DATA LINEARIZATION (least squares line) ln y = Ax + ln c by log of both sides, which we write as z = Ax + B

(x[k], y[k])	(0, 1.5)	(1, 2.5)	(2, 3.5)	(3, 5.0)	(4, 7.5)	10
z[k]	0.4055	0.9163	1.2528	1.6094	2.0149	6.1989
x[k]^2	0	1	4	9	16	30
x[k]z[k]	0	0.9113	2.5036	4.8282	8.0596	16.3097

A(30) + B(10) = 16.3097 A = .3912, B = .4574, so normal Least Squares Line would be y = .3912x + .4574

A(10) + B(5) = 6.1989 De-linearize via y = (e^B)e^(Ax) = 1.5799e^(0.3912x)

POLYNOMIAL FITTING (Ax^2 + Bx + C) A*sum[k=1;n]x[k]^4+B*sum[k=1;n]x[k]^3+C*sum[k=1;n]x[k]^2 = sum[k=1;n]y[k]x[k]^2
A*sum[k=1;n]x[k]^3+B*sum[k=1;n]x[k]^2+C*sum[k=1;n]x[k] = sum[k=1;n]y[k]x[k]
A*sum[k=1;n]x[k]^2+B*sum[k=1;n]x[k]+C = sum[k=1;n]y[k]

CUBIC SPLINES- S[k](x[k]) = y[k], S[k](x[k+1]) = S[k+1](x[k+1]), S'[k](x[k+1]) = S'[k+1](x[k+1]), S"[k](x[k+1]) = S"[k+1](x[k+1])
Extrapolated spline = S[0] and S[1] are the same, S[n-2] and S[n-1] are the same

k	x[k]	y[k]	h[0] = h[1] = h[2] = π/6 = 0.5236
0	0	0	d[0] = (y[1]-y[0])/h[0] = 0.9549
1	π/6 = 0.3236	0.5	u[1] = 6(d[1]-d[0]) = -1.5354
2	π/3 = 1.0472	0.8660	m[0] = 0, m[3] = 0 (from end-point constraints)
3	π/2 = 1.5708	1	h[n-2]m[n-2]+2(h[n-2]+h[n-1])m[n-1]=u[n-1]

S[0,0] = 0, S[0,1] = 0.9936, S[0,2] = 0, S[0,3] = -0.1412, S[0](x) = 0.9936 x - 0.1412 x^3

CENTERED FORMULAS O(h^2): f'(x) ≈ (f(x+h)-f(x-h))/2h, O(h^4): f'(x) ≈ (-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h))/12

NUMERICAL INTEGRATION Trapezoidal: A[1] = ((f[0]+f[1])/2)(Δx)
Simpson: P[2](x) = y[0](((x-x[1])(x-x[2]))/((x[0]-x[1])(x[0]-x[2]))) + y[1](((x-x[0])(x-x[2]))/((x[1]-x[0])(x[1]-x[2])))
+ y[2](((x-x[0])(x-x[1]))/((x[2]-x[0])(x[2]-x[1])))
∫[x[0];x[2]] P[2](x) = y[0]_x/2 * ∫[0;2] (t^2-3t+2)dt - y[1]_x * ∫[0;2] (t^2-2t)dt + y[2]_x/2 * ∫[0;2] (t^2-t)dt = y[0]_x/3 + 4y[1]_x/3 + y[2]_x/3

NUMERICAL OPTIMIZATION f'(x) = 0 indicates local max or minimum (f''(x) < 0 indicates max, otherwise min)

GOLDEN RATIO Divide the interval [a,b] so that a < c < d < b, c = a + (1-r)(b-a), d = a + r(b-a), r = (√5 - 1)/2

STEEPEST DESCENT (gradient) Vector function (∂f(x)/∂x[1], ∂f(x)/∂x[2], ... ∂f(x)/∂x[n]), points in direction of greatest increase
Example: y = x^2 + 1, y' = 2x, G = 2x, h = .1, x[0] = -1, G[0] = -2, x[1] = x[0] - G[0]h = -.8

EULER'S y' = t^2–y, y(0) = 1, h = .2, t[0] = 0, t[1] = .2, y[1] = y[0]+hy'[0] = 1+.2(0^2-1) = .8, t[2] = .4, y[2] = .8+.2(.2^2-.8) = .648

HEUN'S p[k+1] = y[k] + hf(t[k],y[k]), y[k+1] = y[k] + h(f(t[k],y[k])+f(t[1],p[k+1]))/2
Example: f(y,t) = y' = t^2 – y, y(0) = 1, h = .2, y[0]=1, t[0]=0, t[1]=.2, p[1] = y[0]+hf(t[0],y[0]) = 1+.2(0^2 - 1) = .8
y[1] = y[0]+h(f(t[0],y[0])+f(t[1],p[1]))/2 = 1+.2((0^2-1)+((.2)^2-1))/2 = .8240

TAYLOR APPROX. y(t[k] + h) = y(t[k]) + h y'(t[k]) + h^2/2! y"(t[k]) + ... h^n/n! y{n}(t[k])

RUNGE-KUTTA f[1] = f(t[k],y[k]), f[2] = f(t[k]+h/2,y[k]+(h/2)f[1]), f[3] = f(t[k]+h/2,y[k]+(h/2)f[2]), f[4] = f(t[k]+h,y[k]+hf[3])
y[k+1] = y[k] + (h(f[1]+2f[2]+2f[3]+f[4]))/6

SYSTEMS OF EQS. Euler: x-x[0] = f(t,x,y)(t-t[0]), y-y[0] = g(t,x,y)(t-t[0]), x[1] = x[0]+f(t,x,y)(t[1]-t[0]), y[1] = y[0]+f(t,x,y)(t[1]-t[0])
Runge-Kutta: [k+1] = x[k] + (h/6)(f[1]+2f[2]+2f[3]+f[4]), y[k+1] = y[k] + (h/6)(g[1]+2g[2]+2g[3]+g[4]), f[1] = f(t[k],x[k],y[k])
f[2] = f(t[k]+h/2,x[k]+(h/2)f[1],y[k]+(h/2)g[1]), f[3] = f(t[k]+h/2,x[k]+(h/2)f[2],y[k]+(h/2)g[2]), f[4] = f(t[k]+h,x[k]+hf[3],y[k]+hg[3])

BINARY NOTATION (N number F fraction S sign)Fixed Point: $((-1)^S)$(NNNN.FFFF), stored as SNNNNFFFF
Floating Point: $((-1)^S)$(1 + .FFFF) * 2^((NNNN - 1023), stored as SNNNNFFFF

ERROR (p value, p' approx, p = p' + e_1, q = q' + e_2) Absolute: |p – p'|, Relative: |p – p'| / |p|, p ≠ 0
p + q = p' + q' + (e_1 + e_2), pq = p' + q' + (q'e_1 + p'e_2 + e_1e_2)

[F(x) = 0] Monotone: no sign change, Oscillating = changes signs each iteration
Bisection Method (start with A and B); C = (A + B)/2; If(A)f(C) < 0, C = A, Else C = B
Regula Falsi Method (start with A and B); C = B – [(f(B)(B – A))/(f(B) – F(A))], same checks/loop
Newton-Raphson Method (from 1st order Taylor Series, start with p_0): $p_k = p_{k-1} - (f(p_{k-1})/f'(p_{k-1}))$
f'(x) = (f(x + E) - f(x))/E for some very small E; converges order 2 if simple root, 1 if multiple
Secant Method (start with p_0 and p_1), $p_{k+1} = p_k - [((f(p_k)(p_k - p_{k-1}))/(f(p_k) - f(p_{k-1}))]$
Horizontal convergence when | f(x) | < eps, vertical when | x_n – p | < del, f(p) = 0
Well-conditioned = clear/straight near zero, ill = multi-crossing, close to x-axis

[Ax = B] Upper triangular: only 0 below diagonal, Back substituting: solve last equation, substitute up
Can multiply by scalar, add multiples, move order of rows in a matrix; pivot = largest left in column
Gaussian Elimination: Make upper triangular, back substitute (use if only 1 system)
LU factorization (place next to identity matrix, perform same on both, substitute) (use for >1 system)

ITERATIVE [Ax=B] (use on big sparse matrices); Gauss-Seidel Method: Get x_k, y_k from x_{k-1} and y_{k-1}
Jacobi Method: Get x_k, y_k from x_k and y_k, (x_{k-1} and y_{k-1} when not possible)
Converges for strictly diagonal (| a[k][k] | > ∑[j=1;j≠k;j->n] | a[k][j] |, k goes from 1 to n)

POLYNOMIAL APPROX. Horner's Method: derivative of (∑[j=n;j->1] (a_jx^{j+1})/(j+1)) to approx. p(x)
Lagrange: have n points, p[n](x) = ∑[k=0;n] y_k*(π[j=0;j≠k;j->n](x-x_j))/(π[j=0;j≠k;j->n](x_k-x_j))
Newton Method (using "divided differences", f[g(x)]); P_n(x) = P_{n-1}(x) + a_n(π[i=0;n](x-x_i))
f[x_k] = f(x_k), f[x_{k-1},x_k] = (f[x_k] - f[x_{k-1}])/(x_k - x_{k-1}), get a_i from f[x_{i-1},x_i] or f[x_1]
Padé Approximation: $R_{n,m}(x) = P_n(x)/Q_m(x)$, $P_n(x) = p_0 + p_1x + ... p_nx^n$, $Q_m(x) = 1 + q_1x + ... q_mx^m$
$f(x)Q_m(x) - P_n(x) = 0$; $a_0-p_0 = 0$, $a_1+a_0q_1-p_1 = 0$, $a_2+a_1q_1+a_0q_2-p_2 = 0$, etc., get a_i from Taylor exp. of f(x)