How to Pass CS 370

Jim Zhang

Winter 2017

Contents

1 Floating point arithmetic

1.1 Floating point number systems

$\text{F}(\beta, t, L, U)$ contains 0 and

$$\pm 0.b_1 b_2 ... b_t \times \beta^d, \quad b_1 \neq 0, 0 \leq b_i < \beta, L \leq d \leq U$$

For example, $\text{F}(2, 24, -126, 127)$ is single precision, and $F(2, 53, -1022, 1023)$ is double precision. $fl(x)$ brings x into F.

1.2 Absolute error

$$\text{Err}_{abs} = |x_{exact} - x|$$

1.3 Relative error

$$\text{Err}_{rel} = \frac{|x_{exact} - x|}{|x_{exact}|}$$

1.4 Machine epsilon

$$E = \frac{fl(x) - x}{x}$$

$|E|$ is at most β^{1-t} for truncation, $\frac{1}{2}\beta^{1-t}$ for rounding.

1.5 Floating point error (12-13)

Be very rigorous about this. Write $fl(x) = x(1 + \delta_i)$, $\delta_i < \epsilon$. (If x is already in the FPNS, $fl(x) = x$.) **Each operation needs a different** δ_i. Use the triangle inequality to break $|\delta_1 + \delta_2| \leq |\delta_1| + |\delta_2|$, then apply $|\delta_i| < \epsilon$.

1.6 Catastrophic cancelling

For $x \oplus y$,

$$\text{Err}_{rel} = \frac{(|x| + |y|)(2\epsilon + \epsilon^2)}{|x + y|}$$

So catastrophic cancelling happens when $|x + y|$ is small.

1.7 Conditioning

Well-conditioned problems don't change much with the input. **Ill-conditioned** problems are not well-conditioned.

1.8 Stability

To determine stability of an iterative computation, find the error at each step.

$$\varepsilon_n = I_n^{ex} - I_n^{app}$$

Solve for ε_n in terms of initial errors.

Example If we want to compute

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

with the recurrence

$$I_0 = \ln\left(\frac{1 + \alpha}{\alpha}\right)$$
$$I_{n+1} = \frac{1}{n+1} - \alpha I_n$$

Then

$$\begin{aligned} e_{n+1} &= I_{n+1}^{ex} - I_{n+1}^{app} \\ &= \left(\frac{1}{n+1} - \alpha I_n^{ex}\right) - \left(\frac{1}{n+1} - \alpha I_n^{app}\right) \\ &= -\alpha e_n \\ &= (-\alpha)^{n+1} e_0 \end{aligned}$$

So error explodes for $|\alpha| > 1$.

2 Interpolation

You have N points (x_i, y_i), $1 \leq i \leq N$, $x_i < x_{i+1}$. Get a nice curve $p(x)$ that passes through all of them.

2.1 Single polynomial

A single degree $< N$ polynomial that passes through all points.

2.1.1 Vandermonde matrix

Solve

$$\begin{aligned} V\vec{c} &= \vec{y} \\ V &= \begin{bmatrix} 1 & x_1 & ... & x_1^{N-1} \\ ... & ... & ... & ... \\ 1 & x_N & ... & x_N^{N-1} \end{bmatrix} \\ c &= [c_1, ..., c_N]^T \\ y &= [y_1, ..., y_N]^T \end{aligned}$$

Then the curve is

$$p(x) = c_1 + c_2 x_1 + ... + c_N x_N^{N-1}$$

2.1.2 Lagrange form

$$p(x) = \sum_{i=1}^{n} y_i L_i(x)$$
$$L_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

Note that $L_i(x) = 0$ for any x other than x_i.

2.2 Hermite interpolation

Given two points $(x_L, y_L), (x_R, y_R)$ and endpoint slopes s_L, s_R, the polynomial

$$\begin{aligned}
s(x) &= c_1 + c_2(x - x_L) + c_3(x - x_L)^2 + c_4(x - x_L)^3 \\
c_1 &= y_L \\
c_2 &= s_L \\
c_3 &= \frac{3y'_L - s_R - 2s_L}{\Delta x} \\
c_4 &= \frac{-2y'_L + s_R + s_L}{(\Delta x)^2}
\end{aligned}$$

where

$$\begin{aligned}
y'_L &= \Delta y / \Delta x \\
\Delta x &= x_R - x_L
\end{aligned}$$

works.

2.3 Cubic splines

Use $N - 1$ polynomials, each of the form

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

for the piece $x_i \leq x \leq x_{i+1}$.

2.3.1 Unknowns and equations

You have $4N - 4$ unknowns from polynomial coefficients. As for equations, you have:

- $2N - 2$ from endpoints: $S_i(x_i) = y_i$, $S_i(x_{i+1}) = y_{i+1}$
- $2N - 4$ from continuity of s' and s'' at interior points
- 2 from boundary conditions

2.3.2 Boundary conditions

- Natural spline: $s''(x_1) = s''(x_N) = 0$
- Clamped: $s'(x_1) = s_1$, $s'(x_N) = s_N$ for some s_1, s_N
- Periodic: $s'(x_1) = s'(x_N)$, $s''(x_1) = s''(x_N)$
- Not-a-knot: s''' continuous at x_2 and x_{N-1}

2.3.3 Solving

You could solve a size-$(4N - 4)$ linear equation, but that takes $O(N^3)$ time, so don't. Instead, find unknown derivatives $s_1, ..., s_N$ using a tridiagonal matrix:

$$T\vec{s} = \vec{r}$$

where

$$\begin{aligned}
T_{i,i-1} &= \Delta x_i \\
T_{i,i} &= 2(\Delta x_{i-1} + \Delta x_i) \\
T_{i,i+1} &= \Delta x_{i-1} \\
T_{i,_} &= 0
\end{aligned}$$

and

$$r_i = 3(\Delta x_i y'_{i-1} + \Delta x_{i-1} y'_i)$$

Then use Hermite interpolation.

$$\begin{aligned}
a_i &= y_i \\
b_i &= s_i \\
c_i &= \frac{3y'_i - 2s_i - s_{i+1}}{\Delta x_i} \\
d_i &= \frac{s_i + s_{i+1} - 2y'_i}{(\Delta x_i)^2}
\end{aligned}$$

where

$$\begin{aligned}
y'_i &= \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \\
\Delta x_i &= x_{i+1} - x_i
\end{aligned}$$

2.4 Parametric curves

Given points (x_i, y_i) that you want to connect in order, express each dimension in terms of time to get (t_i, x_i) and (t_i, y_i), and interpolate separately. Two ways to pick t:

2.4.1 Simple time

$$t_i = i$$

2.4.2 Approx. arc-length parameterization

$$t_0 = 0, \qquad t_{i+1} = t_i + \sqrt{\Delta x_i^2 + \Delta y_i^2}$$

3 Differential equations

$$\begin{aligned}
y'(t) &= f(t, y(t)) \\
y(t_0) &= y_0
\end{aligned}$$

We wish to find $y(t)$ for all t. How do?

3.1 Stepping schemes

Take steps of size h.

3.1.1 Forward Euler

$$y_{n+1} = y_n + hf(t_n, y_n)$$

Explicit, single-step, LTE $O(h^2)$. Like integrating with rectangles.

3.1.2 Trapezoidal

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_{n+1}, y_{n+1}) + f(t_n, y_n)\right]$$

Implicit, single-step, LTE $O(h^3)$. Like integrating with exact trapezoids.

3.1.3 Improved Euler

$$\begin{aligned} y^*_{n+1} &= y_n + hf(t_n, y_n) \\ y_{n+1} &= y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y^*_{n+1})\right] \end{aligned}$$

Explicit, single-step, LTE $O(h^3)$. Like integrating with approximate trapezoids.

3.1.4 Runge Kutta

Family of methods that generalizes Forward Euler. For example, RK4 is

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\ k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\ k_4 &= f(t_n + h, y_n + hk_3) \end{aligned}$$

Explicit, single-step, LTE $O(h^5)$. Like integrating with parabolas.

3.1.5 Backward Euler

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

Implicit, single-step, LTE $O(h^2)$.

3.1.6 Adams-Bashforth

$$y_{n+2} = y_{n+1} + h\left[\frac{3}{2}f(t_{n+1}, y_{n+1}) - \frac{1}{2}f(t_n, y_n)\right]$$

Explicit, multi-step.

3.2 Systems of equations

If you have a higher degree differential equation, convert it to a system of first-order equations. Then treat each y in the stepping schemes as a vector.

3.3 Stability

(This is a different concept from accuracy.) To determine a method's stability, plug in The Test Equation:

$$y'(t) = -\lambda y(t), \quad \lambda > 0$$

Then solve for the closed form and see for what values it blows up relative to the initial value.

3.4 Local truncation error

To determine local truncation error:

1. Replace RHS of dynamics equation with exact versions.
2. Taylor expand about t_n.
3. Taylor expand the exact solution, $y(t_{n+1})$.
4. Find $y(t_{n+1}) - y_{n+1}$ to get the degree of your error.

How to Taylor expand:

$$\begin{aligned} y(t_{n+1}) &= y(t_n + h) \\ &= y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + \frac{h^3}{6}y'''(t_n) + ... \end{aligned}$$

3.5 Time step control

Use two methods y^A_{n+1}, y^B_{n+1} with different LTEs $O(h^4)$, $O(h^5)$. (here, $p = 4$)

1. Estimate error of A as $|y^A - y^B|$
2. Two options now:
   (a) If error is greater than threshold, $h \to h/2$ and recompute, OR
   (b) $h_{new} = h_{old}\left(\frac{tol}{|y^A_{n+1} - y^B_{n+1}|}\right)^{1/p}$
   Compensate for approximation by scaling tolerance down by a factor α, usually about 1/2

3.6 Convergence

1. **Well-conditioned** ODEs don't change much with a change in initial value.
2. **Round-off error** comes from floating point arithmetic
3. **Truncation error** comes from finite steps / approximate derivatives
4. As $h \to 0$, round-off error increases. As $h \to \infty$, truncation error increases.

4 Fourier analysis

You have N samples $f_0, ..., f_{N-1}$ of a signal in the time domain and you wish to find the frequency domain representation $F_0, ..., F_{N-1}$.

4.1 The Fourier transform

Do it with the Fourier transform.

$$F_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n W^{-nk}$$
$$f_n = \sum_{k=0}^{N-1} F_k W^{nk}$$

where

$$W = e^{2\pi i/N}$$

4.2 Helpful math facts

4.2.1 Geometric series

$$\sum_{n=0}^{N} r^n = \frac{1 - r^{n+1}}{1 - r}$$

4.2.2 Geometric series of W

$$\sum_{n=0}^{A} W^{nk} = \begin{cases} A+1 & k \equiv 0 \pmod{N} \\ 0 & \text{otherwise} \end{cases}$$
$$\sum_{n=0}^{N-1} W^{nk} = \begin{cases} N & k = 0 \\ 0 & k > 0 \end{cases}$$

4.3 The Fast Fourier Transform (90)

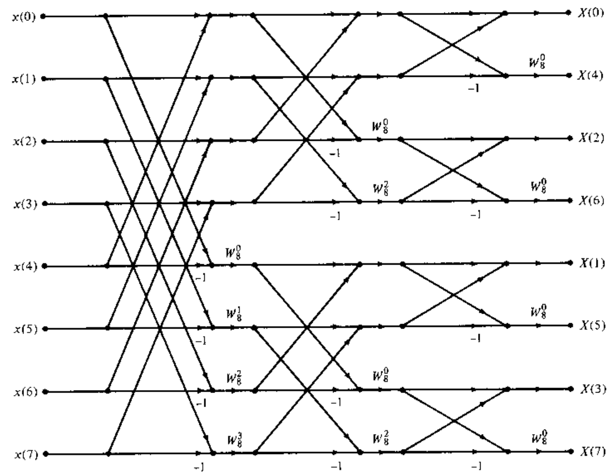
FFT is love. FFT is life. Define two vectors g and h:

$$g_n = \frac{1}{2}\left(f_n + f_{n+N/2}\right)$$
$$h_n = \frac{1}{2}\left(f_n - f_{n+N/2}\right) W^{-n}$$

Obtain $G = DFT(g)$ and $H = DFT(h)$ recursively. Then,

$$F_{0,2,...,N-2} = G$$
$$F_{1,3,...,N-1} = H$$

Elohim essaim. Note that this creates a butterfly.



To quickly get from the last column to the answer, note that the first and last columns are bit-reversed.

5 PageRank

You have a directed graph of R webpages linking to each other. How do we get the probability distribution of the page a random surfer will be on after an arbitrarily long time?

5.1 Google matrix

Let α be the chance that you follow a random link instead of teleporting randomly. The Google matrix M is

$$M = \alpha P' + (1-\alpha)\frac{1}{R} ee^T$$

where the Markov chain matrix is given by

$$P' = P + \frac{1}{R} ed^T$$
$$P_{ij} = \begin{cases} 1/\deg(j) & j \to i \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$
$$d_i = \begin{cases} 1 & \deg(i) = 0 \\ 0 & \text{otherwise} \end{cases}$$

and the random teleportation matrix $\frac{1}{R}ee^T$ is all $\frac{1}{R}$. (e is the all-ones column vector.)

5.2 Eigenstuff

To find eigenvalues λ of M, solve

$$(\lambda I - M)\vec{x} = 0$$

5.2.1 Convergence of PageRank

PageRank converges to the unique eigenvector of M with eigenvalue 1. See p120-121 for proof.

6 Numerical linear algebra

We want to solve $A\vec{x} = \vec{b}$ in $O(n^2)$ time. To do this, do LU-factorization on A.

6.1 LU factorization

Find U by Gaussian elimination on A. Keep track of the elementary operations you do and fill in the correct entries in L.

$$L^{(n-1)} \cdots L^{(1)} A = U$$
$$A = \left(L^{-(1)} \cdots L^{-(n-1)}\right) U$$

6.2 Pivoting

For numerical stability, you may have to permute rows and put the maximum magnitude entry on the diagonal. Represent with

$$PA = LU$$

6.3 Solving

In order to solve $A\vec{x} = \vec{b}$, first factor $PA = LU$. Then solve the following.

$$L\vec{y} = P\vec{b}$$
$$U\vec{x} = \vec{y}$$

6.4 Condition number

6.4.1 Vector norms

$$||x||_1 = \sum |x_i|$$
$$||x||_2 = \sqrt{\sum x_i^2}$$
$$||x||_\infty = \max |x_i|$$

6.4.2 Vector norm properties

1. If $||x|| = 0$, then $x = \vec{0}$
2. $||\alpha x|| = |\alpha| \cdot ||x||$, for scalar α
3. $||x + y|| \leq ||x|| + ||y||$ (triangle inequality)

6.4.3 Matrix norms

$$||A|| = \max_{||x|| \neq 0} \frac{||Ax||}{||x||}$$

In particular,

$$||A||_1 = \text{max absolute column sum}$$
$$||A||_2 = \max |\lambda_i|^{1/2}, \quad \lambda_i \text{ eigenvalues of } A^T A$$
$$||A||_\infty = \text{max absolute row sum}$$

6.4.4 Matrix norm properties

1. $A = \mathbf{0}$ iff $||A|| = 0$
2. $||\alpha A|| = |\alpha| \cdot ||A||$, for scalar α
3. $||A + B|| \leq ||A|| + ||B||$
4. $||A\vec{x}|| \leq ||A|| \cdot ||\vec{x}||$
5. $||AB|| \leq ||A|| \cdot ||B||$
6. $||I|| = 1$

6.4.5 Condition number (108-109)

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

6.4.6 Condition number meaning

$$Ax = b$$

Perturb $b \to b + \Delta b$, and the solution becomes

$$A(x + \Delta x) = b + \Delta b$$

Relative error is bounded by

$$\frac{||\Delta x||}{||x||} \leq \kappa(A) \frac{||\Delta b||}{||b||}$$

6.4.7 Condition number properties

1. $\kappa(A) \geq 1$
2. $\kappa(\alpha A) = \kappa(A)$, for scalar α

6.4.8 Residual (109-110)

$$r = b - A(x + \Delta x)$$

Relative error is bounded by

$$\frac{||\Delta x||}{||x||} \leq \kappa(A) \frac{||r||}{||b||}$$

where

$$\frac{||r||}{||b||} \approx \epsilon_{machine}$$

7 Just in case

7.1 2×2 matrix inverse

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

7.2 Diagonal matrix inverse

$$\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}^{-1} = \begin{bmatrix} 1/a & 0 & 0 \\ 0 & 1/b & 0 \\ 0 & 0 & 1/c \end{bmatrix}$$

7.3 Linear algebra

- $(AB)^T = B^T A^T$
- Eigenvalues of A^{-1} are $1/\lambda_i$, λ_i eigenvalues of A