CS 370 Numerical Computation

Keven Qiu
Instructor: Christopher Batty
Winter 2024

Contents

Part I

Floating Point Numbers

Chapter 1

Floating Point Number Systems

The real numbers $\mathbb{R}$ are infinite in extent and in density. The standard (partial) solution is to use floating point numbers to approximate the reals.

Definition: Floating Point Number System

An approximate representation of $\mathbb{R}$ using a finite number of bits.

An analytic solution is an exact solution, whereas a numerical solution is the approximate solution.

We can express a real number as an infinite expansion relative to some base β. For example, in base 10

$$\frac{73}{3} = 24.3333\ldots = 2 \times 10^1 + 4 \times 10^0 + 3 \times 10^{-1} + \cdots$$

After expressing the real number in the desired base β, we multiply by a power of β to shift it into a normalized form:

$$0.d_1 d_2 d_3 d_4 \cdots \times \beta^p$$

where d_i are digits in base β, i.e., $0 \le d_i < \beta$, normalized implies we shift to ensure $d_1 \neq 0$, and the exponent p is an integer.

Density (or precision) is bounded by limiting the number of digits, t. Extent (or range) is bounded by limiting the range of values for exponent p.

Definition: Floating Point Representation

The finite form

$$\pm 0.d_1 d_2 \cdots d_t \times \beta^p$$

for $L \le p \le U$ and $d_1 \neq 0$.

The four integer parameters (β, t, L, U) characterize a specific floating point system F.

Overflow/underflow errors:

- If the exponent p is too big or too small, our system cannot represent the number.
- When arithmetic operations generate such a number, this is called overflow or underflow.
- For underflow, we simply round to 0.
- For overflow, we typically produce a $\pm\infty$ or NaN.

IEEE single precision (32 bits) has ($\beta = 2, t = 24, L = -126, U = 127$) and IEEE double precision (64 bits) has ($\beta = 2, t = 53, L = -1022, U = 1023$).

Unlike fixed point, floating point numbers are not evenly spaced.

There are two ways to convert reals to floats:

1. Round-to-nearest: rounds to closest available number in F.
2. Truncation: rounds to next number in F towards 0.

1.1 Measuring Error

Our algorithms will compute approximate solutions to problems.

Let x_{exact} be the true analytical solution and x_{approx} be the approximate numerical solution.

Definition: Absolute Error

$$E_{abs} = |x_{exact} - x_{approx}|$$

Definition: Relative Error

$$E_{rel} = \frac{|x_{exact} - x_{approx}|}{x_{exact}}$$

Relative error is often more useful because it is independent of the magnitudes of the numbers involved and related the number of significant digits in the result.

A result is correct to roughly s digits if $E_{rel} \approx 10^{-s}$ or

$$0.5 \times 10^{-s} \leq E_{rel} < 5 \times 10^{-s}$$

For floating point system F, the relative error between $x \in \mathbb{R}$ and its floating point approximation, $fl(x)$, has a bound, E, such that

$$(1 - E)\,|x| \leq |fl(x)| \leq (1 + E)\,|x|$$

Definition: Machine Epsilon/Unit Round-Off Error

The maximum relative error E for converting a real into a floating point system.

It is defined as the smallest value such that $fl(1+E) > 1$ under the given floating point system.

These definitions give a rule $fl(x) = x(1+\delta)$ for some $|\delta| \leq E$. δ may be positive or negative. E is defined as positive.

For a FP system $F = (\beta, t, L, U)$:

- Rounding to nearest: $E = \frac{1}{2}\beta^{1-t}$.
- Truncation: $E = \beta^{1-t}$.

Example: Find E for $F = (\beta = 10, t = 3, L = -5, U = 5)$.

Under round to nearest:

$$E = \frac{1}{2}(10)^{1-3} = 5 \times 10^{-3}$$

Consider the smallest representable number above 1. We have $1 = 0.100 \times 10^1$ in F. The next largest is 0.101×10^1. For $fl(1+E)$ to exceed 1, we must add $0.0005 \times 10^1 = 5 \times 10^{-3}$ to get halfway to the next number, where rounding occurs.

Under truncation:

$$E = 10^{1-3} = 10^{-2}$$

1.2 Arithmetic with Floating Point

IEEE standard requires that for $w, z \in F$,

$$w \oplus z = fl(w+z) = (w+z)(1+\delta)$$

where $\oplus$ is the floating point addition.

This rule only applies to *individual* FP operations. So it is not generally true that

$$(a \oplus b) \oplus c = a \oplus (b \oplus c) = fl(a+b+c)$$

The result is order-dependent and associativity is broken.

Consider the relative error of $(a \oplus b) \oplus c$ for $a, b, c \in F$.

$$\begin{aligned}
E_{rel} &= \frac{|(a \oplus b) \oplus c - (a+b+c)|}{|a+b+c|} \\
&= \frac{|(a+b)(1+\delta_1) \oplus c - (a+b+c)|}{|a+b+c|} \\
&= \frac{|((a+b)(1+\delta_1)+c)(1+\delta_2) - a - b - c|}{|a+b+c|} \\
&= \frac{|a+b+c+(a+b)\delta_1 + (a+b+c+(a+b)\delta_1)\delta_2 - a - b - c|}{|a+b+c|} \\
&= \frac{|(a+b)\delta_1 + (a+b)\delta_1\delta_2 + (a+b+c)\delta_2|}{|a+b+c|} \\
&\leq \frac{|(a+b)\delta_1| + |(a+b)\delta_1\delta_2| + |(a+b+c)\delta_2|}{|a+b+c|} && \text{(Triangle inequality)} \\
&\leq \frac{|a+b|\,|\delta_1| + |a+b|\,|\delta_1\delta_2|}{|a+b+c|} + |\delta_2| \\
&\leq \frac{|a+b|\,E + |a+b|\,E^2}{|a+b+c|} + E && (|\delta| \leq E) \\
&\leq \frac{|a+b|}{|a+b+c|}(E+E^2) + E
\end{aligned}$$

This is the bound on the relative error for $(a \oplus b) \oplus c$. We can weaken the bound slightly to make it symmetric.

$$\begin{aligned}
E_{rel} &\leq \frac{|a+b|}{|a+b+c|}(E+E^2) + \frac{|a+b+c|}{|a+b+c|}E \\
&\leq \left(\frac{|a|+|b|+|c|}{|a+b+c|}\right)(E+E^2) + \left(\frac{|a|+|b|+|c|}{|a+b+c|}\right)E \\
&\leq \left(\frac{|a|+|b|+|c|}{|a+b+c|}\right)(2E+E^2)
\end{aligned}$$

This bound will apply to both $(a \oplus b) \oplus c$ and $a \oplus (b \oplus c)$, i.e. order does not matter.

$\frac{|a|+|b|+|c|}{|a+b+c|}$ is small when the denominator is small. This occurs when quantities have differing signs and similar magnitudes, leading to cancellation.

Catastrophic cancellation occurs when subtracting of about the same magnitude, when the input numbers contain error. All significant digits cancel out, so the result will have no correct digits.

1.3 Conditioning of Problems

Problems may be ill-conditioned or well-conditioned. Consider a problem P with input I and output O.

Definition: Well-Conditioned

If a change to the input ΔI gives a small change in the output ΔO, P is well-conditioned.

1.4 Stability of Algorithms

If any initial error in the data is magnified by an algorithm, the algorithm is considered numerically unstable.

Consider the integration problem

$$I_n = \int_0^1 \frac{x^n}{x+\alpha}\,dx$$

There is a recursive algorithm to solve it. For $n \geq 0$,

$$I_0 = \log\frac{1+\alpha}{\alpha}, I_n = \frac{1}{n} - \alpha I_{n-1}$$

Stability analysis: Consider the given recursive rule for I_n. Assume there is some initial error ε_0 in I_0.

$$\varepsilon_0 = (I_0)_A - (I_0)_E$$

We will ignore any subsequent floating point error introduced during the recursion. What is ε_n, the signed error after n steps? Does ε grow or shrink?

Exact and approximate solutions both follow the recursion, so

$$\begin{aligned}\varepsilon_n &= (I_n)_A - (I_n)_E \\ &= \left(\frac{1}{n} - \alpha(I_{n-1})_A\right) - \left(\frac{1}{n} - \alpha(I_{n-1})_E\right) \\ &= -\alpha((I_{n-1})_A - (I_{n-1})_E) \\ &= -\alpha\varepsilon_{n-1}\end{aligned}$$

We have a simple recursion for ε, so we can find the closed form.

$$\varepsilon_n = -\alpha\varepsilon_{n-1} = (-\alpha)^2\varepsilon_{n-2} = \cdots = (-\alpha)^n\varepsilon_0$$

The initial error is scaled by $(-\alpha)^n$ for n steps, so if $|\alpha| \leq 1$, the error does not grow so it is stable. If $|\alpha| > 1$, then the error grows and it is unstable.

Part II

Interpolation and Splines

Chapter 2

Interpolation

Interpolation Problem

Given a set of data points from an unknown function $y = p(x)$, approximate p's value at other points.

2.1 Polynomial Interpolation

Theorem (Unisolvence Theorem)

Given n data pairs (x_i, y_i) with distinct x_i, there is a unique polynomial $p(x)$ of degree $\leq n - 1$ that interpolates the data.

For n points, find all the coefficients c_i of the generic polynomial

$$p(x) = c_1 + c_2 x + \cdots + c_n x^{n-1}$$

Each (x_i, y_i) gives one linear equation. We then solve the $n \times n$ linear system.

2.2 Vandermonde Matrices

In general, we get a linear system

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} c_1 \\ \cdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \cdots \\ y_n \end{bmatrix}$$

or

$$Vc = y$$

Definition: Vandermonde Matrix

V in the system $Vc = y$.

2.2.1 Monomial Basis

The polynomial $p(x) = \sum_{i=1}^{n} c_i x^{i-1}$ is the monomial form. The sequence $1, x, x^2, \ldots$ is the monomial basis.

2.3 Lagrange Basis

We let the polynomial have form

$$p(x) = \sum_{k=1}^{n} y_k L_k(x)$$

where the y's are the new coefficients (we have these y from the data points).

At each (x_i, y_i) data point, activate only the ith term, $y_i L_i(x)$. So for a data point x_1, we set $L_1(x_1) = 1$ and $L_{i \neq 1} = 0$, since $p(x_1) = y_1$. Repeat this for all data points. This gives us n equations and n unknowns, so it gives a unique polynomial for each L_i.

To solve for these L_i, we can either solve them directly or use this formula:

$$L_k(x) = \frac{(x - x_1) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_1) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}$$

E.g. Lagrange polynomial for 2 points. Find $L_1(x), L_2(x)$ and write $p(x)$ in terms of L_i's with points $(1, 2), (-1, 4)$.

$$L_1(x) = \frac{(x - (-1))}{(1 - (-1))} = \frac{x+1}{2}, L_2(x) = \frac{(x-1)}{(-1-1)} = \frac{1-x}{2}$$

So

$$p(x) = 2\left(\frac{x+1}{2}\right) + 4\left(\frac{1-x}{2}\right) = x + 1 + 2 - 2x = 3 - x$$

Runge's phenomenon says that for higher degree polynomials for many points gives more oscillation. Piecewise polynomials will be used to fix this.

2.4 Piecewise Functions

We want continuity, so we try piecewise linear functions. This is done by fitting a line per pair of adjacent points.

Often piecewise linear is not satisfactory, we want smoothness (continuity of derivatives).

2.4.1 Hermite Interpolation

The problem of fitting a polynomial given function values and its derivative.

Typically we do cubic Hermite interpolation. If we have many points, we can do one cubic per pair of adjacent points. Sharing the derivative at points ensure first derivative (C^1) continuity.

Closed Form Hermite Interpolation: Data is $x_1, y_1, s_1, x_2, y_2, s_2$. We use the polynomial form

$$p(x) = a + b(x - x_1) + c(x - x_1)^2 + d(x - x_1)^3$$

which simplifies out expressions slightly.

The derivative is

$$p'(x) = b + 2c(x - x_1) + 3d(x - x_1)^2$$

Plugging in the data points:

$$\begin{aligned} p(x_1) &= a = y_1 \\ p'(x_1) &= b = s_1 \\ p(x_2) &= a + b(x_2 - x_1) + c(x_2 - x_1)^2 + d(x_2 - x_1)^3 = y_2 \\ p'(x_2) &= b + 2c(x_2 - x_1) + 3d(x_2 - x_1)^2 = s_2 \end{aligned}$$

Solving the system gets

$$\begin{aligned} a &= y_1 \\ b &= s_1 \\ c &= \frac{3y_1' - 2s_1 - s_2}{\Delta x_1} \\ d &= \frac{s_1 + s_1 - 2y_1'}{(\Delta x_1)^2} \end{aligned}$$

where $\Delta x_1 = x_2 - x_1$ and $y_1' = \frac{y_2 - y_1}{\Delta x_1}$. We would use this to get the cubic for each interval.

Definition: Knot

Point where the interpolant transitions from one polynomial/interval to another.

Definition: Node

Point where some control points/data are specified.

For Hermite interpolation, these are the same. For other curve types, they can differ.

2.5 Cubic Spline Interpolation

If we are not given derivatives and we want to fit a cubic, we need more data than just the two points.

Approach: Fit a cubic $S_i(x)$ on each interval, but now require matching first and second derivatives (C^2 continuity) between intervals.

Definition: Interpolating Conditions

On the interval $[x_i, x_{i+1}]$, $S_i(x_i) = y_i$ and $S_i(x_{i+1}) = y_{i+1}$.

Interval endpoint values match.

Definition: Derivative Conditions

At each interior point x_{i+1}, $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$ and $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$.

Interior knot first and second derivatives match across intervals.

Assuming n data point, and a cubic which has 4 unknowns for each $n-1$ intervals has $4n-4$ unknowns.

Assuming n data points, we have 2 endpoint interpolating conditions per interval, so $2n-2$ equations and 2 derivative conditions per interior point, so $2n-4$ equations. We have a total of $4n-6$ equations.

Since $4n-6 < 4n-4$, there is not enough information for a unique solution. We need 2 more equations, usually at domain endpoints called boundary conditions or end conditions.

Definition: Clamped Boundary Conditions

Slope set to a specific value.

Definition: Free Boundary Conditions

Second derivative is set to 0.

Basic algorithms (Gaussian elimination) for linear systems take $O(n^3)$ time for n unknowns. For the special case of cubic splines, we can do $O(n)$.

2.5.1 Cubic Splines via Hermite Interpolation

Use Hermite interpolation as a stepping stone to build a cubic spline.

1. Express unknown polynomials with closed form Hermite equations.

2. Treat s_i as unknowns.

3. Solve for s_i that give continuous second derivatives (force it to satisfy cubic spline).

4. Given s_i, plug into closed form Hermite equations to recover polynomial coefficients a_i, b_i, c_i, d_i.

Part III

Ordinary Differential Equations

Part IV

Fourier Transforms

Part V

Numerical Linear Algebra