1 Lecture 1

Rounding, Approximation, Well Posedness

Definition 1.1. We say that a problem is well posed if its solution exists, is unique and depends continuously on the inputs.

Definition 1.2. Rounding error refers to our representation of real numbers as floating point numbers.

Definition 1.3. Truncation error refers to our representation of infinite collections as finite ones. For example, the truncation of a Taylor series to a finite quantity of terms is an example of truncation.

Definition 1.4. Approximating a solution can be divided into two stages: approximation before the calculation and approximation during the calculation. Approximation before the calculation includes the following:

- How we model the problem.
- The accuracy of our input data.
  - How we measure the input data
  - How we compute the input data

Approximation during the calculation includes:

- Rounding Error
- Truncation Error

Remark 1.5. Recall that norms allow for constant pullout, satisfy the triangle inequality and output 0 if and only the input to the norm is also 0.

Example 1.6. It is easy to undestand rounding and truncation error. The first set of approximations can be understood as follows:

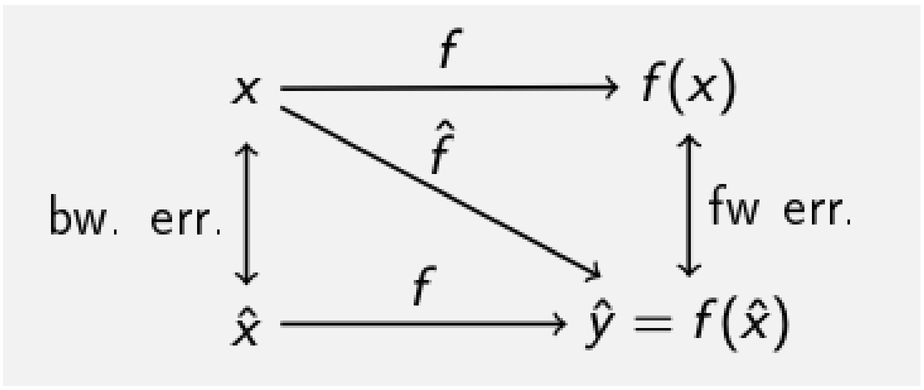
Suppose we use the surface area of the earth as an input. To calculate the surface area, we first assume that the earth is a sphere, then we somehow measure the radius and then we compute that area by multiplying the radius squared by π.

2 Lecture 2

Rounding and Error

Definition 2.1. Let x be some input, f some true function and $\hat{f}$ some estimated function. Absolute forward error is the difference $\left|f(x) - \hat{f}(x)\right|$. Relative forward error is $\frac{\left|f(x)-\hat{f}(x)\right|}{|f(x)|}$.

Definition 2.2. Backward error is the difference between the original input and the input that, under the true function f, would have given you $\hat{f}(x)$, the output of the original input under the approximate function.



Remark 2.3. Suppose I present to you two problems p_1 and p_2 that have the same forward error but the backward error of p_1, $b(p_1)$ is such that $b(p_1) > b(p_2)$. Is problem 1 or 2 more favorable? Is any more favorable? We can generally say that p_1 is more favorable, because it takes a larger input perturbation in p_1 to produce, under a true function f, the same output perturbation as p_2. This would suggest that p_1 is more *well conditioned.*

Remark 2.4. Truncation and rounding error make up the ways in which an operation is approximate. By truncation error, we refer to the process of replacing infinite sums and expressions with finite ones and by rounding error, we refer to the error introduced in representing real numbers with computer based number systems.

Lemma 2.5. This is sort of a lemma:

The relative error in the forward error tells us how many digits are significant in our estimate of the solution to a problem.

Example 2.6. Suppose we estimate $\sqrt{2} = 1.414\ldots$ as 1.4. Then the negative log of the relative error is $- \approx \log(0.01) = 2$, meaning that (correctly) our estimate is accurate up to 2 digits.

Definition 2.7. The condition number is

$$\sup_{x \in \text{domain}} \frac{\text{Relative Forward Error}}{\text{Relative backward error}}$$

Theorem 2.8. Assuming that a true function f is differentiable, then its condition number is given by

$$\sup_{x \in \text{Domain}} \frac{x f'(x)}{f(x)}$$

Proof.

$$\begin{aligned}\Delta y = \frac{f(x+\Delta x) - f(x)}{1} &= f'(x)\Delta(x)\\ \implies \frac{\Delta y}{y} \Big/ \frac{\Delta x}{x}\\ = \frac{f'(x)\Delta x}{y} \Big/ \frac{\Delta x}{x}\end{aligned}$$

and the latter simplies to the claimed expression. □

Problem Forward Error. Approximate $f(x) = 1/x$ by the taylor expansion $1 - (x-1)$. What is the forward error when we let $x = 0.5$?

It is 0.5

Problem Backward Error. What about the backward error if we use $x = 0.5$

We need to find the value x' such that $f(x') = 1.5$. That is $x' = \frac{2}{3}$. Therefore, $|x - x'| = 0.16$.

Problem Condition Number. Determine the condition number of $\sin(x)$ on the interval $[0, \pi/2]$.

Proof. The formula gives us that $k(f) = \sup_{x \in [0,\pi/2]} \frac{x \cos x}{\sin x}$. Note that this is a decreasing function on. the given interval, so that the largest value is taken on at $x = 0$. By L'Hopital's rule, that gives us:

$$\frac{\cos x - x \sin x}{\cos x}.$$

which evaluates to 1 at $x = 0$. Thus, the condition number is 1. □

Problem Larger Condition Number. Over which interval is the condition number larger: $[0, 2\pi]$ or $[10^4\pi, 10^4\pi + 2\pi]$.

Proof. Recall that the formula is $\frac{xf'(x)}{f(x)}$, and that $f'(x)$ and $f(x)$ will assume the same values up to equivalence classes of $\sin$. Thus, the x in the numerator accounts for any potential difference, making the second interval have a greater condition number. □

Definition 2.9. A problem is well conditioned if, when considering a problem, small perturbations in an input do not result in large perturbations in the problem's output.

Definition 2.10. A method is stable if, when using it to solve a problem, the solution produced by the method is an exact solution to a nearby problem (ie the same problem applied to a different input). In other words, the error introduced by the method is no worse than the introduction of a little error into the input data.

Remark 2.11. We see that conditioning describes a problem, whereas stability describes a method used to solve that problem.

Definition 2.12. A method is accurate if it produces solutions that are very close to the true answer. A solution is accurate if both the method is stable and the problem well conditioned. If a problem has a stable algorithm but is not itself well conditioned, then the solution may not necessarily be accurate.

3 Lecture 3: Floating Point

The following questions were prelecture questions:

Problem Operator Conditioning. Let $x > 1$. Which of $f(x) = 3x, f(x) = 3/x, f(x) = 3 - x, f(x) = \sqrt{x}$ is ill posed (here, make the restrictive definition that a well posed problem must have a finite condition number.?

Proof. We can compute the condition numbers using the formula $xf'(x)/f(x)$ since all functions are differentiable over the domain $x > 1$. This gives:

- $$\frac{3x}{3x} = 1$$
- $$\frac{x(-3/x^2)}{3/x} = \frac{(-3/x)}{3/x} = -1$$
- $$\frac{x(-1)}{3-x} \to \infty \text{ as } x \to 3$$
- $$\frac{-1/2}{\sqrt{x}}$$

□

Problem Forward and Backward Errors. Approximate $\sqrt{1+x}$ by $1+x/2$. What is the absolute backward error using $x = 8$.

Proof. To get $1 + 8/2 = 5$ using f, we need to let $x = 24$. Hence, the absolute backward error is $|8 - 24| = 16$. □

Problem Why choose a stable algorithm?. A stable algorithm always:

- produces an accurate result
  - False. If a problem is additionally well conditioned, then a stable algorithm applied to it may result in an accurate solution; otherwise, no
- insensitive to data error
  - No, a stable algorithm only assures us that an answer is a solution to a nearby problem
- improves the conditioning of the problem
  - No, this is distinct from stability. See the first retort.

- gives the correct answer to a nearby problem
    - This is exactly the definition.

Problem Condition Number of Square Root. At $x = 7$?

Proof.

$$\frac{1/2}{\sqrt{x}}x \Big/ \sqrt{x} = 1/2$$

□

Problem Backward Error. Given $x = 0.5$, approximate $\cos x$ by $1 - \frac{x^2}{2}$. What is the backward error?

Proof. Find $\hat{x}$ such that $\cos \hat{x} = 0.875$. Subtract that from 0.5. □

Problem Estimate Condition Number. Estimate $f(x) = \log(1 + x)$ (remember that log means natural log) with $x - \frac{x^2}{2} + \frac{x^3}{3}$. Suppose we know that $f(0.5168967) = \hat{f}(0.5)$. Then we know that the relative backward error is $\frac{0.516897 - 0.5}{0.5}$. Now use this information to approximate the condition number of f.

Proof. We already know the relative backward error, and the condition number is relative forward error over relative backward error. Hence, the condition number estimate is

$$\frac{\hat{f}(0.5) - f(0.5)}{f(0.5)} \Big/ \text{Relative Backward Error}$$

□

Definition 3.1. Suppose that we represent numbers using 64 bits. 32 bits will be dedicated to the whole portion of any number − we will progress under this system, going from rightmost bit to left most bit, from 2^0 to 2^32; similarly, 32 bits will be dedicated to the fractionalpart of any number − we will progress from rightmost bit to leftmost bit from 2^-32 to 2^-1.

Remark 3.2. Suppose that we wish to represent 2^{32}. Observe that largest number representable using this scheme is $2^{32} - 2^{-32}$.

It follows that the relative error of our represntation of 2^{32} is

$$\frac{2^{-32}}{2^{32}} = 2^{-64}$$

Now suppose that we wish to represent some number x in the range $(2^{-32}, 2^{-31})$. Assuming that 2^{-32} is closer to this number then 2^{-32}, the relative error of this representation is

$$\frac{|x - 2^{-32}|}{|x|} \leq \frac{|2^{-33}|}{|x|} \leq \frac{|2^{-33}|}{2^{-32}} = 2^{-1}$$

Observe that whereas the relative error in the former is good, in the latter it is poor. This motivates our definition of the double precision floating point system.

Example 3.3. Represent 13 in scientific notation in a binary number system. Require that the first and only digit before the decimal point (remember that this is scientific notation) is a 1; further require that 1 must occupy this position. You will find that the representation is uniquely $(1.101)_2 * 2^3$.

Remark 3.4. Suppose I told you that we we will represent all numbers as we did above. Leaving aside how we would represent 0, observe that since we require that the first and only digit before the decimal point is a 1, if we want to store such representations, there is no point in storing the 1. This allows us to say the following:

Definition 3.5.

- The double precision floating point system allocates 52 bits to store the fractional part of a number represented using the scheme above, where a 1 in the units place implicitly precedes the fractional part.
- 11 bits are allocated to store an exponent and 1 bit is allocated to store a sign bit.
  - The 11 exponent bits allow us to get integers in the range $[0, 2047]$. Assuming that we begin with an offest of -1023, meaning that 0 maps to -1023, we can represent an exponent in the range $[-1023, 1024]$.
- When the exponent becomes 0, instead of mapping to -1023, we to -1022 (yes, this contradicts the previous line, but introducting the contradiction and then correcting it is pedagogically more gradual). We also do away with the implicit 1 that precedes the fractional part of the significand (recall tha the fractional part of the significand is the only part which is stored in our representation). 0 is represented if we let all bits of the fractional part be 0.
  - Since we are no longer required to have an implicit, leading 1, notice that we can now drop down past $1 * 2^{-1022}$. Represent 2^{-1023} as $0.100\cdots \times 2^{-1022}$; represent 2^{-1024} as $0.0100\cdots \times 2^{-1022}$ and so forth. UNRESOLVED
- We map 2^{1024} to ∞ and if the fractional part contains any non-zero digit, we call that values ∞ instead. UNRESOLVED
- Define the UFL and OFL to be the smallest and greatest normal numbers that are not subnormal nor ∞.

Definition 3.6. Suppose that we have a number $x = b.\quad \underbrace{b_1 b_2 \ldots b_{52}}_{\text{all useable digits consumed}} \quad d_1 d_2 d_3.$ We need to somehow represent x in our floating point system. To do so, we round on the basis of $d_1d_2d_3$. Now we could make the scheme round to nearest, meaning that add 1 to b_{52} if $d_1d_2d_3 > 100$ and do nothing if $d_1d_2d_3 < 100$, but we must then choose what to do when $d_1d_2d_3 = 100$. What then? If we make the consistent, arbitrary decision to round up, we will produce statistical noise. Thus, the scheme employed is to round to even:

If b_{52} is 0, then do nothing, so that x is even; if b_{52} is 1, then add 1 to b_{52}.

Remark 3.7. Some floating point exercises `https://relate.cs.illinois.edu/course/cs450-s19/flow/inclass-floating-point/start/`

Definition 3.8. Let ϵ be the smallest number such that $\mathrm{fl}(x(1+\epsilon)) > x$. If we use round to even, it is 2^{-52}; if we use round to nearest, with rounding up in case of a tie, it is 2^{-53} – in which case we also distinguish 2^{-52} with some title.

Lemma 3.9. The relative error in representing any number x within the normal range is at most ϵ.

Proof. Given x, there is an a such that

$$2^a \leq x \leq 2^{a+1}$$

Since we can represent any number $y = 2^a + \epsilon * n$ up to $y = 2^{a+1}$, it follows that there is at least one representable number in the range $[x, x+\epsilon]$. This allows us to say that the absolute error between this number and x is at most ϵ, which is formalized below:

$$\frac{|x-(1+\epsilon)x|}{|x|} = \frac{|x|\epsilon}{|x|} = \epsilon$$

$\square$

4 Lecture 4

Example 4.1. To perform floating point addition, suppose that you have $b_0.b_1 \dots b_n \times 2^x$ and $d_0.d_1 \dots d_n \times 2^y$ where $x > y$; make sure that you multiply the second number by 2^{x-y} and then perform grade school addition.

Definition 4.2. Catastrophic cancellation takes place when the number of accurate digits in the result of an operation are far less than the number of accurate digits in the members of that operation.

Example 4.3. Supposing that we opereate in double precision and that a and b both have 12 digits of accuracy. Suppose that a and b agree up to their first 10 digits.

When a and b are subtracted, the first 10 digits will be cancelled and the final 2 digits will shift leftwards, becoming the 2, sole accurate digits. The remaining digits will be filled with garbage. From this, we see that whereas we had 12 digits of accuracy, we now only have 2 (ie our relative error is 10^-2).

Definition 4.4. Given a vector norm $\|\cdot\|$, we define the matrix norm to be

$$\sup_{x\in\mathcal{D}, x\neq 0} \frac{\|Ax\|}{\|x\|}$$

This is equivalent to

$$\sup_{x\in\mathcal{D}, x\neq 0} \frac{\|Ax\|}{\|x\|}$$
$$\sup_{x\in\mathcal{D}, x\neq 0} \frac{\left\|A\frac{x}{\|x\|}\right\|}{1}$$
$$\sup_{x\in\mathcal{D}, \|x\|=1} \frac{\|Ax\|}{1}$$

We denote this norm by $\|A\|$.

Proposition 4.5.

$$\|A\|\|x\| \geq \|Ax\|$$

for all x that A can multiply.

Proof. Follows by definition. □

Proposition 4.6.

$$\|A\|_1 = \max_j \sum_i |A_{i,j}|$$

$$\|A\|_\infty = \max_i \sum_j |A_{i,j}|$$

An easy way to remember the 1 norm and ∞ norm is to note that these matrix norms also coincide for a vector, for which the 1 norm is the sum of absolute values, and the ∞ norm is the maximum value.

Definition 4.7. Suppose that A is invertible and that we wish to find the condition number of the problem of solving $Ax = b$ where b is the input and x the ouput; then the condition number is given by

$$\frac{\|\Delta x\|}{\|x\|} \bigg/ \frac{\|\Delta b\|}{\|b\|}$$

$$\frac{\|\Delta x\|}{\|x\|} \bigg/ \frac{\|\Delta b\|}{\|b\|}$$

Let $x = A^{-1}b$ and $\Delta x = A^{-1}\Delta b$

$$\frac{\|A^{-1}\Delta b\|}{\|A^{-1}b\|} \bigg/ \frac{\|\Delta b\|}{\|b\|}$$

Now apply submultiplicativity

$$\leq \|A\|\|A^{-1}\|$$

This inequality can be shown to be sharp for every matrix: for every matrix, there is a vector b for which this bound is attained, and so the condition number of the problem is $\|A\|\|A^{-1}\|$

We also call this quantity the condition number of the matrix.

Definition 4.8. If a matrix is not invertible, then we say that $k(A) = \infty$.

Proposition 4.9. $k(A) = k(A^{-1})$ and matrix vector multiplication has the same conditioning as solving a system of linear equations.

Proof. The first is easy. To find the y such that $Ax = y$, take x as the input of $A^{-1}y = x$. $\square$

Proposition 4.10.

- The following properties hold:
  - $\|AB\| \leq \|A\|\|B\|$
    - For $\|ABx\| \leq \|A\|\|Bx\| \leq \|A\|\|B\|\|x\|$. The previous was true for all x; in particular, for all $x \neq 0$, this implies that
    $$\|AB\| = \frac{\|ABx\|}{\|x\|} \leq \|A\|\|B\|$$
  - $\|A + B\| \leq \|A\| + \|B\|$.
  - $\left\|AA^{-1}\right\| = 1 \leq \|A\|\left\|A^{-1}\right\| \implies 1 \leq k(A)$
- $k(\gamma A) = \gamma k(A)$.
- If D is a diagonal matrix, then $k(D) = \frac{\max d_{i,i}}{\min_{d_{i,i}}}$

Remark 4.11. A computational technique to evaluating a condition number is multiply a matrix A by many vectors that have unit norm and take the maximum such norm. A visualization of the resultant vectors also shows (UNRESOLVED page 55).

The following is the definition of the residual.

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}.$$

Remark 4.12. The residual itself does not reveal much. Suppose we calculate $r = b - Ax$. Now solve for $kAx = kb$ and the residual required to solve that is k times as great. This is why we define the relative residual:

$$\frac{\|r\|}{\|A\| \cdot \|\hat{x}\|}$$

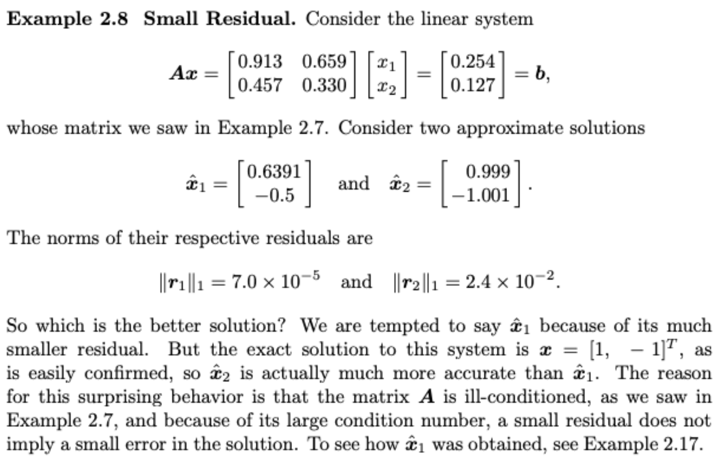
We can obtain a bound on the relative forward error required to solve $Ax = b$ in terms of r.

$$\|\Delta \boldsymbol{x}\| = \|\hat{\boldsymbol{x}} - \boldsymbol{x}\| = \|\boldsymbol{A}^{-1}(\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{b})\| = \|-\boldsymbol{A}^{-1}\boldsymbol{r}\| \leq \|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{r}\|.$$

Dividing both sides by $\|\hat{\boldsymbol{x}}\|$ and using the definition of $\operatorname{cond}(\boldsymbol{A})$, we then have

$$\frac{\|\Delta \boldsymbol{x}\|}{\|\hat{\boldsymbol{x}}\|} \leq \operatorname{cond}(\boldsymbol{A}) \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{A}\| \cdot \|\hat{\boldsymbol{x}}\|}.$$

Remark 4.13. This bound tells us that if the residual is small and the matrix and well conditioned, then the relative error is low.



Example 2.8 Small Residual. Consider the linear system

$$\boldsymbol{Ax} = \begin{bmatrix} 0.913 & 0.659 \\ 0.457 & 0.330 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.254 \\ 0.127 \end{bmatrix} = \boldsymbol{b},$$

whose matrix we saw in Example 2.7. Consider two approximate solutions

$$\hat{\boldsymbol{x}}_1 = \begin{bmatrix} 0.6391 \\ -0.5 \end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{x}}_2 = \begin{bmatrix} 0.999 \\ -1.001 \end{bmatrix}.$$

The norms of their respective residuals are

$$\|\boldsymbol{r}_1\|_1 = 7.0 \times 10^{-5} \quad \text{and} \quad \|\boldsymbol{r}_2\|_1 = 2.4 \times 10^{-2}.$$

So which is the better solution? We are tempted to say $\hat{\boldsymbol{x}}_1$ because of its much smaller residual. But the exact solution to this system is $\boldsymbol{x} = [1, \ -1]^T$, as is easily confirmed, so $\hat{\boldsymbol{x}}_2$ is actually much more accurate than $\hat{\boldsymbol{x}}_1$. The reason for this surprising behavior is that the matrix $\boldsymbol{A}$ is ill-conditioned, as we saw in Example 2.7, and because of its large condition number, a small residual does not imply a small error in the solution. To see how $\hat{\boldsymbol{x}}_1$ was obtained, see Example 2.17.

Demo: Vanilla Gaussian Elimination

What do we get by doing Gaussian Elimination?

Row Echelon Form.

How is that different from being upper triangular?

Zeros allowed on and above the diagonal.

What if we do not just eliminate downward but also upward?

That's called *Gauss-Jordan elimination*. Turns out to be computationally inefficient. We won't look at it.

Remark 4.14. Also note that a matrix is in row echelon form if the first non-zero entry of each row (what was the pivot during gaussian elimination) is to the first of the first non-zero entry of any preceding row; moreover, entries in rows above the pivot (but in the same column) must be 0.

Elimination Matrices

What does this matrix do?

$$\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ -\frac{1}{2} & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix} \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}$$

- Add $(-1/2)\times$ the first row to the third row.
- One elementary step in Gaussian elimination
- Matrices like this are called *Elimination Matrices*

Remark 4.15. If we add k to the identity matrix at entry i, j, and left multiply the resultant matrix C by some matrix of interest A, then the result is to take the j th row of A multiply it by k and then add it to i. We can undo this process by using the same matrix but, in place of k, using $-k$. This second matrix is the inverse to the elimination matrix C.

Elimination Matrices

What does this matrix do?

$$\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ -\frac{1}{2} & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix} \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}$$

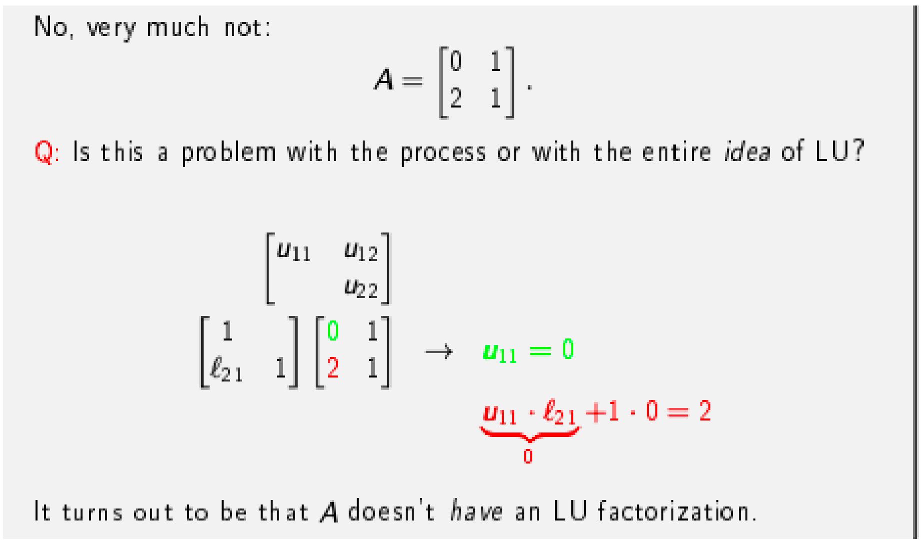
- Add $(-1/2)\times$ the first row to the third row.
- One elementary step in Gaussian elimination
- Matrices like this are called *Elimination Matrices*

Remark 4.16. Suppose that we multiply A by an elimination matrix M_1, then by M_2 up to M_l, where M_l is the last matrix required to turn A into Row Echelon Form. Eventually, we will have

$$(M_l \dots M_1)A = U \implies A = (M_l \dots M_1)^{-1}U$$

At first glance, this is okay, because it turns out that left multiplication of an elimination matrix X by Y such that X has a non-zero off diagonal at column i and Y has a non-zero off diagonal at column j where $i < j$ results in an elimination matrix that just merges X and Y[1]

For whatever reason, pivoting foils this attempt:



No, very much not:

$$A = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}.$$

Q: Is this a problem with the process or with the entire *idea* of LU?

$$\begin{bmatrix} u_{11} & u_{12} \\ & u_{22} \end{bmatrix}$$

$$\begin{bmatrix} 1 & \\ \ell_{21} & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} \to u_{11} = 0$$

$$\underbrace{u_{11} \cdot \ell_{21}}_{0} + 1 \cdot 0 = 2$$

It turns out to be that A doesn't *have* an LU factorization.

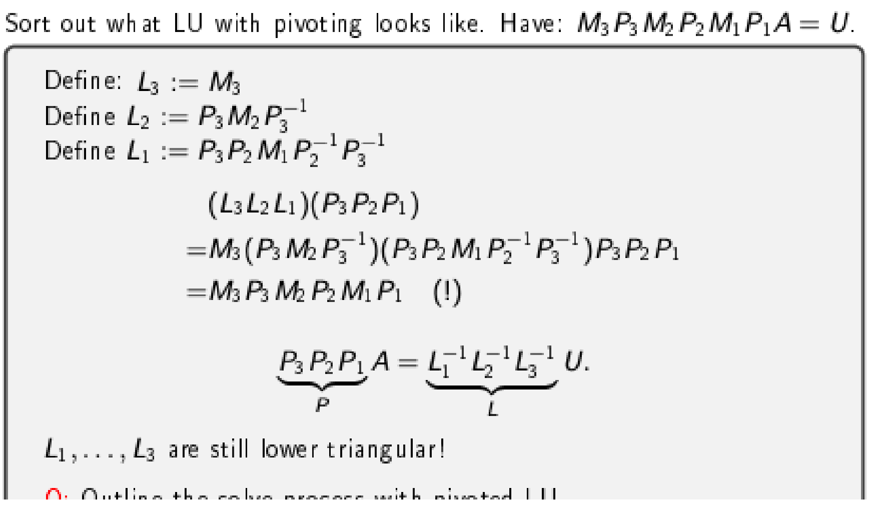
The solution is to repeatedly apply permutations to A (in the form of permutation matrices) so that the pivot is the largest element in terms of absolute value in its column.

Thus, we now have

$$(M_l P_l \dots M_1 P_1)A = U \implies A = (M_l P_l \dots M_1 P_1)^{-1} U$$

However, what should be L above is not always left triangular. It can be shown that a factorization of $(M_l P_l \dots M_1 P_1)^{-1}$ does, however, give us a lower triangular system.

[1]Note that merging also takes place if we multiply two elimination matrices that have their off diagonal non-zero entry in the same column as each other.



Sort out what LU with pivoting looks like. Have: $M_3P_3M_2P_2M_1P_1A = U$.

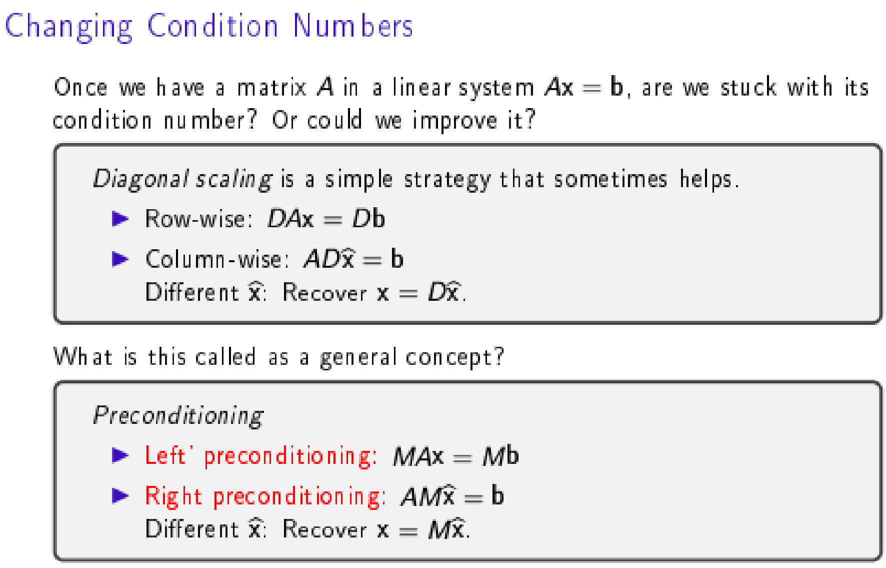
Define: $L_3 := M_3$
Define $L_2 := P_3M_2P_3^{-1}$
Define $L_1 := P_3P_2M_1P_2^{-1}P_3^{-1}$

$$\begin{aligned}&(L_3L_2L_1)(P_3P_2P_1)\\ =&M_3(P_3M_2P_3^{-1})(P_3P_2M_1P_2^{-1}P_3^{-1})P_3P_2P_1\\ =&M_3P_3M_2P_2M_1P_1 \quad (!)\end{aligned}$$

$$\underbrace{P_3P_2P_1}_{P}A = \underbrace{L_1^{-1}L_2^{-1}L_3^{-1}}_{L}U.$$

$L_1, \dots, L_3$ are still lower triangular!

Q: Outline the solve process with pivoted LU



Changing Condition Numbers

Once we have a matrix A in a linear system $A\mathbf{x} = \mathbf{b}$, are we stuck with its condition number? Or could we improve it?

Diagonal scaling is a simple strategy that sometimes helps.

- Row-wise: $DA\mathbf{x} = D\mathbf{b}$
- Column-wise: $AD\hat{\mathbf{x}} = \mathbf{b}$
  Different $\hat{\mathbf{x}}$: Recover $\mathbf{x} = D\hat{\mathbf{x}}$.

What is this called as a general concept?

Preconditioning

- 'Left' preconditioning: $MA\mathbf{x} = M\mathbf{b}$
- Right preconditioning: $AM\hat{\mathbf{x}} = \mathbf{b}$
  Different $\hat{\mathbf{x}}$: Recover $\mathbf{x} = M\hat{\mathbf{x}}$.

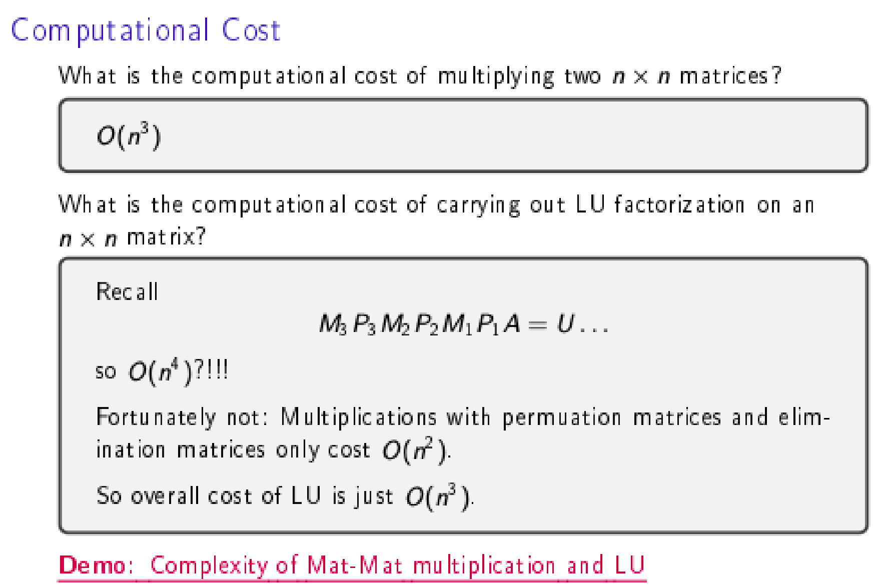
Remark 4.17. Suppose that D above satisfies $k(D) \approx 1$. Then

$$k(DA) = \|DA\|\|(DA)^{-1}\| \leq \|D\|\|A\|\|A^{-1}\|\|D^{-1}\| \leq k(A)$$

so that the condition number of $K(DA)$ is no greater than the condition number of A.

Assuming that D is invertible, then the set of x satisfying $Ax = b$ is precisely the set of x satisfying $Ax = b$. Left multiplication by D of A is called, understandably, left preconditioning and scales A in a row-wise manner; right multiplication by D of A is called right preconditioning.

Remark 4.18.



Computational Cost

What is the computational cost of multiplying two $n \times n$ matrices?

$O(n^3)$

What is the computational cost of carrying out LU factorization on an $n \times n$ matrix?

Recall

$$M_3 P_3 M_2 P_2 M_1 P_1 A = U \dots$$

so $O(n^4)$?!!!

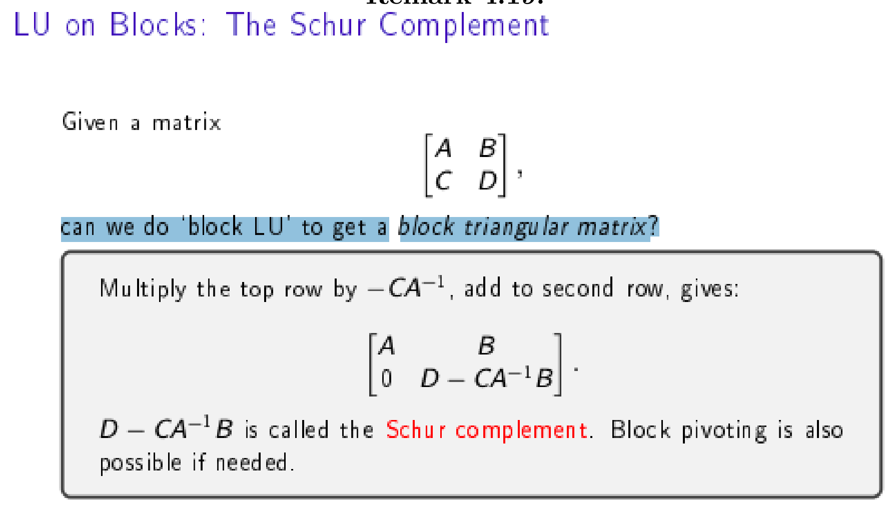
Fortunately not: Multiplications with permuation matrices and elimination matrices only cost $O(n^2)$.

So overall cost of LU is just $O(n^3)$.

Demo: Complexity of Mat-Mat multiplication and LU

Multiplication by a permutation matrix is only an n operation, since it involves switching rows. Multiplication by an elimination matrix simply involves scaling one row and mulitplying it by another, and this process is done at most n times for any one elimination matrix (making it $O(n^2)$ as well). Since these transformations are applied at most n times, the process of getting a matrix into LU form is only $O(n^3)$.

Remark 4.19.



LU on Blocks: The Schur Complement

Given a matrix

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

can we do 'block LU' to get a *block triangular matrix*?

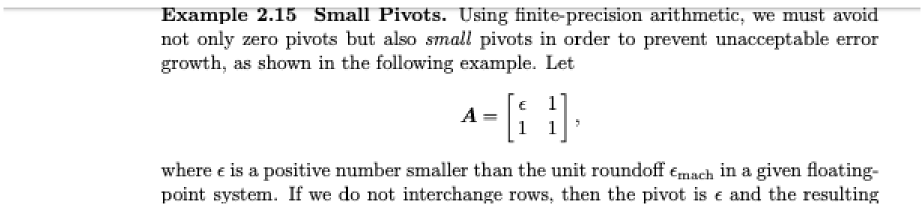
Multiply the top row by $-CA^{-1}$, add to second row, gives:

$$\begin{bmatrix} A & B \\ 0 & D - CA^{-1}B \end{bmatrix}.$$

$D - CA^{-1}B$ is called the Schur complement. Block pivoting is also possible if needed.

Not sure why this is significant.

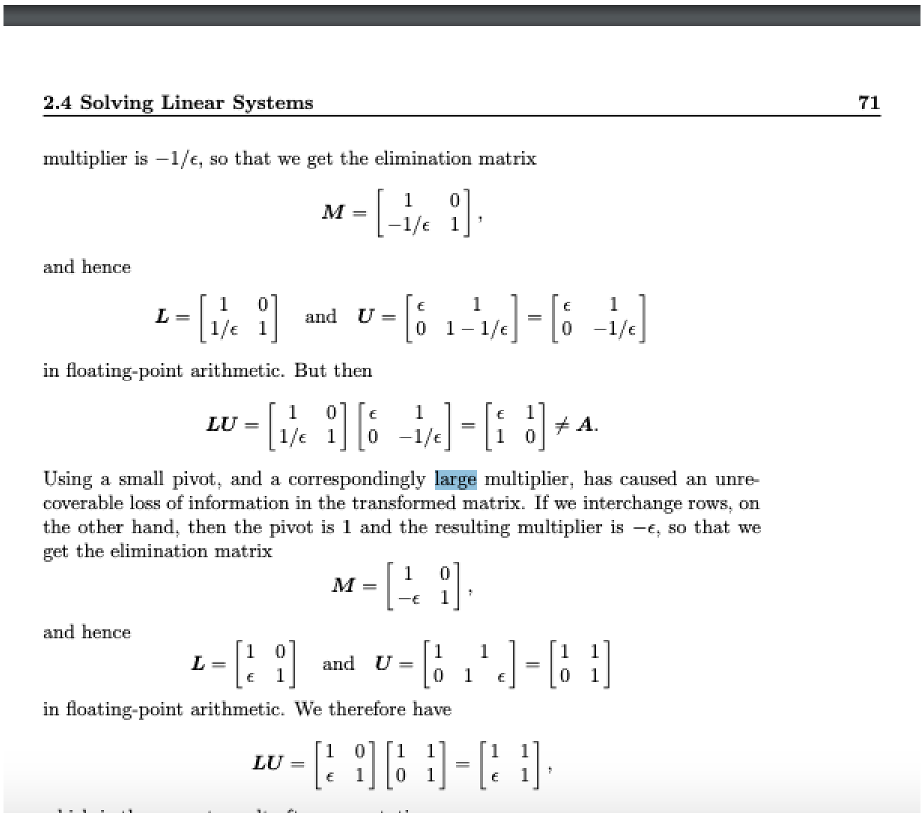
Remark 4.20. Unresolved



Example 2.15 Small Pivots. Using finite-precision arithmetic, we must avoid not only zero pivots but also *small* pivots in order to prevent unacceptable error growth, as shown in the following example. Let

$$A = \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix},$$

where ϵ is a positive number smaller than the unit roundoff ϵ_{mach} in a given floating-point system. If we do not interchange rows, then the pivot is ϵ and the resulting



multiplier is $-1/\epsilon$, so that we get the elimination matrix

$$M = \begin{bmatrix} 1 & 0 \\ -1/\epsilon & 1 \end{bmatrix},$$

and hence

$$L = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} \epsilon & 1 \\ 0 & 1 - 1/\epsilon \end{bmatrix} = \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix}$$

in floating-point arithmetic. But then

$$LU = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix} = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix} \neq A.$$

Using a small pivot, and a correspondingly large multiplier, has caused an unrecoverable loss of information in the transformed matrix. If we interchange rows, on the other hand, then the pivot is 1 and the resulting multiplier is $-\epsilon$, so that we get the elimination matrix

$$M = \begin{bmatrix} 1 & 0 \\ -\epsilon & 1 \end{bmatrix},$$

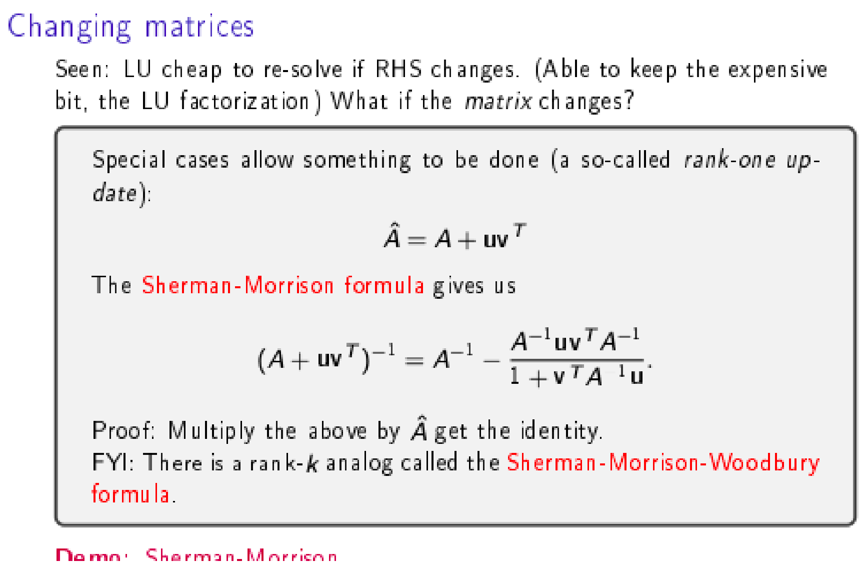
and hence

$$L = \begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} 1 & 1 \\ 0 & 1 - \epsilon \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

in floating-point arithmetic. We therefore have

$$LU = \begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ \epsilon & 1 \end{bmatrix},$$

Remark 4.21. Notice that if we already have an LU factorization, then computing a rank 1 update is just an $O(n^2)$ operation.



Changing matrices

Seen: LU cheap to re-solve if RHS changes. (Able to keep the expensive bit, the LU factorization) What if the *matrix* changes?

Special cases allow something to be done (a so-called *rank-one update*):

$$\hat{A} = A + \mathbf{u}\mathbf{v}^T$$

The Sherman-Morrison formula gives us

$$(A + \mathbf{u}\mathbf{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1 + \mathbf{v}^T A^{-1}\mathbf{u}}.$$

Proof: Multiply the above by $\hat{A}$ get the identity.
FYI: There is a rank-k analog called the Sherman-Morrison-Woodbury formula.

Demo: Sherman-Morrison

For

$$(A + uv^T)^{-1} b = A^{-1}b - \frac{(A^{-1}u)\, v^T A^{-1} b}{1 + v^T A^{-1} u}$$

And $A^{-1}x$ for any x is an $O(n^2)$ operation. The only other operation in this formula is to compute a dot product.

5 Lecture 7

5.1 Quiz