector norms. In fact, it is not hard to show that

$$\|A\|_1 = \max_j \sum_i |a_{i,j}|,$$
$$\|A\|_\infty = \max_i \sum_j |a_{i,j}|.$$

Computing the matrix norm corresponding to the l_2 vector norm involves the singular value decomposition (SVD), which is discussed in a later chapter. MATLAB computes matrix norms with `norm(A,p)` for `p = 1, 2,` or `inf`.

The basic result in the study of roundoff error in Gaussian elimination is due to J. H. Wilkinson. He proved that the computed solution x_* exactly satisfies

$$(A + E)x_* = b,$$

where E is a matrix whose elements are about the size of roundoff errors in the elements of A. There are some rare situations where the intermediate matrices obtained during Gaussian elimination have elements that are larger than those of A, and there is some effect from accumulation of rounding errors in large matrices, but it can be expected that if ρ is defined by

$$\frac{\|E\|}{\|A\|} = \rho\epsilon,$$

then ρ will rarely be bigger than about 10.

From this basic result, we can immediately derive inequalities involving the residual and the error in the computed solution. The residual is given by

$$b - Ax_* = Ex_*,$$

and hence

$$\|b - Ax_*\| = \|Ex_*\| \leq \|E\|\|x_*\|.$$

The residual involves the product Ax_*, so it is appropriate to consider the *relative residual*, which compares the norm of $b - Ax$ to the norms of A and x_*. It follows directly from the above inequalities that

$$\frac{\|b - Ax_*\|}{\|A\|\|x_*\|} \leq \rho\epsilon.$$

If A is nonsingular, the error can be expressed using the inverse of A by

$$x - x_* = A^{-1}(b - Ax_*),$$

and so

$$\|x - x_*\| \leq \|A^{-1}\|\|E\|\|x_*\|.$$

It is simplest to compare the norm of the error with the norm of the computed solution. Thus the *relative error* satisfies

$$\frac{\|x - x_*\|}{\|x_*\|} \leq \rho\|A\|\|A^{-1}\|\epsilon.$$

Hence

$$\frac{\|x - x_*\|}{\|x_*\|} \leq \rho\kappa(A)\epsilon.$$

The actual computation of $\kappa(A)$ requires knowing $\|A^{-1}\|$. But computing A^{-1} requires roughly three times as much work as solving a single linear system. Computing the l_2 condition number requires the SVD and even more work. Fortunately, the exact value of $\kappa(A)$ is rarely required. Any reasonably good estimate of it is satisfactory.

MATLAB has several functions for computing or estimating condition numbers.

- `cond(A)` or `cond(A,2)` computes $\kappa_2(A)$. Uses `svd(A)`. Suitable for smaller matrices where the geometric properties of the l_2-norm are important.
- `cond(A,1)` computes $\kappa_1(A)$. Uses `inv(A)`. Less work than `cond(A,2)`.
- `cond(A,inf)` computes $\kappa_\infty(A)$. Uses `inv(A)`. Same as `cond(A',1)`.
- `condest(A)` estimates $\kappa_1(A)$. Uses `lu(A)` and a recent algorithm of Higham and Tisseur [9]. Especially suitable for large, sparse matrices.
- `rcond(A)` estimates $1/\kappa_1(A)$. Uses `lu(A)` and an older algorithm developed by the LINPACK and LAPACK projects. Primarily of historical interest.