CS370 Midterm Cheat Sheet

FP Number System

Specific floating point system: $\{\beta, t, L, U\}$.
IEEE single precision (32 bits:)

matissa sign = 1 bit exponent sign = 1 bit	mantissa = 23 bit exponent = 7 bits

Relative error: $E_{rel} = E_{abs} \;/\; |x_{exact}|$.
A result is correct to *roughly* s digits if

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

Machine epsilon: the smallest value E such that $fl(1 + E) > 1$ under the given FP system.
FP Addition: $w \oplus z = fl(w + z) = (w + z)(1 + \delta)$ with $|\delta| \leq E$. Note that it is **NOT** true that

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

Interpolation

Unisolvence: Given n data pairs (x_i, y_i), $i = 1, \ldots, n$ with distinct x_i, there is a unique polynomial $p(x)$ of degree $\leq n - 1$ that interpolates the data.
For polynomial interpolation, we have Vandermonde matrix:

$$\begin{bmatrix} 1 & x_1 & \cdots & {x_1}^{n-1} \\ 1 & x_2 & \cdots & {x_2}^{n-1} \\ & & \cdots & \\ 1 & x_n & \cdots & {x_n}^{n-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

and Langrange form:

$$p(x) = y_1 L_1(x) + y_2 L_2(x) + \cdots + y_n L_n(x) = \sum_i y_i L_i(x)$$

`Runge's phenomenon` suggests that we need to use other methods for high-order polynomials.

Piecewise Hermite

Hermite Interpolation: fitting function values and derivatives.
For piece wise Hermite, we have

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

where

$$a_i = y_i \qquad c_i = \frac{3{y_i}' - 2s_i - s_{i+1}}{\Delta x_i}$$

$$b_i = s_i \qquad d_i = \frac{s_{i+1} + s_i - 2{y_i}'}{{\Delta x_i}^2}$$

Knots are points where the interpolant transitions from one polynomial / interval to another.
Nodes are points where some control points/data is specified.

Cubic Splines

Fit a cubic, $S_i(x)$, on each interval, but now require matching first and second derivatives between intervals.

- Clamped/ Complete: $S'(x_1)$ and $S'(x_n)$ are specified;
- Free/ Natural: $S''(x_1) = S''(x_n) = 0$;
- Periodic: $S'(x_1) = S'(x_n)$ and $S''(x_1) = S''(x_n)$.

Derivation of Cubic Splines Equations

With clamped condition or free boundary condition:
$s_1 + s_2/2 = 3/2y'_1$ and $s_{n-1}/2 + s_n = 3/2y'_{n-1}$, and

$$\boxed{3\Delta x_{i-1}{y_i}' + 3\Delta x_i {y_{i-1}}' = \Delta x_i s_{i-1} + \Delta x_{i-1} s_{i+1} + 2s_i(\Delta x_{i-1} + \Delta x_i)}$$

for $i = 2, \ldots, n - 1$

Parametric

IDEA: Let x and y each be a function of a new parameter t. Two options:

1. Use $t_i = i$
2. Set $t_1 = 0$ and compute $t_{i+1} = t_i\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$.

ODE

Forward Euler: explicit, single-step.

1: Repeat until done:
2: $\quad y'_n = f(t_n, y_n)$;
3: $\quad y_{n+1} = y_n + h \cdot y'_n$

The (local truncation) error is $\dfrac{-h^2}{2!}y''(t_n) + O(h^3) \in O(h^2)$.

Trapezoidal:	$y(t_{n+1}) = y_n + \frac{h}{2}\Big[y'(t_{n+1}) + y'(t_n)\Big] + O(h^3)$.
Improved Forward Euler:	$y(t_{n+1}) = y_n + \frac{h}{2}\Big[f(t_{n+1}, y^*_{n+1}) + f(t_n, y_n)\Big] + O(h^3)$.
Backward Euler:	$y_{n+1} = y_n + h \cdot f(t_{n+1, y_{n+1}})$.
BDF2:	$y_{n+1} = \frac{4}{3}y_n - \frac{1}{3}y_{n-1} + \frac{2}{3}hf(t_{n+1}, y_{n+1})$.
Adams-Bashforth:	$y_{n+1} = y_n + \frac{3}{2}hf(t_n, y_n) - \frac{1}{2}hf(t_{n-1}, y_{n-1})$.

Backward/Implicit Euler and Trapezoidal are unconditionally stable.

Name			LTE
Forward Euler	Single	Explicit	$O(h)$
Improved Euler and Midpoint (2nd order Runge Kutta schemes)	Single	Explicit	$O(h^2)$
4th Order Runge Kutta	Single	Explicit	$O(h^4)$
Trapezoidal	Single	Implicit	$O(h^2)$
Backwards/Implicit Euler (BDF1)	Single	Implicit	$O(h)$
BDF2	Multi	Implicit	$O(h^2)$
2-step Adams-Bashforth	Multi	Explicit	$O(h^2)$
3rd order Adams-Moulton	Multi	Implicit	$O(h^3)$

Stability

Test Equation: $y'(t) = -\lambda \cdot y(t)$, $y(0) = y_0$, for constant $\lambda > 0$.

1. Apply a given time stepping scheme to our test equation.
2. Find the closed form of its numerical solution and error behavior.
3. Find the conditions on the timestep h that ensure stability (error approaching zero).

Truncation Error and Adaptive Time Stepping

Given a time-stepping scheme, $y_{n+1} = RHS$

1. Replace approximations on RHS with exact versions. e.g, $y_n \to y(t_n)$ and $f(t_{n+1}, y_{n+1}) \to y'(t_{n+1})$, etc.
2. Taylor expand all RHS quantities about time t_n (if necessary).
3. Taylor expand the exact solution $y(t_{n+1})$ to compare against.
4. Compute difference $y(t_{n+1}) - y_{n+1}$. Lowest degree non-canceling power of h gives the local truncation error.

Fourier

Some useful results:

$$\cos(\theta) = \frac{e^{i\theta} + e^{-i\theta}}{2} \qquad \text{and,} \qquad \sin(\theta) = \frac{e^{i\theta} - e^{-i\theta}}{2i}$$

We define N^{th} **roots of unity** to be $W := \exp\left(\frac{2\pi i}{N}\right)$, who:

$$\sum_{j=0}^{N-1} W^{j(k-\ell)} = N\delta_{k,\ell}$$

Now we have our discrete **fourier transform** pair (time-domain data f_n, frequency domain F_k):

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

- The sequence $\{F_k\}$ is doubly infinite and periodic. i.e., if we allow $k < 0$ or $k > N-1$, the F_k coefficients repeat;
- If data f_n is real, $F_k = \overline{F_{N-k}}$.

However, this takes $O(N^2)$ complex floating point operations, so we introduce **fast fourier transform**:

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

There will be log2 N of these stages. Each stage requires O(N) complex floating point operations.

Google Page

We first have our markov chain matrix P:

$$P_{ij} = \begin{cases} \frac{1}{\deg(i)} & \text{if } i \to j \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

and if there is dead end, we let vector d to be such that $d_i = 1$ if $\deg(i) = 0$, and define

$$P' = P + \frac{1}{R} e d^T$$

in addition, to avoid closed cycles, we define

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

for some constant α. If M is a positive markov matrix, the iteration

$$p^{\infty} = \lim_{k\to\infty} (M^k) p^0$$

converges to a unique vector p^{∞}, for any p^0.

NLA

If $A = \begin{bmatrix} a & b & c \\ \boxed{d} & e & f \\ \boxed{g} & \boxed{h} & i \end{bmatrix} = LU$, then

$L = \begin{bmatrix} 1 & 0 & 0 \\ d & 1 & 0 \\ g & h & 0 \end{bmatrix}$ and $U = \begin{bmatrix} a & b & c \\ 0 & e & f \\ 0 & 0 & i \end{bmatrix}$. LU factorization:

```
    For k = 1, ..., n                // iterate over all rows
        For i = k + 1, ..., n        // iterate each row i below row k
            mult := aik/akk          // determine row i's multi factor
            aik := mult              // store this factor
            For j = k + 1, ..., n    // iterate over all columns in the
                row
                aij := aij - mult * akj     // subtract the scaled data

// Note that the resulting factors are stored back into A for space
```

The runtime (number of FLOPs) for the above algorithm is $\dfrac{2n^3}{3} + O(n^2)$.

Forward Solve: Solving $Lz = b$ for z;

```
    For i = 1, ..., n
        zi := bi
        For j = 1, ..., i - 1
            zi := zi - lij * zj
```

Backward Solve: Solving $Ux = z$ for x;

```
    For i = n, ..., 1
        xi := zi
        For j = i + 1, ..., n
            xi := xi - uij * x
```

The runtime (number of FLOPs) for the above algorithm is $n^2 + O(n)$.

Transpose of a permutation matrix is the same as its inverse.

Norm of vectors and matrices are defined as

$$\|x\|_p = \left(\sum_{i=1}^{n} {x_i}^p\right)^{1/p} \qquad \text{and,} \qquad \|A\| = \max_{\|x\|\neq 0} \frac{\|Ax\|}{\|x\|}$$

and as a result, we have

$$\|A\|_1 = \underbrace{\max_j \sum_{i=1}^{n} |A_{ij}|}_{\text{(max absolute column sum)}} \qquad \|A\|_\infty = \underbrace{\max_i \sum_{j=1}^{n} |A_{ij}|}_{\text{(max absolute row sum)}}$$

Also remind you that

$$\|A\|_2 = \sqrt{\lambda_{max} A^* A}$$

Condition number of a matrix A is defined as

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

If $\kappa \approx 1$, then A is well-conditioned, else if $\kappa \gg 1$, then A is ill-conditioned.

Properties of condition numbers:

- $\kappa(A) \geq 1$
- $\kappa(A) = \kappa(A^T)$
- $\kappa(A) = \kappa(A^*)$
- $\kappa(A^{-1}) = \kappa(A)$
- $\kappa(AB) \leq \kappa(A) \cdot \kappa(B)$
- $\kappa(A) = 1 \iff A$ is orthogonal/unitary (for 2-norm)
- $\kappa(A) = \dfrac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$ (for 2-norm)

Some facts:

- $\det(A - \lambda I) = 0 \to$ eigenvalues;
- $tr(A) =$ sum of eigenvalues;
- $\det(A) =$ product of eigenvalues;
- A invertible $\iff$ 0 not an eigenvalue;
- A diagonalizable $\iff$ n lin. indep. eigenvectors;
- Symmetric $\Rightarrow$ diagonalizable w/ real eigenvalues;
- Triangular $\Rightarrow$ eigenvalues = diagonal entries;
- Permutation matrix is orthogonal, $P^T = P^{-1}$.