1 Floating Point Numbers

- **Taylor Series:** The Taylor Series of $f(x)$ evaluated at a is $\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$
- **Floating Point System:** The floating point system $\{\beta, t, L, U\}$ represents numbers of the form $\pm 0.1 d_1 d_2 \cdots d_t \cdot \beta^p$ for $L \le p \le U$. To go from a real number x to its floating point representation:
  1. Express x in base β.
  2. Normalize x by writing it in the form $\bar{x} \cdot \beta^p$ for $L \le p \le U$ so that $\bar{x}$ has a leading 0.
  3. Either round or truncate/chop $\bar{x}$ to t digits.
- **Absolute error:** $E_{\text{abs}} = |x_{\text{exact}} - x_{\text{approx}}|$.
- **Relative error:** $E_{\text{rel}} = \frac{|x_{\text{exact}} - x_{\text{approx}}|}{|x_{\text{exact}}}$. A result is roughly correct to s digits if $E_{\text{rel}} \approx 10^{-s}$.
- **Machine Epsilon:** The maximum relative error E, i.e., the smallest value such that $\text{fl}(1+E) > 1$. When rounding, $E = \frac{1}{2}\beta^{1-t}$, when truncating $E = \beta^{1-t}$. For any $x \in \mathbb{R}$, $\text{fl}(x) = x(1+\delta)$ for some $|\delta| \le E$. For any $w, z \in F$, we have $w \oplus z = \text{fl}(w+z) = (w+z)(1+\delta)$. Note floating point operations aren't associative.
- To minimize error, sum numbers of approximately same size and sign.

2 Polynomial Interpolation

- **Vandermonde Matrices:** To fit a polynomial to $(x_1, y_1), \ldots, (x_n, y_n)$, create an $n-1$ degree polynomial and solve the system mapping x to y in n coefficients.
- **Lagrange Bases:** For data $(x_1, y_1), \ldots, (x_n, y_n)$, a polynomial interpolant is given by $p(x) = y_1 L_1(x) + y_2 L_2(x) + \cdots + y_n L_n(x)$, where

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

  Note the kth entry is missing, and $L_i(x_j) = \delta_{i,j}$.
- **Hermite Interpolation:** For data $(x_1, y_1, s_1), \ldots, (x_n, y_n, s_n)$ where s_i is the slope at x_i, we have a piecewise cubic interpolation where the polynomial on the ith interval $[x_i, x_{i+1}]$ is given by $p_i(x) = a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3$ for

$$a_i = y_i, \quad b_i = s_i, \quad c_i = \frac{3y_i' - 2s_i - s_{i+1}}{\Delta x_i}, \quad d_i = \frac{s_{i+1} + s_i - 2y_i'}{(\Delta x_i)^2},$$

  where $\Delta x_i = x_{i+1} - x_i$ and $y_i' = \frac{y_{i+1}-y_i}{\Delta x_i}$.
- **Cubic Splines:** For data $(x_1, y_1), \ldots, (x_n, y_n)$, we fit a piecewise interpolation by enforcing C^2 continuity, i.e.,

$$S_i(x_i) = y_i \qquad S_i(x_{i+1}) = y_{i+1} \qquad S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}) \qquad S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$$

  To do this, we solve the system in s_i for $i = 2, \ldots, n-1$

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

  where $\Delta x_i = x_{i+1} - x_i$ and $y_i' = \frac{y_{i+1}-y_i}{\Delta x_i}$. We also need boundary conditions. Common choices are:
  - *Clamped:* $S'(x_1) = s_1$ and $S'(x_n) = s_n$ are specified.
  - *Free/Natural:* $S''(x_1) = 0$ and $S''(x_n) = 0$, done adding equations $s_1 + \frac{s_2}{2} = \frac{3}{2}y_1'$ and $\frac{s_{n-1}}{2} + s_n = \frac{3}{2}y_{n-1}'$.

  After solving for slope information s_i, plug into Hermite Interpolation formula.
- We can generalize this to parametric curves, usually using $t_i = i$ or an arc-length parameterization via $t_{i+1} = t_i + \|p_i - p_{i+1}\|_2$ and $t_1 = 0$ for points $p_1, \ldots, p_n$.

3 Ordinary Differential Equations

- **ODE:** Given an IVP $y'(t) = f(t, y(t))$ and $y(t_0) = y_0$, find y_i for some later point t_i. Generally $y_n \approx y(t_n)$ is our approximation.
- **Local (Truncation) Error:** It is given by $|y_{n+1} - y(t_{n+1})|$ assuming exact data $y_n = y(t_n)$. To compute it for y_{n+1} = RHS:
  1. Replace approximations in RHS with exact values.
     e.g., $y_{n-1} \to y(t_{n-1})$ and $f(t_n, y_n) \to y'(t_n)$.
  2. Taylor expand RHS quantities about time t_n.
  3. Taylor expand exact solution $y(t_{n+1})$.
  4. Compute $y(t_{n+1}) - y_{n+1}$, lowest degree non-canceling term is LTE.
- **Global/Absolute Error:** It is given by $|y_n - y(t_n)|$ (no exact data). Generally $(\textit{Global Error}) \le (\textit{Local Error}) \cdot O(h^{-1})$.
- **Higher Order ODEs:** To convert a higher order ODE of the form $y^{(n)}(t) = f(t, y(t), y'(t), \ldots, y^{(n-1)}(t))$, set $y_i = y^{(i-1)}$ for $i = 1, \ldots, n$ and then solve the system of ODEs given by $y_i' = y_{i+1}$ and $y_{n,i}' = f(t, y_{1,i}, y_{2,i}, \ldots, y_{n-1,i})$.
- **Stability:** To determine the stability condition of a scheme:
  1. Apply it to the test equation $y'(t) = -\lambda y(t)$ for $\lambda > 0$.
  2. Find the closed form solution of its error $\epsilon_n = y_n^{(p)} - y_n$ where $y_0^{(p)} = y_0 + \epsilon_0$.
  3. Find a condition (if any) on h such that $\lim_{n\to\infty} \epsilon_n = 0$

  Example: For FE we have $y_{n+1} = y_n + h(-\lambda y_n) = y_0(1-h\lambda)^n$. So $\epsilon_n = \epsilon_0(1-h\lambda)^n$ which goes to zero if and only if $|1-h\lambda| < 1$ or equivalently $h < \frac{2}{\lambda}$
- **Adaptive Time-stepping:** Follow the algorithm below
  1. Approximate solution with two schemes of different orders.
  2. Estimate the error by taking their difference.
  3. While error > tolerance, set $h \leftarrow h/2$ and recompute steps 1 and 2.
  4. Predict good next step size $h_{\text{new}} = h_{\text{old}} \left(\frac{\text{tolerance}}{|y_{n+1}^A - y_{n+1}^B|} \right)^{1/p}$ where p is the lower order of the schemes.
  5. Repeat until end time is reached.
- **Common Time-stepping Schemes:**

Name	Multi-Step	Implicit	LTE	Stability
Forward Euler			$O(h^2)$	$h < \frac{2}{\lambda}$
	$y_{n+1} = y_n + h \cdot f(t_n, y_n)$			
Trapezoidal Rule		✓	$O(h^3)$	unconditional
	$y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n) + f(t_{n+1}, y_{n+1}))$			
Improved Euler			$O(h^3)$	$h < \frac{2}{\lambda}$
	$y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n) + f(t_{n+1}, y^*_{n+1}))$ where $y^*_{n+1} = y_n + h \cdot f(t_n, y_n)$			
Backwards Euler		✓	$O(h^2)$	unconditional
	$y_{n+1} = y_n f(t_{n+1}, y_{n+1})$			
Midpoint Method			$O(h^3)$	
	$y_{n+1} = y_n + k_2$ where $k_1 = h \cdot f(t_n, y_n)$ and $k_2 = h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2})$			
BDF2	✓	✓	$O(h^3)$	

4 Ordinary Differential Equations

- **Orthongality Identities:**
  - $\int_0^{2\pi} \cos(kt)\sin(jt)dt = 0$ for all $j,k \in \mathbb{Z}$.
  - $\int_0^{2\pi} \cos(kt)\cos(jt)dt = 0 = \int_0^{2\pi} \sin(kt)\sin(jt)dt$ for $j \neq k$ and $j,k \in \mathbb{Z}$.
  - $\int_0^{2\pi} \cos(kt)dt = 0 = \int_0^{2\pi} \sin(kt)dt$ for all $j,k \in \mathbb{Z}$.
  - $\int_0^{2\pi} e^{ikt}e^{-i\ell t}dt = 2\pi\delta_{k,\ell}$ for all $k,\ell \in \mathbb{Z}$.
  - $\sum_{j=0}^{N-1} W^{jk}W^{-j\ell} = \sum_{j=0}^{N-1} W^{j(k-\ell)} = N\delta_{k,\ell}$.
- Using the above orthogonality identities we can solve for the Fourier transform $f(t) = a_0 + \sum_{k=1}^{\infty} a_k\cos(kt) + \sum_{k=1}^{\infty} b_k\sin(kt)$ of a function by integrating. We can transform this to the form $f(t) = \sum_{k=-\infty}^{\infty} c_k e^{ikt}$ by $c_k = \frac{a_k - ib_k}{2}$ and $c_{-k} = \frac{a_k+ib_k}{2}$ where $|c_0| = |a_0|$ and $|c_k| = |c_{-k}| = \frac{1}{2}\sqrt{a_k^2+b_k^2}$. Higher order terms thus represent higher frequency components.
- **Roots of Unity:** For a given N, $W = e^{2\pi i/N}$ is the Nth root of unity, i.e., $W^N = 1$. Note that $W^k = e^{2\pi ik/N} = \cos(2\pi k/N) + i\sin(2\pi k/N)$ thus there is a parallel to the unit circle.
- **Discrete Fourier Transform (DFT):**

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

  The sequence F_k is doubly infinite and N-periodic. I.e., $F_{k\pm N} = F_k$. If the data f_n is real, then $F_k = \overline{F_{N-k}}$. Note F_k represents the component of frequency k for $0 \leq k \leq \frac{N}{2}$ and of frequency $\frac{N}{2} - k$ for $\frac{N}{2} \leq k \leq N$ and F_0 represents the average of $\{f_n\}$.
- **Matrix DFT:** Where M is a matrix whose kth column is $\frac{1}{N}\begin{bmatrix} W^0 & W^{-k} & W^{-2k} & \cdots & W^{-(N-1)k}\end{bmatrix}^T$, we have $F = Mf$ (DFT) and $f = M^{-1}F = N\overline{M^T}F$ (IDFT).
- **Fast Fourier Transform (FFT):**

$$\text{If} \quad g_n = \frac{1}{2}(f_n + f_{n+\frac{N}{2}}) \quad \text{and} \quad h_n = \frac{1}{2}(f_n - f_{n+\frac{N}{2}})W^{-n} \quad \text{then} \quad F_{\text{even}} = G = \text{DFT}(g) \quad \text{and} \quad F_{\text{odd}} = H = \text{DFT}(h)$$

  Or, if we keep dividing all the way (remembering that each time we change our value of N), then by representing the indices our FFT coefficients in binary and reversing the bits, we unscramble our coefficients.
- **2D DFT:** For an image of size $M \times N$ and Nth and Mth roots of units W_N and W_M respectively,

$$F_{k,\ell} = \frac{1}{N}\sum_{n=0}^{N-1} W_N^{-nk}\left(\frac{1}{M}\sum_{j=0} f_{n,j}W_N^{-j\ell}\right)$$

  where k and j are the horizontal and vertical frequencies respectively. That is, we apply a DFT to each row, then to each column.

5 NLA and PageRank:

- **Random Surfer:** A surfer starts at some page and then follows links at random for k steps. Repeatedly visited pages are more important.
- **Google Matrix:** Let there be R pages in a directed graph and let $\deg(j)$ mean the **out**degree of a node. Let P be the matrix where $P_{i,j} = \frac{1}{\deg(j)}$ if there is a link from j to i and 0 otherwise (each column that isn't a dead end links should sum to 1). Let d be the vector where $d_i = 1$ if $\deg(i) = 0$ and 0 otherwise be the vector of dead ends. Let e be the all 1 vector. Then for some α (teleport probability), the Google matrix is $M = \alpha(P + \frac{1}{R}ed^T) + (1-\alpha)\frac{1}{R}ee^T$. This Google matrix is a Markov matrix.
- Given an initial probability vector (of which page to start on) p_0, we may compute our next probability vector (of pages) by $p_{n+1} = Mp_n$.
- **Markov Matrices:** Let Q be a Markov matrix. Q has 1 as an eigenvalue, and if Q is positive then $|\lambda| = 1$ has only one linearly independent eigenvector. Moreover, every eigenvalue of Q has $|\lambda| \leq 1$. The convergence rate of a positive Markov matrix is dictated by its second largest eigenvalue $|\lambda_2|$. For a Google matrix, $|\lambda_2| \approx \alpha$.
- **LU Factorization:** To solve $Ax = b$, take the augmented matrix $[A \mid b]$ and using row operations (subtractions), transform it to upper triangular form, the resulting matrix (excluding b') is U. Starting with $L = I_n$, whenever you perform the row subtraction $R_i = R_i - aR_j$ (for $j < i$), set $L_{ij} = a$. Note from the $A = LU$ factorized form we can solve $Ax = b$ by solving $Lz = b$ and then $Ux = z$.
- **LU Factorization with Pivoting:** To solve $PA = LU$ by pivoting (which has fewer numerical/floating-point errors) start with $P = I_n$. Then before every row subtraction to get U, first swap rows so that the topmost element of the current column is maximal. From $PA = LU$ we can solve $Ax = b$ by computing $b' = Pb$, then $Lz = b'$, then $Ux = z$
- **p-norms** $\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$ for $x \in \mathbb{R}^n$. For matrices $A \in \mathbb{R}^{n\times n}$ we have $\|A\|_p = \sup_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p}$. That said, $\|A\|_1$ is the maximum absolute column sum of A and $\|A\|_\infty$ is the maximum absolute row sum of A. Moreover, $\|A\|_2 = \max_i \sqrt{|\lambda_i|}$ if λ_i are the eigenvalues of A^TA. Finally, we have the following properties:
  - $\|x\| = 0 \iff x = 0$.
  - $\|\alpha x\| = |\alpha| \cdot \|x\|$ for $\alpha \in \mathbb{R}$.

  - $\|x+y\| \leq \|x\| + \|y\|$.
  - $\|Ax\| \leq \|A\| \cdot \|x\|$ for $x \in \mathbb{R}^n$.
  - $\|AB\| \leq \|A\| \cdot \|B\|$ for $B \in \mathbb{R}^{n \times n}$.
  - $\|I\| = 1$.

- **Linear Solving Conditioning:** For a given matrix norm, the condition number of a matrix A is $\kappa(A) = \|A\| \cdot \|A^{-1}\|$, if $\kappa(A) \approx 1$ then A is well-conditioned, if $\kappa(A) \gg 1$ then A is ill-conditioned. For a system $Ax = b$, we may bound the relative change in x with respect to the relative change in b or A or the residual $r = b - Ax_{\text{approx}}$ (note $r = 0$ iff $x_{\text{exact}} = x_{\text{approx}}$), respectively by

$$\frac{\|\Delta x\|}{\|x\|} \leq \kappa(A)\frac{\|\Delta b\|}{\|b\|} \qquad \text{and} \qquad \frac{\|\Delta x\|}{\|x + \Delta x\|} \leq \kappa(A)\frac{\|\Delta A\|}{\|A\|} \qquad \text{and} \qquad \frac{\|\Delta x\|}{\|x\|} \leq \kappa(A)\frac{\|r\|}{\|b\|}$$