Error in Gaussian Elimination

Fox, in 1971, published a paper called "How to Get Meaningless Answers in Scientific Computation" whose section titles outline the common ways you might get worthless numerical answers:

1. Your problem might be ill-conditioned.
2. Your method might be unstable.
3. You expect too much "analysis" from the computer, such as using an inappropriate convergence stopping criteria for an iterative method.
4. You accept consistency too easily.
5. A successful method may fail in slightly different circumstances.
6. Your test examples may be too special.

In this handout, we will define and explore the first two items. Our problem is: Given a real square matrix A and a right-hand-side $\mathbf{b}$, find $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}$. One of the big ideas in numerical analysis is that the problem itself, regardless of the method/algorithm applied, limits how accurate your solution can be, and this is called the conditioning of the problem. Second, we look at the Gaussian Elimination without Pivoting Algorithm (the LU factorization) and its effects on solution accuracy.

For both goals, we need definitions of forward and backward error for the problem $A\mathbf{x} = \mathbf{b}$. For an approximate answer $\mathbf{x}_a$, we call the difference $\mathbf{b} - A\mathbf{x}_a$ the *residual.*

Example 1. Compute the residual for $\mathbf{x}_a = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $A = \begin{bmatrix} 1 & 1 \\ 1.1 & 1 \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$. Is the residual a good measure of error? Forward or backward?

Since $A\mathbf{x} = \begin{bmatrix} 2 \\ 2.1 \end{bmatrix}$, the residual is $\begin{bmatrix} 0 \\ -0.1 \end{bmatrix}$. We definitely use the residual as part of the error analysis, and as the difference between outputs it's closer to the backward error. However, as a vector instead of a number, it isn't really enough, i.e., would this residual be better or worse than a residual of $\begin{bmatrix} 0.05 \\ -0.05 \end{bmatrix}$?

In order to define the forward and backward error, we transform this residual to a length using vector norms.

Definition 1. A *vector norm* $||\mathbf{x}||$ is any way of assigning length to a vector with the following properties.

1. $||\mathbf{x}|| \geq 0$ with equality if and only if $\mathbf{x} = \mathbf{0}$.
2. for each scalar c and vector $\mathbf{x}$, $||c\mathbf{x}|| = |c|||\mathbf{x}||$.
3. for vectors $\mathbf{x}, \mathbf{y}$, $||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$.

We also will need norms for matrices, which have a very similar set of required properties:

Definition 2. A *matrix norm* $||A||$ is any way of assigning length to a matrix with the following properties:

1. $||A|| \geq 0$ with equality if and only if $A = 0$.
2. for each scalar c and matrix A, $||cA|| = |c|||A||$.
3. for compatible matrices A, B, $||A + B|| \leq ||A|| + ||B||$.

When it comes to matrix norms, we are interested only in a few *operator* or *induced norms*, that is, norms that can be defined as

$$||A|| = \max \frac{||A\mathbf{x}||}{||\mathbf{x}||}$$

where the maximum is being found over all nonzero vectors $\mathbf{x}$. Such norms are *submultiplicative*, meaning they satisfy a fourth property:

$$||AB|| \leq ||A|| \cdot ||B||.$$

There are infinitely many matrix and vector norms satisfying these properties. Among the most common vector norms are:

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

  Add the absolute values of all elements in the vector.

- $||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$

  Add the squares of all elements in the vector, then square root the total.

- $||\mathbf{x}||_\infty = \max_{1 \leq i \leq n} |x_i|$

  Find the element whose magnitude is the largest.

While you're probably most familiar with the 2-norm (it's the distance formula between two points, for example), the corresponding matrix norm (called the Frobenius norm) is not an induced norm. So, we will only be using the 1-norm and ∞-norm. A proof that the ∞-norm is a vector norm is provided at the end of this handout; proving that the 1-norm is a vector norm is in your homework.

Example 2. Find the 1-norm and ∞-norm of the vectors $\mathbf{r}_1 = \begin{bmatrix} 0 \\ -0.1 \end{bmatrix}$ and $\mathbf{r}_2 = \begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix}$.

$||\mathbf{r}_1||_1 = 0.1$ and $||\mathbf{r}_2||_1 = 0.1$.

$||\mathbf{r}_1||_\infty = 0.1$ and $||\mathbf{r}_2||_\infty = 0.05$.

The corresponding matrix norms are not the largest element nor the total of all the elements, as you might expect. Instead,

- $||A||_1 = \max\limits_{1\le j\le n} ||\mathbf{a}_j||_1 = \max\limits_{1\le j\le n} (\sum\limits_{i=1}^{n} |x_{ij}|)$ where $\mathbf{a}_j$ is the j^{th} column of A.

  $||A||_1$ is the maximum column sum of A.

- $||A||_\infty = \max\limits_{1\le i\le n} ||\mathbf{a}_i^*||_1 = \max\limits_{1\le i\le n} (\sum\limits_{j=1}^{n} |x_{ij}|)$ where $\mathbf{a}_i^*$ is the i^{th} row of A.

  $||A||_\infty$ is the maximum row sum of A.

Example 3. Compute the 1-norm and ∞-norm of $A = \begin{bmatrix} 1 & -5 \\ 1.1 & 2 \end{bmatrix}$.

$||A||_1 = \max\{2.1, 7\} = 7$.

$||A||_\infty = \max\{6, 3.1\} = 6$.

Conditioning

We can now define the forward and backward error for solving a square system of equations.

Definition 3. The backward error is $||\mathbf{b} - A\mathbf{x}|| = ||\mathbf{r}||$ and the forward error is $||\mathbf{x} - \mathbf{x}_a||$. The relative backward error is $\frac{||\mathbf{b}-A\mathbf{x}||}{||\mathbf{b}||}$ and the relative forward error is $\frac{||\mathbf{x}-\mathbf{x}_a||}{||\mathbf{x}||}$.

When we talk about the condition of a problem, we're primarily looking at the relationship between the forward error and the backward error. Specifically, we expect this inequality to hold,

$$\text{forward error} \lesssim \text{condition number} \cdot \text{backward error},$$

where the "condition number" is something specific to the type of problem completing this relationship. Recalling the example from the last homework, I asked for a polynomial with a root where it was possible to get an approximation with a backward error of 10^{-6}, but a forward error of 1. That's because the problem itself (root finding) is ill-conditioned (meaning prone to large condition numbers). Any polynomial with a root that has a large multiplicity may have this issue.

Definition 4. The *error magnification factor* is the ratio $\dfrac{\text{relative forward error}}{\text{relative backward error}}$. The *condition number* of a problem is the largest possible error magnification factor.

Definition 5. The *condition number of A* is the largest possible error magnification factor for all right-hand-sides $\mathbf{b}$, and is given by the formula

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

Example 4. Compute the condition number of $A = \begin{bmatrix} 1 & 1 \\ 1.1 & 1 \end{bmatrix}$, whose inverse is $A^{-1} = \begin{bmatrix} -10 & 10 \\ 11 & -10 \end{bmatrix}$, in the ∞-norm.

For the ∞-norm, $||A||_\infty$ is the maximum row sum, so it is 2.1. Then, the maximum row sum of A^{-1} is 21. Thus $\kappa(A) = 2.1(21) = 44.1$. For this specific problem, the condition number is exactly the same in the 1-norm. This is not typically the case that norms give exactly the same number, but if a matrix has a large norm in a particular norm, it should have a large norm in other norms where "large" and "small" may depend on the norm. For the 1-norm and ∞-norm, matrices whose condition number is in the tens or hundreds won't really encounter significant numerical issues for solving $A\mathbf{x} = \mathbf{b}$. However, once the condition number starts getting into the thousands, numerical issues may arise, and here's why.

In practical terms, the condition number of the problem $A\mathbf{x} = \mathbf{b}$ tells us how many digits of accuracy we should expect to lose in computing $\mathbf{x}$. That is, the error in the input of A and $\mathbf{b}$ in double precision should be of order ϵ_{mach} or roughly 10^{-16}. If $\kappa(A) \approx 10^k$, then we can only get $10^k \epsilon_{mach}$ digits of accuracy in $\mathbf{x}_a$. For the A of Example 4, we have a pretty small condition number, roughly 4×10^1, and can still expect error on the order of 10^{-15} depending on our solution method. However, if we now look at $A = \begin{bmatrix} 1 & 1 \\ 1.0001 & 1 \end{bmatrix}$, whose inverse is $\begin{bmatrix} -10000 & 10000 \\ 10001 & -10000 \end{bmatrix}$, the condition number is $\kappa(A) = 40004.0001 \approx 4 \times 10^4$ and the best we can do is 12 digits of accuracy.

Question 6. What happens if you try to solve a system with a condition number of 10^{17}? Do you think such a matrix exists?

You cannot solve such a system in double precision. And yes, such matrices do exist, though I wouldn't call them particularly common. The notorious example is the Hilbert matrix H whose entries are given by $H_{ij} = \frac{1}{i+j-1}$. The 2×2 Hilbert matrix is

$$\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix}.$$

The condition number of a Hilbert matrix exceeds 10^{16} as small as 11×11.

Gaussian Elimination

Once you've determined the problem is sufficiently well-conditioned for solving, next we turn to an algorithm to produce a solution. In this case, we're examining Gaussian Elimination without Pivoting, sometimes called Classical Gaussian Elimination. What we want is a *stable* algorithm. While a technical definitions and proofs of stability go beyond the scope of the course, we can understand stability in the following sense. The following comes from *Numerical Linear Algebra* by Trefethen & Bau.

> A stable algorithm gives nearly the right answer
> to nearly the right question.

In other words, both the relative forward and the relative backward error are small, preferably on order ϵ_{mach} though that's not always realistic. An even better algorithm is backward stable.

> A backward stable algorithm gives exactly the right answer
> to nearly the right question.

In order to be stable, an algorithm must first always produce a solution.

Question 7. Does every matrix have an LU factorization?

No. Take any matrix whose first entry is 0 that also has a nonzero entry in the first column, and that matrix does not have an LU factorization. One such example is $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$. So Classical Gaussian Elimination must not be stable in the rigorous sense. What about for all the matrices for which it does produce a solution?

Question 8. What happens when you compute the LU factorization of $\begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}$ in double precision? Compute the relative forward and backward errors for $\mathbf{b} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$.

$\begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{bmatrix}$, which in double precision will round to $U = \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix}$ with $L = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}$. Note that $LU = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix}$, so we do not regain A from the product. This numerical effect is sometimes called *swamping* - the multiplier is so much larger than the entries that it swamps them and they disappear from the computation.

The exact solution to $A\mathbf{x} = \mathbf{b}$ is pretty close to $\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ so we'll call that the true solution. Using LU with back substitution, we get $\begin{bmatrix} 1 & 0 & 1 \\ 10^{20} & 1 & 3 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -10^{20} \end{bmatrix}$, then

$\begin{bmatrix}10^{-20} & 1 & 1\\ 0 & -10^{20} & -10^{20}\end{bmatrix} \sim \begin{bmatrix}10^{-20} & 1 & 1\\ 0 & 1 & 1\end{bmatrix} \sim \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 1\end{bmatrix}$ so the computed solution is $\begin{bmatrix}0\\1\end{bmatrix}$ instead of $\begin{bmatrix}2\\1\end{bmatrix}$.

Absolute forward error: $||\mathbf{x} - \mathbf{x}_a||_\infty = || \begin{bmatrix}2\\0\end{bmatrix} ||_\infty = 2$

Relative forward error: $\frac{||\mathbf{x}-\mathbf{x}_a||_\infty}{||\mathbf{x}||_\infty} = \frac{2}{2} = 1$ (definitely not near 10^{-16}.)

Absolute backward error: $||\mathbf{b} - A\mathbf{x}_a||_\infty = || \begin{bmatrix}1\\3\end{bmatrix} - \begin{bmatrix}1\\1\end{bmatrix} ||_\infty = 2$

Relative backward error: $\frac{||\mathbf{b}-A\mathbf{x}_a||_\infty}{||\mathbf{b}||_\infty} = \frac{2}{3}$ (also not near 10^{-16}.)

Thankfully, reintroducing row swaps will help address both of these issues!

Proof Addendum

Example 5. A proof that the ∞-norm is a vector norm.

Proof. Let $\mathbf{x}$ be a vector of size $n \times 1$.

Property 1. First note that $||\mathbf{x}||_\infty \geq 0$ due to the nonnegativity of the absolute value. It remains to show that $||\mathbf{x}||_\infty = 0$ if and only if $\mathbf{x} = \mathbf{0}$.

($\Rightarrow$) We proceed by contrapositive, that is, if $\mathbf{x} \neq \mathbf{0}$ then $||\mathbf{x}||_\infty \neq 0$. So assume $\mathbf{x} \neq \mathbf{0}$. Then at least one element x_i of $\mathbf{x}$ satisfies $|x_i| > 0$, and since $||\mathbf{x}||_\infty \geq |x_i|$, $||\mathbf{x}||_\infty \neq 0$.

($\Leftarrow$) Assume $\mathbf{x} = \mathbf{0}$. Then for all elements $1 \leq i \leq n$, $|x_i| = 0$ so $||\mathbf{x}||_\infty = 0$.

Property 2. Let c be any scalar. Then $||c\mathbf{x}||_\infty = \max\limits_{1\leq i\leq n} |cx_i| = \max\limits_{1\leq i\leq n} |c||x_i| = |c| \max\limits_{1\leq i\leq n} |x_i| = |c|||\mathbf{x}||_\infty$ as desired.

Property 3. Let $\mathbf{y}$ be another vector of size $n \times 1$. Then

$$\begin{aligned}
||\mathbf{x}+\mathbf{y}||_\infty &= \max_{1\leq i\leq n} |x_i + y_i| \\
&\leq \max_{1\leq i\leq n} (|x_i| + |y_i|) \text{ by the triangle inequality of real numbers} \\
&\leq \max_{1\leq i\leq n} |x_i| + \max_{1\leq j\leq n} |y_j| \\
&= ||\mathbf{x}||_\infty + ||\mathbf{y}||_\infty.
\end{aligned}$$

Thus $|| \cdot ||_\infty$ is a vector norm. □