Lecture 11

Accuracy, Condition Numbers and Pivoting

In this lecture we will discuss two separate issues of accuracy in solving linear systems. The first, *pivoting*, is a method that ensures that Gaussian elimination proceeds as accurately as possible. The second, *condition number*, is a measure of how bad a matrix is. We will see that if a matrix has a bad condition number, the solutions are unstable with respect to small changes in data.

The effect of rounding

All computers store numbers as finite strings of binary floating point digits (bits). This limits numbers to a fixed number of significant digits and implies that after even the most basic calculations, rounding must happen.

Consider the following exaggerated example. Suppose that our computer can only store 2 significant digits and it is asked to do Gaussian elimination on

$$\left(\begin{array}{cc|c} .001 & 1 & 3 \\ 1 & 2 & 5 \end{array}\right).$$

Doing the elimination exactly would produce

$$\left(\begin{array}{cc|c} .001 & 1 & 3 \\ 0 & -998 & -2995 \end{array}\right),$$

but rounding to 2 digits, our computer would store this as

$$\left(\begin{array}{cc|c} .001 & 1 & 3 \\ 0 & -1000 & -3000 \end{array}\right).$$

Backsolving this reduced system gives

$$x_1 = 0 \quad \text{and} \quad x_2 = 3.$$

This seems fine until you realize that backsolving the unrounded system gives

$$x_1 = -1 \quad \text{and} \quad x_2 = 3.001.$$

Row Pivoting

A way to fix the problem is to use pivoting, which means to switch rows of the matrix. Since switching rows of the augmented matrix just corresponds to switching the order of the equations, no harm is done:

$$\left(\begin{array}{cc|c} 1 & 2 & 5 \\ .001 & 1 & 3 \end{array}\right).$$

Exact elimination would produce

$$\left(\begin{array}{cc|c} 1 & 2 & 5 \\ 0 & .998 & 2.995 \end{array}\right).$$

Storing this result with only 2 significant digits gives

$$\left(\begin{array}{cc|c} 1 & 2 & 5 \\ 0 & 1 & 3 \end{array}\right).$$

Now backsolving produces

$$x_1 = -1 \quad \text{and} \quad x_2 = 3,$$

which is the true solution (rounded to 2 significant digits).

The reason this worked is because 1 is bigger than 0.001. To pivot we switch rows so that the largest entry in a column is the one used to eliminate the others. In bigger matrices, after each column is completed, compare the diagonal element of the next column with all the entries below it. Switch it (and the entire row) with the one with greatest absolute value. For example in the following matrix, the first column is finished and before doing the second column, pivoting should occur since $|-2| > |1|$:

$$\left(\begin{array}{ccc|c} 1 & -2 & 3 & 4 \\ 0 & 1 & 6 & 7 \\ 0 & -2 & -10 & -10 \end{array}\right).$$

Pivoting the 2nd and 3rd rows would produce

$$\left(\begin{array}{ccc|c} 1 & -2 & 3 & 4 \\ 0 & -2 & -10 & -10 \\ 0 & 1 & 6 & 7 \end{array}\right).$$

Condition number

In some systems, problems occur even without rounding. Consider the following augmented matrices:

$$\left(\begin{array}{cc|c} 1 & 1/2 & 3/2 \\ 1/2 & 1/3 & 1 \end{array}\right) \quad \text{and} \quad \left(\begin{array}{cc|c} 1 & 1/2 & 3/2 \\ 1/2 & 1/3 & 5/6 \end{array}\right).$$

Here we have the same A, but two different input vectors:

$$\mathbf{b}_1 = (3/2, 1)' \quad \text{and} \quad \mathbf{b}_2 = (3/2, 5/6)'$$

which are pretty close to one another. You would expect then that the solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ would also be close. Notice that this matrix does not need pivoting. Eliminating exactly we get

$$\left(\begin{array}{cc|c} 1 & 1/2 & 3/2 \\ 0 & 1/12 & 1/4 \end{array}\right) \quad \text{and} \quad \left(\begin{array}{cc|c} 1 & 1/2 & 3/2 \\ 0 & 1/12 & 1/12 \end{array}\right).$$

Now solving we find

$$\mathbf{x}_1 = (0, 3)' \quad \text{and} \quad \mathbf{x}_2 = (1, 1)'$$

which are *not close at all* despite the fact that we did the calculations exactly. This poses a new problem: some matrices are very sensitive to small changes in input data. The extent of this sensitivity is measured

by the **condition number**. The definition of condition number is: consider all small changes δA and $\delta \mathbf{b}$ in A and $\mathbf{b}$ and the resulting change, $\delta \mathbf{x}$, in the solution $\mathbf{x}$. Then

$$\text{cond}(A) \equiv \max \left(\frac{\|\delta \mathbf{x}\|/\|\mathbf{x}\|}{\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|}} \right) = \max \left(\frac{\text{Relative error of output}}{\text{Relative error of inputs}} \right).$$

Put another way, changes in the input data get multiplied by the condition number to produce changes in the outputs. Thus a high condition number is bad. It implies that small errors in the input can cause large errors in the output. It is not obvious from our definition above, but one can prove that the condition number of a matrix is at least 1.

In MATLAB enter

```
≫ H = hilb(2)
```

which should result in the matrix above. MATLAB produces the condition number of a matrix with the command

```
≫ cond(H)
```

Thus for this matrix small errors in the input can get magnified by 19 in the output. Next try the matrix

```
≫ A = [ 1.2969 0.8648 ; .2161  .1441]
≫ cond(A)
```

For this matrix small errors in the input can get magnified by 2.5×10^8 in the output! (We will see this happen in the exercise.) This is obviously not very good for engineering where all measurements, constants and inputs are approximate.

Is there a solution to the problem of bad condition numbers? Usually, bad condition numbers in engineering contexts result from poor design. So, the engineering solution to bad conditioning is **redesign**.

Finally, find the determinant of the matrix A above:

```
≫ det(A)
```

which will be small. If $\det(A) = 0$ then the matrix is singular, which is bad because it implies there will not be a unique solution. The case here, $\det(A) \approx 0$, is also bad, because it means the matrix is almost singular. Although $\det(A) \approx 0$ generally indicates that the condition number will be large, they are actually independent things. To see this, find the determinant and condition number of the matrix `[1e-10,0;0,1e-10]` and the matrix `[1e+10,0;0,1e-10]`.

Exercises

11.1 Let

$$A = \begin{bmatrix} 1.2969 & .8648 \\ .2161 & .1441 \end{bmatrix}.$$

(a) Find the condition number, determinant and inverse of A (using MATLAB).

(b) Let B be the matrix obtained from A by rounding off to three decimal places ($1.2969 \mapsto 1.297$). Find the inverse of B. How do A^{-1} and B^{-1} differ? Explain how this happened.

(c) Set `b1 = [1.2969; 0.2161]` and do `x = A \ b1` . Repeat the process but with a vector `b2` obtained from `b1` by rounding off to three decimal places. Explain exactly what happened. Why was the first answer so simple? Why do the two answers differ by so much?

11.2 To see how to solve linear systems symbolically, try

```
≫ B = [sin(sym(1)) sin(sym(2)); sin(sym(3)) sin(sym(4))]
≫ c = [1; 2]
≫ x = B \ c
≫ pretty(x)
```

Now input the matrix

$$\mathsf{Cs} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

symbolically as above by wrapping each number in `sym`. Create a numerical version via `Cn = double(Cs)` and define the two vectors `d1 = [4; 8]` and `d2 = [1; 1]`. Solve the systems (as in the third line of code above) `Cs*x = d1`, `Cn*x = d1`, `Cs*x = d2`, and `Cn*x = d2`. Explain the results. Does the symbolic or non-symbolic way give more information?