What is the Condition Number of a Matrix?

CONTENTS Close

A couple of questions in comments on recent blog posts have prompted me to discuss matrix condition numbers.

In a comment on my post about Hilbert matrices, a reader named Michele asked:

- Can you comment on when the condition number gives a tight estimate of the error in a computed inverse and whether there is a better estimator?

And in a comment on my post about quadruple precision, Mark asked:

- Do you have any idea of the slowdown factor ... for large linear equation solves with extremely ill-conditioned matrices?

My short answers are that the error estimate is rarely tight, but that it is not possible to find a better one, and that it takes the same amount of time to solve ill-conditioned linear equations as it does to solve well-conditioned systems.

Contents

Condition Number for Inversion

I should first point out that there are many different condition numbers and that, although the questioners may not have realized it, they were asking about just one of them -- the condition number for inversion. In general, a condition number applies not only to a particular matrix, but also to the problem being solved. Are we inverting the matrix, finding its eigenvalues, or computing the exponential? The list goes on. A matrix can be poorly conditioned for inversion while the eigenvalue problem is well conditioned. Or, vice versa.

A condition number for a matrix and computational task measures how sensitive the answer is to perturbations in the input data and to roundoff errors made during the solution process.

When we simply say a matrix is "ill-conditioned", we are usually just thinking of the sensitivity of its inverse and not of all the other condition numbers.

Norms

In order to make these notions more precise, let's start with a vector norm. Specifically, the *Euclidean norm* or 2- *norm*.

$$\|x\| \ = \ (\sum_i x_i^2)^{1/2}$$

This is the "as the crow flies" distance in *n*-dimensional space.

The corresponding norm of a matrix A measures how much the mapping induced by that matrix can stretch vectors.

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

It is sometimes also important to consider how much a matrix can shrink vectors.

$$m \ = \ \min \frac{\|Ax\|}{\|x\|}$$

The reciprocal of the minimum stretching is the norm of the inverse, because

$$m \ = \ \min \frac{\|Ax\|}{\|x\|} \ = \ \min \frac{\|y\|}{\|A^{-1}y\|} \ = \ 1/\max \frac{\|A^{-1}y\|}{\|y\|} \ = \ 1/\|A^{-1}\|$$

A *singular* matrix is one that can map nonzero vectors into the zero vector. For a singular matrix

$$m \ = \ 0$$

and the inverse does not exist.

The ratio of the maximum to minimum stretching is the condition number for inversion.

$$\kappa(A) \ = \ \frac{M}{m}$$

An equivalent definition is

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

If a matrix is singular, then its condition number is infinite.

Linear equations

The condition number $\kappa(A)$ is involved in the answer to the question: How much can a change in the right hand side of a system of simultaneous linear equations affect the solution? Consider a system of equations

$$Ax \ = \ b$$

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

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

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 resulting *relative* change in the solution. The advantage of using relative changes is that they are *dimensionless* --- they are not affected by overall scale factors.

This inequality 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.

Example

Let's investigate a system of linear equations involving

$$A = \begin{pmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{pmatrix}$$

Take b to be the first column of A, so the solution to $Ax = b$ is simply

$$x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Switch to MATLAB

```
A = [4.1 2.8; 9.7 6.6]
b = A(:,1)
x = A\b
```

```
A =
    4.1000    2.8000
    9.7000    6.6000
b =
    4.1000
    9.7000
x =
    1.0000
   -0.0000
```

Now add `0.01` to the first component of b.

```
b2 = [4.11; 9.7]
```

```
b2 =
    4.1100
    9.7000
```

The solution changes dramatically.

```
x2 = A\b2
```

```
x2 =
    0.3400
    0.9700
```

This sensitivity of the solution x to changes in the right hand side b is a reflection of the condition number.

```
kappa = cond(A)
```

```
kappa =
   1.6230e+03
```

The upper bound on the possible change in x indicates changes in all of the significant digits.

```
kappa*norm(b-b2)/norm(b)
```

```
ans =
    1.5412
```

The actual change in x resulting from this perturbation is

```
norm(x-x2)/norm(x)
```

```
ans =
    1.1732
```

So this particular change in the right hand side generated almost the largest possible change in the solution.

Close to singular

A large condition number means that the matrix is close to being singular. Let's make a small change in the second row of A.

```
A
A2 = [4.1 2.8; 9.676 6.608]
```

```
A =
    4.1000    2.8000
    9.7000    6.6000
A2 =
    4.1000    2.8000
    9.6760    6.6080
```

The resulting matrix is effectively singular. If we try to compute its inverse, we get a warning message.

```
A2inv = inv(A2)
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.
RCOND =  1.988677e-17.
A2inv =
   1.0e+15 *
   -1.4812    0.6276
    2.1689   -0.9190
```

The quantity RCOND in the warning is an estimate of the reciprocal of the condition number. Using the reciprocal is left over from the days before we had IEEE floating point arithmetic with `Inf` to represent overflow and infinity. For this example RCOND is on the order of `eps(1)` and the scale factor for `A2inv` implies that its elements are useless. It's impossible to compute something that doesn't exist.

A fairly recent addition to MATLAB is the function `condest` that estimates $\kappa(A)$. Now `condest(A)` is preferable to `1/rcond(A)`.

Inverse

The condition number $\kappa(A)$ also appears in the bound for how much a change E in a matrix A can affect its inverse.

$$\frac{\|(A+E)^{-1} - A^{-1}\|}{\|A^{-1}\|} \leq \kappa(A)\frac{\|E\|}{\|A\|}$$

Jim Wilkinson's work about roundoff error in Gaussian elimination showed that each column of the computed inverse is a column of the exact inverse of a matrix within roundoff error of the given matrix. Let's fudge this a bit and say that `inv(A)` computes the exact inverse of $A + E$ where $\|E\|$ is on the order of roundoff error compared to $\|A\|$.

We don't know E exactly, but for an n -by- n matrix we have the estimate

`norm(E)` $\approx$ `n*eps(norm(A))`

So we have a simple estimate for the error in the computed inverse, relative to the unknown exact inverse.

`X` = exact inverse of `A`

`Z = inv(A)`

`norm(Z - X)/norm(X)` $\approx$ `n*eps*cond(A)`

For our 2-by-2 example the estimate of the relative error in the computed inverse is

```
2*eps*condest(A)
```

```
ans =
   9.9893e-13
```

This says we can expect 12 or 13 (out of 16) significant digits.

Wilkinson had to assume that every individual floating point arithmetic operation incurs the maximum roundoff error. But only a fraction of the operations have any roundoff error at all and even for those the errors are smaller than the maximum possible. So this estimate can be expected to an overestimate. But no tighter estimate is possible.

For our example, the computed inverse is

```
format long
Z = inv(A)
```

```
Z =
 -66.000000000000057  28.000000000000025
  97.000000000000085 -41.000000000000036
```

It turns out that the exact inverse has the integer entries produced by

```
X = round(Z)
```

```
X =
   -66    28
    97   -41
```

We can compare the actual relative error with the estimate.

```
format short
norm(Z - X)/norm(X)
```

```
ans =
   8.7342e-16
```

So we actually have more than 15 significant digits of accuracy.

Speed

In simplified outline, the algorithm for computing the inverse of an n-by-n matrix, or for solving a system of n linear equations, involves loops of length n nested three deep. Each of the n^2 elements is accessed roughly n times. So the computational complexity is proportional to n^3. Lots of things can be done that affect the coefficient of proportionality, but it's still order n^3.

The actual numbers in the matrix (generally) don't affect the execution time. A nearly singular matrix can be inverted just as fast as a well-conditioned one. The answer might not be very accurate if the condition number is large, but $\kappa(A)$ does not play a role in the speed.

Published with MATLAB® R2017a
