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:

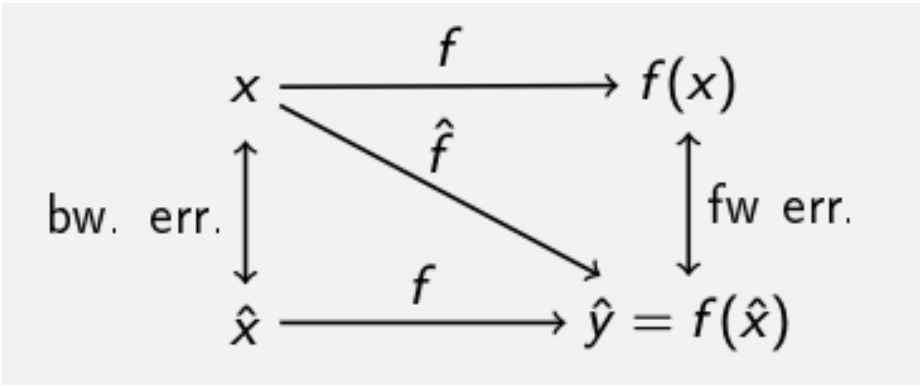
Suppose we use the surface area of the earth as an input. To calculate the surface area, we first assume that the earth is a sphere, then we somehow measure the radius and then we compute that area by multiplying the radius squared by π.

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

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

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

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

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 $x = b.\underbrace{b_1 b_2 \ldots b_{52}}_{\text{all useable digits consumed}} d_1 d_2 d_3$.

We need to somehow represent x in our floating point system. To do so, we round on the basis of $d_1 d_2 d_3$. Now we could make the scheme round to nearest, meaning that add 1 to b_{52} if $d_1 d_2 d_3 > 100$ and do nothing if $d_1 d_2 d_3 < 100$, but we must then choose what to do when $d_1 d_2 d_3 = 100$. 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 $\text{fl}(x(1+\epsilon)) > x$. 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:

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

□

4 Lecture 4

4.1 Quiz

Problem 1.

Estimating Output Error with Condition Numbers A function f(x) has a condition number of 2.3×10^2. For x=65 with an absolute error Δx=5×10−5, what is an upper bound on the relative error of the output?

TIP: On any quiz/exam question where you are asked to enter a number, you can also enter an expression, which will be evaluated for you. No need to bring out the calculator. (I.e. in the box below, you could simply enter "100*13", which would count the same as entering "1300".)

Make sure your answer has at least two accurate digits.

Solution 4.1. Relative Error <= Relative Input Error * Condition Number Relative Error <= (Machine Epsilon or some problem specific relative input error) * condition Number <= $(5 * 10^-5)/(65) * 2.3 * 10^2$

Problem 2.

Floating Point Rounding What is the value of $2\sqrt{\times}$ when calculated in a decimal floating point system with 3 digits that uses rounding? (Apply rounding to both the input data and the result of the arithmetic.)

Solution 4.2. $\sqrt{2} \approx 1.41$ and $\pi \approx 3.14$. Multiply both and round to 3 decimal places.

Problem 3.

In a normalized floating point system that supports subnormal numbers, which of the following binary operations on two positive numbers could lead to overflow?

Solution 4.3. Addition, Multiplication and Division can all lead numbers to explode outside of the normalized floating number range.

If two addends are already normalized floating point numbers, however, then there is no chance that subtracting these numbers can lead to a result that is not a normalized floating point[1]

Problem 4.

1 point Machine Epsilon and UFL Which of the following statements are true?

Select all that apply: Machine epsilon places a lower bound on the magnitude of normal floating point numbers. Machine epsilon is determined by the number of bits in the significand. Machine epsilon places an upper bound on the relative error of representing a real number in a floating point format. The UFL is determined by the number of bits in the exponent.

Solution 4.4. The bottom 3 are correct; the first is wrong, because the magnitude of normal floating point numbers is determined by the exponent range; the second is correct, because machine epsilon is $\beta^{-(p-1)}$ where β is a base and p a precision; the third is correct, because given a floating point number y, the smallest perturbation to y that results in a different floating point number is $\epsilon|y|$. The fourth is correct, because the order of magnitude of UFL is β^{L+1} where the exponent range falls in $[L, U]$.

Problem 5.

Floating Point Relative Error Which of the following is closest to the relative error from representing $e \times 2 - 145$ in IEEE single precision.

Solution 4.5. Notice that $e \times 2^{-145}$ is subnormal number. The underflow limit for single precision IEEE floating point numbers is 2^{-126}. Anything smaller requires that the leading 1 be shifted along the fractional part of a number. Since $e \times 2^{-145} \approx 2^{-144}$, it follows that we need to shift the leading 1 18 units to the right of the decimal. This will leave 5 digits (bits) in the fractional component for futher reprsentation. If we have n digits, then our relative error is approximately β^{-n}. Whence, the relative error here is 2^{-5}.

Problem 6.

Properties of Floating Point Operations Which of the following is true for floating point multiplication and addition? Select all that apply: Floating point addition is associative. Floating point multiplication is commutative. Floating point multiplication is associative. Floating point addition is commutative.

Solution 4.6. Floating point multiplication and addition are commutative but not associative.

I skip problem 7 – if you understood the solution to an earlier problem, you'll be okay.

Example 4.7. To perform floating point addition, suppose that you have $b_0.b_1 \ldots b_n \times 2^x$ and $d_0.d_1 \ldots d_n \times 2^y$ where $x > y$; make sure that you multiply the second number by 2^{x-y} and then perform grade school addition.

[1]Recall that a normalized floating point number has 1 in the fractional part before the significand.

Definition 4.8. 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 4.9. 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 4.10. 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 4.11.

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

for all x that A can multiply.

Proof. Follows by definition. □

Proposition 4.12.

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

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

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 4.13. 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

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

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

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

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

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

Proposition 4.15. $k(A) = k(A^{-1})$ 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 $A^{-1}y = x$. □

Proposition 4.16.

- 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

$$\|AB\| = \frac{\|ABx\|}{\|x\|} \leq \|A\|\|B\|$$

  - $\|A + B\| \leq \|A\| + \|B\|$.
  - $\|AA^{-1}\| = 1 \leq \|A\|\|A^{-1}\| \implies 1 \leq k(A)$
- $k(\gamma A) = k(A)$.
- $\|\gamma A\| = \gamma\|A\|$.
- $k(A) = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|} (\min_{x\neq 0} \frac{\|Ax\|}{\|x\|})^{-1}$

–

By definition:

$$\left\|A^{-1}\right\| = \max_{x \neq 0} \frac{\left\|A^{-1}x\right\|}{\|x\|}$$

Let $y = A^{-1}x$, meaning that $Ay = x$. Then

$$= \max_{x \neq 0} \frac{\|y\|}{\|Ay\|}$$

Since $y = A^{-1}y$, we can consider the max over y instead of x:

$$= \max_{y \neq 0} \frac{\|y\|}{\|Ay\|}$$
$$= (\min_{y \neq 0} \frac{\|Ay\|}{\|y\|})^{-1}$$

- If D is a diagonal matrix, then $k(D) = \frac{\max d_{i,i}}{\min_{d_{i,i}}}$
- An orthogonal matrix has norm 1 in any norm.

Remark 4.17. A computational technique to evaluating a condition number is multiply a matrix A by many vectors that have unit norm and take the maximum norm of the resultant vectors divided by the minimum norm of the resultant vectors.