Computer arithmetic & algorithms	$d_m d_{m-1}\ldots d_0.d_{-1}d_{-2}\ldots d_{-n} = d_m \times 10^m + d_{m-1} \times 10^{m-1} + \ldots + d_0 \times 10^0 + d_{-1} \times 10^{-1} + d_{-2} \times 10^{-2} + \ldots + d_{-n} \times 10^{-2}$ $b_m b_{m-1}\ldots b_0.b_{-1}\ldots b_{-n} = b_m \times 2^m + b_{m-1} \times 2^{m-1} + \ldots + b_0 \times 2^0 + b_{-1} \times 2^{-1} + \ldots + b_{-n} \times 2^{-n}$	base 10 (decimal), $d_m d_{m-1}\ldots d_0$: integer, $d_{-1}d_{-2}\ldots d_{-n}$: fractional converting binary to decimal
	$d_m d_{m-1}\ldots d_0.d_{-1}d_{-2}\ldots d_{-n} = (b_r b_{r-1}\ldots b_0.b_{-1}b_{-2}\ldots)_2$ $d_m d_{m-1}\ldots d_0 = (b_r b_{r-1}\ldots b_0)_2$ & $d_{-1}d_{-2}\ldots d_{-n} = (b_{-1}b_{-2}\ldots)_2$ To find integer part, - divide $d_m d_{m-1}\ldots d_0$ by 2 (remainder = b_0) - divide quotient by 2 (remainder = b_1)...repeat process until quotient is 0 For fractional part, - multiply $.d_{-1}d_{-2}\ldots d_{-n}$ by 2 (integer part = b_{-1}) - multiply new fractional part by 2 (integer part = b_{-2})...repeat until fractional part is 0; otherwise fractional part is infinite	converting decimal to binary integer = integer & fractional = fractional E.g. 2.4 = 10.011001100110... 2/2 = 1R0. 1/2 = 0R1 0.4 x 2 = 0 + 0.8 0.8 x 2 = 1 + 0.6 0.6 x 2 = 1 + 0.2 0.2 x 2 = 0 + 0.4 $0.01100110\ldots = 2^{-2} + 2^{-3} + 2^{-6} + 2^{-7} + \ldots =$ $(2^{-2} + 2^{-3})\sum_{i=0}^{\infty} 2^{-4i} = \frac{3}{8}\left(\frac{1}{1-2^{-4}}\right) = 0.4$
Floating-point formats	Binary num in scientific notation normalized form	$(b_r b_{r-1}\ldots b_0.b_{-1}\ldots b_{-n})_2 = \pm(1.b_{r-1}\ldots b_0 b_{-1}\ldots b_{-n})_2 \times 2^r$ ($0.00101_2 = 1.01_2 \times 2^{-3}$)
	$\pm(1.s_1 s_2\ldots s_N)_2 \times 2^k$ Double precision floating point: $\pm(1.b_1 b_2\ldots b_{52})_2 \times 2^k$	IEEE standard precision: double; Number of bits: sign 1, exponent 11, mantissa (N) 52
	machine epsilon = dist btw 1 and smallest floating point num > 1 and for double precision, $\epsilon_{mach} = 2^{-52}$	
Rounding rule	IEEE Rounding to Nearest Rule 1. 53rd bit = 0: truncate after 52nd bit 2. 53rd bit = 1: a. 54th bit onwards all 0 & 52nd bit = 0: truncate after 52nd bit b. else 1 added to 52nd bit	Applies to both normal and subnormal num $(1.*\ldots*_{52}0_{53})_2 \times 2^k = (1.*\ldots*_{52})_2 \times 2^k$ $(1.*\ldots*0_{52}1_{53})_2 \times 2^k = (1.*\ldots*0_{52})_2 \times 2^k$ $(1.*\ldots*1_{52}1_{53})_2 \times 2^k = (*.*\ldots*0_{52})_2 \times 2^k$ $(1.*\ldots**_{52}1_{53}\ldots*1*\ldots)_2 \times 2^k = (*.*\ldots**_{52})_2 \times 2^k$
Subnormal floating point	For num smaller than $2^{-1022} \approx 2.2 \times 10^{-308}$, subnormal floating point number is used: $\pm(0.s_1 s_2\ldots s_{52})_2 \times 2^{-1022}$	So, smallest +ve double precision num = $(0.00\ldots1)_2 \times 2^{-1022} = (1 \times 2^{-52}) \times 2^{-1022} = 2^{-1074}$
	Although num below ϵ_{mach} is machine representable, adding to 1 may have no effect	
Computer arithmetic	For a num x, fl(x) = num stored in computer (not exact)	Abs error = $\lvert p^*-p\rvert$. Relative error = $\frac{\lvert p*-p\rvert}{\lvert p\rvert} \le \frac{1}{2}\epsilon_{mach}$
	Use special symbol to represent computer arithmetic	$x \oplus y = fl(fl(x) + fl(y))$ $x \ominus y = fl(fl(x) - fl(y))$...
Matrix multi-plication	$A = (a_{ij})_{m \times n} \in \mathbb{R}^{m\times n}$, $B = (b_{ij})_{n \times p} \in \mathbb{R}^{n\times p}$, $C = AB = (c_{ij})_{m \times p} \in \mathbb{R}^{m\times p}$ $c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}$, ∀ i = 1,...,m, j = 1,...,p O(mpn). Exact is O(mpn + mp(n-1)) = O(mp(2n-1))	3 main qns: Find i, k s.t. $a_{ik} \neq 0$. Find k, j s.t. $b_{kj} \neq 0$. Find i, j s.t. $c_{ij} \neq 0$
Lower triangular multi-plication	Suppose A and B are lower triangular n x n matrix, C = AB also lower triangular $c_{ij} = 0$ ∀ i < j, $C = (c_{ij})_{n \times n}$ For i ≥ j: – A is lower triangular: $a_{ik} = 0$, ∀ i < k – B is lower triangular: $b_{kj} = 0$, ∀ k < j – $a_{ik}b_{kj} = 0$ if i < k or k < j. So multiplication only needed for k ≤ i or k ≥ j $c_{ij} = \sum_{k=j}^{i} a_{ik}b_{kj}$, 1 ≤ j ≤ i ≤ n	$\sum_{i=1}^{n} 1 = (n - 1) + 1$ $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ = num of terms * (1st term + last term) / 2 $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
Upper Hessen-berg multi-plication	A = $(a_{ij})_{n \times n}$ be tridiagonal matrix = $\begin{pmatrix} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & a_{32} & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & a_{n-1,n} \\ 0 & 0 & 0 & a_{n,n-1} & a_{n,n} \end{pmatrix}$, C = $(c_{ij})_{n \times n}$ $\begin{pmatrix} c_{11} & c_{12} & \cdots & \cdots & c_{1n} \\ c_{21} & c_{22} & c_{23} & \cdots & \vdots \\ 0 & c_{32} & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & c_{n-1,n} \\ 0 & 0 & 0 & c_{n,n-1} & c_{n,n} \end{pmatrix}$ B be upper triangular and C = AB be upper Hessenberg matrix $c_{ij} = 0$, ∀j < i - 1, 3 ≤ i ≤ n For j ≥ i - 1: – $a_{ik} = 0$, ∀k < i - 1, 3 ≤ i ≤ n or k > i + 1, 1 ≤ i ≤ n-2 – $b_{kj} = 0$, ∀k > j – $a_{ik}b_{kj} = 0$, ∀k < i - 1, 3 ≤ i ≤ n or k > min(i+1, j), 1 ≤ i ≤ n - 2 ≠ 0, ∀i - 1 ≤ k ≤ min(i+1,j), ∀i = 1,...,n, ∀j = max(1,i-1),..., n $c_{ij} = \sum_{k=max(1,i-1)}^{min(i+1,j)} a_{ik}b_{kj}$, ∀i = 1,...,n, ∀j = max(1,i-1),..., n	

Bisection Mtd	Intermediate Value Theorem (IVT): Let f be cts fn on [a,b], where f(a)f(b) < 0. Then ∃ r ∈ (a,b) s.t. f(r) = 0 Bisection mtd: Solve for eqn: f(r) = 0. Keep dividing interval by 2 until length of new interval/2 ≤ TOL. Soln: exact root r = approx root ± TOL After n bisection steps: Approximate root = midpoint of $(a_n, b_n) = (a_n + b_n)/2$ Error of approx soln = $\lvert$exact root - approx root$\rvert \le \frac{b_n-a_n}{2} = \frac{1}{2}\left(\frac{b-a}{2^n}\right) = \left(\frac{b-a}{2^{n+1}}\right)$ Num of fn evaluations = n + 2 (f(a), f(b), n times of f(c)). Convergence rate = 1/2; need predefine initial interval. Convergence guaranteed	
	Soln is correct within p d.p if error is less than 0.5×10^{-p}. Num of steps for bisection: $\frac{b-a}{2^{n+1}} < 0.5 \times 10^{-p}$	
Fixed-Point Iteration (FPI)	Solve eqn: g(x) = x, by iterating $x_{i+1} = g(x_i)$. r is a fixed pt if g(r) = r Suppose f(x) = g(x) - x, and x_a is approximation for r where f(r) = 0	Backward error = $\lvert g(x_i+1) - x_{i+1}\rvert = \lvert f(x_a)\rvert$. Forward error = $\lvert r-x_a\rvert$ Method relies on Fixed-Point Thrm
	Convergence Thrm: If $\lvert g'(r)\rvert < 1$: will converge with rate $S = \lvert g'(r)\rvert$ So S = 0: fastest convergence rate. S < 1: cfm converge. S > 1: won't converge. S = 1: may or may not	
Horner's Mtd	Horner's Mtd: most optimal mtd for finding value of polynomial. O(m). Given x, evaluate h(x). Common that h is polynomial. $P_m(x) = a_o + a_1x + \ldots + a_{m-1}x^{m-1} + a_mx^m = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{m-1} + xa_m)))$ $p_m = a_m$. $p_{m-1} = a_{m-1} + xp_m$. ... $p_1 = a_1 + xp_2$. $p_0 = a_0 + xp_1$	

Gaussian Elimina-tion $O(n^3)$ Leading term: $\frac{2n^3}{3}$	$A_{n \times n}x_{n \times 1} = b_{n \times 1}$. Note lower triangular part not set to zero as they would not be used in further computation Subtraction: $\frac{n(n-1)(2n+5)}{6}$ (without labelling rows). Multiplication: $\frac{n(n-1)(2n+5)}{6}$ (without labelling rows) Division: n(n+1)/2 (without labelling rows). Note zero column = non-invertible Could have cases where $\mathbf{a_{r_i i}} = 0$. So label rows to keep track of swapping rows Due to computer arithmetic errors, num supposed to be 0 becomes a very small num and cause loss of significant digits Backward sub: $O(n^2)$
Partial Pivoting	1st mtd to perform row swap to minimise errors Select pivot elem s.t. its absolute value is largest in a particular col. Num of comparisons is $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} 1 = \sum_{i=1}^{n-1}(n-i) = \frac{n(n-1)}{2}$

	However, if remaining entries of pivot row also has large magnitude, then would cause loss of significant digits in the other rows.
Relative absolute ratio	A way to do this is to compare the relative absolute sizes in each col j = 1,...,n , relative absolute value of row i = $\frac{\lvert a_{ij}\rvert}{\lvert max(row\ i)\rvert}$. Divisions = $\frac{(n+2)(n-1)}{2}$, Comparisons = $\frac{n(n-1)(2n+5)}{6}$ By choosing largest relative absolute size for each col as pivot elem, loss of sig digits is minimised
Scaled Partial Pivoting (SPP)	2nd mtd to perform row swap to minimise errors. ≤ : don't swap. > : swap To save computational cost, assume max entry of row does not change too much in elimination process, and only find max for each row once at beginning, then use this max from original row to calculate relative absolute ratio: scaled partial pivoting Comparisons = $\frac{3}{2}n(n-1)$. Note max(row i) stay fixed even if row swap is performed
LU factorization $O(n^3)$ + $O(2pn^2)$	Solve for multiple L.S with same coefficient matrix, $Ax_1 = b_1$, $Ax_2 = b_2$..., $Ax_p = b_p$. So can preprocess A to not repeat ops. Find A = LU by Gaussian elimination w/o pivoting strategies, where L is lower triangular, U: upper triangular. LUx = b. L = strictly lower part of processed A (multipliers m_{ji}) + diagonal all 1. U = upper triangular of processed A Forward substitution (solve Ly = b for y). Backward substitution (solve Ux = y for x) Time complexity for both substitution = $O(n^2)$. So, solving p LS = $O(2pn^3)$
PA = LU factorization $O(n^3)$ + $O(2pn^2)$	Matrix which require row swap to get REF cannot be LU factorized. (or can check det of top left entry, top left 2 x 2 entries, top left 3 x 3 ... ≠ 0 then can be factorized). Hence, need perform row swap at start with SPP, PAx = Pb, where P is a permutation matrix (n x n matrix consisting all 0, except for a single 1 in every row and col) Now PA = LU. L = strictly lower part of processed A (multipliers m_{ji}) + diagonal all 1. U = upper triangular of processed A Solve Ly = Pb, then solve Ux = y. No need to explicitly find P, just output r (stored row index) to replace P
A = LU & PA = LU	If n is large, memory to store L and U might be consuming
Special matrix	Symmetric positive-definite matrix. n x n matrix A is symmetric if $A^T = A$. A is positive-definite if $x^TAx > 0$ ∀ col vector x ≠ 0. To check if matrix is positive-definite, could expand algebraically with x and then complete the sq to check > 0. OR If A is symmetric, A is positive-definite iff all eigenvalues > 0 Principal submatrix of sq matrix A is a sq submatrix whose diag entries = diag entries of A. (a_{11}, a_{22}, a_{33}, $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, $\begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix}$, $\begin{pmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{pmatrix}$) Any principal submatrix of a symmetric positive-definite matrix is symmetric positive-definite.
Cholesky factorization $O(n^3)$ + $O(pn^2)$	Cholesky factorization: every symmetric positive-definite matrix A can be factored as $A = R^TR$ (R is upper triangular). Hence would save roughly half the memory compared to A = LU. Idea: Use row/col ops to reduce A into identity matrix to get $A = R^TR$. Forward sub: $R^Ty = b$ for y. Backward sub: Rx = y for x. $O(n^2)$ A = $\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{pmatrix}$ k = 1: R = $\begin{pmatrix} \sqrt{a_{11}} & u_1^T := \frac{1}{\sqrt{a_{11}}}(a_{12} \dots a_{1n}) \\ - & - \end{pmatrix}$, A = $\begin{pmatrix} - & - \\ - & \widetilde{A_1} := \begin{pmatrix} a_{22} & \cdots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{2n} & \cdots & a_{nn} \end{pmatrix} - u_1u_1^T \end{pmatrix}$, k = 2: R = $\begin{pmatrix} R_{11} & R_{12} & R_{13} \dots R_{1n} \\ - & \sqrt{\tilde{a}_{11}} & u_2^T := \frac{1}{\sqrt{\tilde{a}_{11}}}(\tilde{a}_{12} \dots \tilde{a}_{1,n-1}) \\ - & - & - \end{pmatrix}$, A = $\begin{pmatrix} - & - \\ - & \widetilde{A_2} := \begin{pmatrix} \tilde{a}_{22} & \cdots & \tilde{a}_{2,n-1} \\ \vdots & \ddots & \vdots \\ \tilde{a}_{2,n-1} & \cdots & \tilde{a}_{n-1,n-1} \end{pmatrix} - u_2u_2^T \end{pmatrix}$ Repeat until k = n, then A becomes I so $A = R^TR$. Note $\tilde{A} = K_1 - u_1u_1^T$ also positive definite.
Strictly Diagonally Dominant	A is stricly diagonally dominant (sdd) if for each 1 ≤ i ≤ n, $\lvert a_{ii}\rvert > \sum_{j\neq i}\lvert a_{ij}\rvert$, i.e. diagonal entry > sum of non-diagonal entries in same row, then Jacobi and Gauss-Seidel mtd will converge If A sdd, then A is a nonsingular matrix. If A not sdd, MIGHT still converge (check spectral radius)
Spectral Radius	Another mtd to check convergence is spectral radius p(B) = max magnitude of eigenvalues of B. If p(B) < 1, and c is arbitrary, then for any vector x_0, $x_{k+1} = Bx_k + c$ converges. In particular, check $p(D^{-1}(L + U)) < 1$. Use $\det(D^{-1}(L + U) - \lambda I) = 0$ to find eigenvalues. Note determinant of matrix = product of eigenvalues
Jacobi Method $O(n^2)$	$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix} + \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}$ = L + D + U Similar to fixed-point iteration. $A_{n\times n}$ = L + D + U. Ax = b. (L + D + U)x = b. Dx = b - (L + U)x. x = D^{-1}(b- (L + U)x). D^{-1} is just reciprocal of all entries in D. So $x^{(k+1)} = D^{-1}(b - Lx^{(k)} - Ux^{(k)})$ Then solve eqn element-wise, ∀i = 1,...,n: $x_i^{(k+1)} = \frac{1}{a_{ii}}\left[b_i - a_{i1}x_1^{(k)} - \cdots - a_{i,i-1}x_{i-1}^{(k)} - a_{i,i+1}x_{i+1}^{(k)} - \cdots - a_{in}x_n^{(k)}\right]$ So calculation can be parallelized
Gauss-Seidel Mtd $O(n^2)$	Similar to Jacobi. Now, $x^{(k+1)} = D^{-1}(b - Lx^{(k+1)} - Ux^{(k)})$ ∀i = 1,...,n: $x_i^{(k+1)} = \frac{1}{a_{ii}}\left[b_i - a_{i1}x_1^{(k+1)} - \cdots - a_{i,i-1}x_{i-1}^{(k+1)} - a_{i,i+1}x_{i+1}^{(k)} - \cdots - a_{in}x_n^{(k)}\right]$ Now cannot parallelized, as x_2,... x_n dependent on x_1, ..., x_{n-1}. But since updated values are used, will converge faster than Jacobi
Successive Over-Relaxation (SOR) Mtd $O(n^2)$	Let ω be a real num, and $x^{(k+1)} = (1 - \omega)x^{(k)} + \omega D^{-1}(b - Lx^{(k+1)} - Ux^{(k)})$. ω is called relaxation parameter and ω>1 = over-relaxation ∀i = 1,...,n: $x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left[b_i - a_{i1}x_1^{(k+1)} - \cdots - a_{i,i-1}x_{i-1}^{(k+1)} - a_{i,i+1}x_{i+1}^{(k)} - \cdots - a_{in}x_n^{(k)}\right]$ GS: ω = 1. SOR: ω > 1. Need to choose ω wisely, usually 1.1 or 1.2. Faster convergence than GS
When to use which mtd?	Direct mtd: Gaussian elimination, A = LU or PA = LU, Cholesky factorization. $O(n^3)$ for preprocessing, $O(n^2)$ for finding Iterative mtd: Jacobi, Gauss-Seidel, SOR. $O(n^2)$ Use iterative mtd if 1. requirement of accuracy not high, save computational cost 2. good approximation already known (to be used as initial guess) 3. If A is sparse (many entries = 0). Most expensive op is matrix-vector multiplication which would be cheaper

Interpolation	Given data points (x_0, $f(x_0)$), (x_1, $f(x_1)$), ..., (x_n, $f(x_n)$). Want to find polynomial of deg n, $P_n(x)$ to connect these pts to restore original fn f, i.e. $P_n(x_i) = f(x_i)$, ∀i = 0,1,...,n	Weierstrass approximation theorem: Let f be a cts fn on [a, b]. For any ϵ > 0, ∃ a polynomial P(x) s.t. $\lvert f(x) - P(x)\rvert < \epsilon$, ∀ x ∈ [a, b]
1. Gaussian Elmt	Write $P_n(x) = a_0 + a_1x + \dots + a_nx^n$ From general eqn for polynomial, sub data points in to get n+1 eqns and solve L.S to find coeff	
Basis fn	Basis fn: {$\varphi_0(x)$, $\varphi_1(x)$, ..., $\varphi_n(x)$}, where $\varphi_k(x_j) = \delta_{jk}$,	Kronecker delta, $\delta_{jk} = \begin{cases} 0, if\ j \neq k \\ 1, otherwise \end{cases}$, j = 0,1,..., n, k = 0,1, ..., n
2. Lagrange polynomial	$L_k(x) = \prod_{j=0,j\neq k}^{n}\frac{x-x_j}{x_k-x_j} = \frac{(x-x_0)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)}$. So $L_k(x_j) = \delta_{jk}$. $L_k(x)$ = k^{th} Lagrange basis polynomial Then $P_n(x) = f(x_0)L_0(x) + f(x_1)L_1(x) + \dots + f(x_n)L_n(x)$. $P_n(x)$ = Lagrange interpolating polynomial of deg n	

	LS is guaranteed to have a soln if (deg n, num of eqns m) m=n: By constructing Lagrange interpolating polynomial m > n: Not guaranteed m < n: Infinitely many solution	Using LS: Pros – direct mtd | Cons: tedious computations Lagrange: Pros – easy to analyse. Good if we need to interpolate many fns with same set of interpolating nodes | Cons: not convenient to add more data points
Uniqueness of Lagrange polynomial	If $x_0, x_1, \dots, x_n$ are n + 1 distinct nums and f is a fn whose values are given at these nums, then a unique polynomial $P_n(x)$ of degree at most n exists with $f(x_k) = P_n(x_k)$, for k = 0,1,...,n. $P_n(x)$ = Lagrange polynomial	
Adding more inter-polating nodes	Let P_{n-1} be the Lagrange interpolating polynomial of f(x) with n nodes $x_0, x_1, \dots, x_{n-1}$. Suppose we get one more data point $(x_n, f(x_n))$. Let $Q_n(x) = P_n(x) - P_{n-1}(x)$. $Q_n(x)$ is the unique interpolating polynomial that interpolates $(x_0, 0), (x_1, 0), \dots, (x_{n-1}, 0), (x_n, f(x_n - P_{n-1}(x_n))$ $Q_n(x) = f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1)\dots(x - x_{n-1})$ where $f[x_0, x_1, \dots, x_n]$ = n^{th} divided diff of f = $\sum_{k=0}^{n} f(x_k)\prod_{j=0, j\neq k}^{n}\frac{1}{x_k - x_j}$ So $P_n(x) = P_{n-1}(x) + Q_n(x) = P_{n-1}(x) + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1)\dots(x - x_{n-1}) = \dots = P_0(x) + f[x_0, x_1](x - x_0) + \dots + f[x_0, \dots, x_n](x - x_0)(x - x_1)\dots(x - x_{n-1})$	
3. Newton's Poly-nomial (Easier to add more nodes)	$P_n(x) = \sum_{k=0}^{n} f[x_0, \dots, x_k](x - x_0)(x - x_1)\dots(x - x_{k-1}) = P_0(x) + f[x_0, x_1](x - x_0) + \dots + f[x_0, \dots, x_n](x - x_0)(x - x_1)\dots(x - x_{n-1})$ And n^{th} divided diff of f = $f[x_0, x_1, \dots, x_n] = \sum_{k=0}^{n} f(x_k)\prod_{j=0, j\neq k}^{n}\frac{1}{x_k - x_j}$. where $f[x_0] = P_0(x) = f(x_0)$. Note. order of nodes don't matter. i.e. $f[x_0, x_1, x_2] = f[x_1, x_2, x_0]$ Let $x_0, \dots, x_n$ be n+1 distinct real nums. Then $f[x_0, x_1, \dots, x_n] = \frac{f[x_1, x_2, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}]}{x_n - x_0}$. Compute $f[x_0], \dots, f[x_n]$ first, then $f[x_0, x_1] = \frac{f[x_1] - f[x_0]}{x_1 - x_0}$... $O(n^2)$ for computing n^{th} divided diff. Num of entries = n + (n-1) + ... + 1 = $\frac{n(n+1)}{2}$. So total need n(n+1) subtraction and $\frac{n(n+1)}{2}$ divisions To compute $P_n(x)$ for some x, use Horner's method. O(n) $P \leftarrow f[x_0, x_1, \dots, x_n]$; for k = n-1,...,0 do {$P \leftarrow f[x_0, x_1, \dots, x_k] + (x - x_k)P$}	
Error of interpola-tion	Runge's phenomenon: Wider oscillation at ends (i.e. worse interpolation at ends). ↑ num of nodes only worsen approximation at ends. Error of interpolation, $\lvert f(x) - P_n(x)\rvert$: error is 0 on all nodes. $g_n(x) = \lvert(x-x_0)(x-x_1)\dots(x-x_n)\rvert = \lvert\prod_{k=0}^{n}(x - x_k)\rvert$ has similar pattern to error $\lvert f(x) - P_n(x)\rvert$ w equally-spaced nodes Let $x_0 < x_1 < \dots < x_n$ be n+1 distinct pts on [a, b]. If $f \in C^{n+1}([a,b])$, i.e. all derivatives $f, f^{(1)}, \dots, f^{(n+1)}$ are cts in [a, b], and P_n is the interpolating polynomial of f w deg ≤ n at $x_0, x_1, \dots, x_n$. Then $\forall x \in \mathbb{R}, \exists \xi \in \mathbb{R}$ dependant on $x_0, \dots, x_n$, x, and $\xi \in$ min and max of $\{x_0, x_1, \dots, x_n, x\}$, s.t. $f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)(x-x_1)\dots(x-x_n)$	
Cheby-shev Interpola-tion	Chebyshev nodes: $x_k = \cos\left(\frac{(k+1/2)\pi}{n+1}\right)$, k = 0,1,..., n in [-1, 1]. Let $T_n(x)$:= (deg n, n order) Chebyshev polynomial = cos((n)arccos x) (n nodes) Using Chebyshev nodes means $\lvert\prod_{k=0}^{n}(x - x_k)\rvert = \frac{1}{2^n}T_{n+1}(x) \le \frac{1}{2^n}$ is the smallest. So now $g_n(x) = \lvert\frac{1}{2^n}\cos((n+1)\arccos x)\rvert \le \frac{1}{2^n}$. $g_n \to 0$ as $n \to \infty$. Error of interpolation for Chebyshev = $f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\frac{1}{2^n}T_{n+1}(x) \le \frac{f^{(n+1)}(\xi)}{(n+1)!}\frac{1}{2^n}$ For [a,b]: $x_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{(k+1/2)\pi}{n+1}\right)$, k = 0,1,..., n. $\lvert\prod_{k=0}^{n}(x - x_k)\rvert \le \frac{((b-a)/2)^{n+1}}{2^n}$ By using Chebyshev nodes to interpolate w Lagrange/divided diff, this polynomial = Chebyshev interpolating polynomial w deg n-1 Using Chebyshev nodes, error is worse ard center, but much milder near ends (error more evenly distributed). As n ↑, Chebyshev → f(x)	

Linear Least Square Problem	Inconsistent sys: SLE w no solution, typically m ≥ n Length = Euclidean norm = $\lVert x\rVert_2 = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$ Dot product of 2 n-dimensional col vectors = $u \cdot v = u^Tv = u_1v_1 + u_2v_2 + \dots + u_nv_n$ If $u^Tv = 0$, then u and v are perpendicular/orthogonal to each other. $u \perp v$	Note $x^Tx = \lVert x\rVert^2$. Normal equations: $A^TA\bar{x} = A^Tb$ $\bar{x}$ is L.S sol to Ax = b, which minimizes Euclidean norm of the residual r = b - Ax If r = 0, then $\bar{x}$ is the sol to Ax = b.
	If need to use $A_{m \times n}$ (full col rank = cols all LI) for multiple inconsistent sys, can use Cholesky factorization for $A^TA = R^TR$, where R is an upper triangular matrix	A^TA is symmetric positive-definite. For all x ≠ 0, $x^TA^TAx = (Ax)^T(Ax) = y^TY = \lVert y\rVert^2 > 0$ Forward sub: $R^Ty = A^Tb$. Backward sub: $R\bar{x} = y$
QR factorization	A^TA not numerically stable as could have rounding errors. A = QR: (m x n) = (m x n)*(n x n) So, use (reduced) QR factorization: $A_{m \times n} = Q_{m \times n}R_{n \times n}$, where Q is an orthogonal matrix ($Q^T = Q^{-1}$), R is upper triangular. Then $Rx = Q^Tb$. Since Q is orthogonal matrix, it has orthonormal cols (i.e. $\lVert\mathbf{q}_i\rVert = 1$, $\mathbf{q}_i^T\mathbf{q}_j = 0$ if i ≠ j) $\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & \cdots & q_{mn} \end{pmatrix}\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{nn} \end{pmatrix}$ $(\boldsymbol{a}_1 \quad \boldsymbol{a}_2 \quad \cdots \quad \boldsymbol{a}_n) = (r_{11}\boldsymbol{q}_1 \quad r_{12}\boldsymbol{q}_1 + r_{22}\boldsymbol{q}_2 \quad \cdots \quad r_{1n}\boldsymbol{q}_1 + \dots + r_{nn}\boldsymbol{q}_n)$	
Gram-Schmidt orthogonalization	$\mathbf{y}_1 = \mathbf{a}_1$, $r_{11} = \lVert\mathbf{y}_1\rVert$, $\mathbf{q}_1 = \frac{y_1}{r_{11}}$ $r_{12} = \mathbf{q}_1^T\mathbf{a}_2$, $\mathbf{y}_2 = \mathbf{a}_2 - r_{12}\mathbf{q}_1$, $r_{22} = \lVert\mathbf{y}_2\rVert$, $\mathbf{q}_2 = \frac{y_2}{r_{22}}$	$\mathbf{y}_j = \mathbf{a}_j - r_{1j}\mathbf{q}_1 - r_{2j}\mathbf{q}_2 - \dots - r_{j-1,j}\mathbf{q}_{j-1}$, $\mathbf{q}_j = \frac{y_j}{r_{jj}}$ $r_{ij} = \mathbf{q}_i^T\mathbf{a}_j$, $r_{jj} = \lVert y_j\rVert$,

Taylor's Thrm	Suppose $f:\mathbb{R} \to \mathbb{R}$ is n+1 times differentiable on some open interval w the n^{th} derivative $f^{(n)}$ cts on [a, x]. Then $\exists c \in [a,x]$ s.t. $f(x) = f(a) + f'(a)(x-a) + \frac{1}{2!}f''(a)(x-a)^2 + \dots + \frac{1}{n!}f^{(n)}(a)(x-a)^n + \frac{1}{(n+1)!}f^{(n+1)}(c)(x-a)^{n+1}$ = P(x) + approximation error The n^{th} order Taylor polynomial for f at a is $P(x) = f(a) + f'(a)(x-a) + \frac{1}{2!}f''(a)(x-a)^2 + \dots + \frac{1}{n!}f^{(n)}(a)(x-a)^n$, w approximation error = $\frac{1}{(n+1)!}f^{(n+1)}(c)(x-a)^{n+1}$
2-point forward-diff formula	If f is twice continuously differentiable, then by Taylor's thrm, let x = x+h, a = x, then $f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(c)$ where $c \in [x, x+h]$ (w error term). 1st order method since error $\frac{h}{2}f''(c)$ is O(h)
Generalized IVT	Let f be a cts fn on interval [a,b]. Let $x_1, \dots, x_n \in [a,b]$ and $a_1, \dots, a_n > 0$. Then $\exists c \in [a,b]$ s.t. $(a_1 + \dots + a_n)f(c) = a_1f(x_1) + \dots + a_nf(x_n)$
3-point centered-diff formula	Expand f(x+h) and f(x-h) to f''' w Taylor to get $f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{6}h^2f'''(c)$, where x-h < c < x+h 2nd order method since error $\frac{1}{6}h^2f'''(c)$ is $O(h^2)$, approx is better. Generally, higher order appox formula more accurate
Approx formula for higher derivatives	For f''(x), $f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f^{(4)}(c_1)$ & $f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f^{(4)}(c_2)$ where $x-h < c_2 < x < c_1 < x+h$ Adding both, 3-point centered diff formula for $f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$ w error $\frac{h^2}{12}f^{(4)}(c)$ where x-h < c < x+h
Rounding error	Let $\hat{f}(x+h)$, $\hat{f}(x-h)$ be floating-point version of f(x+h), f(x-h), i.e. $f(x+h) = \hat{f}(x+h) + \epsilon_1$, $f(x-h) = \hat{f}(x-h) + \epsilon_2$, for some machine rounding error ϵ_1, ϵ_2. Then error in approx for 3-point centered-diff = $\lvert f'(x) - \hat{f}'(x)\rvert \le \frac{\epsilon}{h} + \frac{h^2M}{6}$, where $\epsilon_1, \epsilon_2 < \epsilon > 0$ and $\lvert f'''(c)\rvert \le M$. Smallest error at $h = \sqrt[3]{\frac{3\epsilon}{M}}$
Extrapolation	order n formula for approximating Q: $Q = F_n(h) + K(h)h^n$, where K(h) depends on h but can be treated as constant over range of h (Richardson) extrapolation: $F_{n+1}(h) + O(h^{n+1}) = \frac{2^nF_n(h/2) - F_n(h)}{2^n - 1} + O(h^{n+1})$, where $F_{n+1}(h)$ is at least an order n+1 formula

MVT for integrals	Let f be cts fn and g be an integrable fn that does not change sign on [a,b]. Then ∃ c ∈ [a,b] s.t. $\int_a^b f(x)g(x)\,dx$ = f(c) $\int_a^b g(x)\,dx$
Newton-Cotes (NC) approach	Trapezoid Rule: $\int_{x_0}^{x_1} f(x)\,dx = \frac{h}{2}(y_0 + y_1) - \frac{h^3}{12}f''(c)$:= approx + error where $h = x_1 - x_0$ and $c \in [x_0, x_1]$ Simpson's Rule: $\int_{x_0}^{x_2} f(x)\,dx = \frac{h}{3}(y_0 + 4y_1 + y_2) - \frac{h^5}{90}f^{(4)}(c)$:= approx + error where $h = x_1 - x_0 = x_2 - x_1$ and $c \in [x_0, x_2]$
Deg of precision	Deg of precision of a numerical integration mtd is the greatest int k for which all deg k or less polynomials are integrated exactly by the mtd. Deg of precision of Trapezoid rule = 1. Deg of precision of Simpson Rule = 3.
Composite NC Formulas	Composite Trapezoid Rule: $\int_a^b f(x)\,dx = \frac{h}{2}(y_0 + 2\sum_{i=1}^{m-1} y_i + y_m) - \frac{(b-a)h^2}{12}f''(c)$ where $h = \frac{b-a}{m}$ and c ∈ [a,b]. 2nd order mtd since $O(h^2)$. Composite Simpson's Rule: $\int_a^b f(x)\,dx = \frac{h}{3}(y_0 + 4\sum_{i=1}^{m} y_{2i-1} + 2\sum_{i=1}^{m-1} y_{2i} + y_{2m}) - \frac{(b-a)h^4}{180}f^{(4)}(c)$ where $h = \frac{b-a}{2m}$ and c ∈ [a,b]. 4th order mtd
Open Newton-Cotes Mtd	Use if fn not valid on endpoints. Applicable for fn whose f'' is cts on [a,b] Midpoint Rule: $\int_{x_0}^{x_1} f(x)\,dx = hf(w) + \frac{h^3}{24}f''(c)$ where $h = (x_1 - x_0)$, w is midpoint := $x_0 + h/2$ and $c \in [x_0,x_1]$. Composite Midpoint Rule: $\int_a^b f(x)\,dx = h\sum_{i=1}^{m} f(w_i) + \frac{(b-a)h^2}{24}f''(c)$ where $h = \frac{b-a}{m}$ and c ∈ [a,b]. 2nd order mtd since $O(h^2)$
Romberg integration	So $F_{2k}(h) := \frac{2^{2(k-1)}F_{2(k-1)}(h/2) - F_{2(k-1)}(h)}{2^{2(k-1)}-1}$ diff h $\begin{array}{ccccc} F_2(h) & & & & \\ F_2(h/2) & F_4(h) & & & \\ F_2(h/4) & F_4(h/2) & F_6(h) & & \\ F_2(h/8) & F_4(h/4) & F_6(h/2) & F_8(h) & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$ $= \frac{2^2 F_2(\frac{h}{2}) - F_2(h)}{2^2-1}$ Romberg triangle: Comp. Trap. rule $\begin{array}{ccccc} R_{11} & & & & \\ R_{21} & R_{22} & & & \\ R_{31} & R_{32} & R_{33} & & \\ R_{41} & R_{42} & R_{43} & R_{44} & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$ where $R_{jk} = \frac{2^{2(k-1)}R_{j,k-1} - R_{j-1,k-1}}{2^{2(k-1)}-1}$ R_{22} : 4nd order R_{33} : 6th order R_{44} : 8th order. R_{11}, R_{22}, ... are just normal composite trap rule w 1,2,... panels Applying Composite Trapezoid Rule with 1,2,4, ... panels $h_1 = b - a \quad R_{11} = \frac{h_1}{2}[f(a) + f(b)]$ $h_2 = h_1/2 \quad R_{21} = \frac{1}{2}R_{11} + h_2 f(a + h_2)$ $h_3 = h_2/2$ $R_{31} = \frac{1}{2}R_{21} + h_3[f(a + h_3) + f(a + 3h_3)]$ $\begin{array}{cccc} R_{11} & & & \\ R_{21} & R_{22} & & \\ R_{31} & R_{32} & R_{33} & \\ \vdots & \vdots & \vdots & \ddots \end{array}$ $R_{i2} = \frac{4R_{i1} - R_{i-1,1}}{4-1} \quad R_{i3} = \frac{16R_{i2} - R_{i-1,2}}{16-1}$ Applying extrapolation to the previous column