1. Errors and Taylor Series

True Error = E_t = True Value – Approx. Relative Error = $\varepsilon_t = E_t$ / True Value * 100%

Approx. Error = E_a = Curr. – Prev. Approx Relative Error = $\varepsilon_a = E_a$ / Curr. * 100%

Stopping Criteria = ε_s is when $|\varepsilon_a| < \varepsilon_s$

Round-off errors:

Float: sign | exponent (signed int) | mantissa (normalized)

Overflow, underflow, # operations, small + large numbers, subtractive cancellation

Truncation errors:

Comes from dropping terms in infinite series

Taylor Series: $P(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$

Or: $P(x_0+h) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} h^n$

To use Taylor Series for Error Estimate:

$\Delta f(\bar{x}) = |f(x) - f(\bar{x})| = |f^1(\bar{x})|(x-\bar{x}) = |f^1(\bar{x})|\Delta x$

Multi Variable Functions (1st order):

$\Delta f(\bar{x}_1, \bar{x}_2 \ldots \bar{x}_n) = \left|\frac{\partial f}{\partial x_1}\right| \Delta\bar{x}_1 + \left|\frac{\partial f}{\partial x_2}\right| \Delta\bar{x}_2 + \cdots + \left|\frac{\partial f}{\partial x_n}\right| \Delta\bar{x}_n$

2. Solving Non-Linear Equations (Roots of Equations)

Quadratic Equation: $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

Approach 1: Isolate variable on one side (often hard)

Approach 2: Make equation equal to 0 and solve root

Incremental Search: eval different x's and look for sign changes

Bisection Method (Linear convergence): Based on Intermediate Value Theorem:

Step 1: Choose x_L and x_u such that function changes sign

Over the interval $[x_L, x_u]$. Check with: $f(x_L)f(x_u) < 0$

Step 2: Estimate midpoint $x_m = \frac{x_L + x_u}{2}$

Step 3: If $f(x_L)f(x_m) < 0$, then root lies in lower subinterval.

If $f(x_L)f(x_m) > 0$, then root lies in upper subinterval.

Else $f(x_L)f(x_m) = 0$ means you've found exact root.

Step 4: Get new estimate using new x_L and x_m.

Step 5: Determine $|\varepsilon_a| = |\frac{x_m^{new} - x_m}{x_m^{new}}| * 100\%$

If $\varepsilon_a > \varepsilon_s$ repeat. Else, $x_r = x_m^{new}$

Error Bound: Accuracy = $|x_n - x_r| \leq \frac{1}{2^n}(x_u^0 - x_L^0)$

Order of Convergence: Let x_n be sequence converging to r and $e_i = |x_n - r|$

If there exists positive constants λ and α so: $e_{i+1} = \lambda e_i^{\alpha}$

Then x_n converges to r with order α and

asymptotic error constant λ

Newton-Raphson Method (Quadratic convergence):

$x_{i+1} = x_i - \frac{f(x_i)}{f^1(x_i)}$, Limitations: x_0 must be close or may not converge, needs $f^1(x)$ be known.

Modified Newton-Raphson (Multiple Roots): converges better only if multiple roots

$X_{i+1} = x_i - f(x_i)f^1(x_i) / f^1(x_i)^2 - f(x_i)f^2(x_i)$

Secant Method (Superlinear convergence):

Starting with two initial approxs. x_0 and x_1 the next approx is the x-intercept of the straight line joining $(x_0, f(x_0))$ and $(x_1, f(x_1))$. Doesn't require sign change (not bracketing). Not grntd to converge. Almost as fast as Newton-Raphson. No need to know derivative

Modified Secant Method (Multiple Roots): converges better only if multiple roots

$X_{i+1} = x_i - u(x_i)(x_{i-1} - x_i)/(u(x_{i-1}) - u(x_0))$

False-Position Method (Superlinear convergence):

Same as Secant Method except we always choose to keep the one of our last two points that has opposite sign as our new point. Check with: $f(x_n)*f(x_{n-a}) < 0$

Not always faster, consider graph like backwards L.

Fixed Point Iteration Method (Linear Convergence):

Break function f(x) up into g(x)=x and h(x)=(the rest) Then find $x_{i+1} = h(x_i)$

Fixed Point Theorem:

a) if function h(x) is continuous on [a,b] and value h(x) is between [a,b] for all x within [a,b] then fixed point exists.

b) if h(x) has continuous derivative on [a,b] a constant 0<k<1 so

$|h^1(x)| \leq k$ for all x in [a,b] and $|h^1(x)| < 1$

Then fixed point is unique and for any x_i in [a,b] will converge.

Bracket vs Open Methods: Bracket methods always converge, good error control, simple, robust. But slow and limited to 1D.

Open methods guess root fast, simple, useful in nD. But not necessarily stable or robust.

Mean Value Theorem: If f is cont and diff'able on [a,b] then must exist point c with slope equals the slope of the line a-b. $f^1(c) = (f(b) - f(a)) / (b-a)$

3. System of Linear Algebraic Equations

Matrix Properties: AB = BA only if $A=B^{-1}$ and $B=A^{-1}$

$AA^{-1} = A^{-1}A = I$, AI = A, (AB)C = A(BC), d(AB) = (dA)B = A(dB)

$\#R_1 \text{ x } \#C_1 * \#R_2 \text{ x } \#C_2 = \#R_1 \text{ x } \#C_2$ where $\#C_1$ must equal $\#R_2$

Determinants:

$\det(A) = |A| = a_{11} * \det\begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix} - a_{12} * \det\begin{pmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{pmatrix}$

$+ a_{13} * \det\begin{pmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}$, $\det(B) = b_{11}b_{22} - b_{12}b_{21}$

Cramer's Rule:

For [A]{x,y,z}={b}

$X = D_x / D$ where D_x is det(A) with col x = {b}

$Y = D_y / D$ where D_y is det(A) with col y = {b}

$Z = D_z / D$ where D_z is det(A) with col z = {b}

Gauss Elimination: Forward elimination by using pivot row to get rid of elinination row so we end with upper right triangle. Backward substitute equations to solve.

Gauss-Jordan: instead of backward sub, use backward elimination to get identity matrix.

Problems: Can lead to div(0), round off errors, ill conditioned systems.

Scaling: Before choosing a pivot row, scale so each row's largest coefficient is one. ONLY use to judge pivot row, return values for actually proceeding with calculations.

Partial Pivotting: Before each pivot selection, choose largest absolute value (after Scaling if necessary).

System Condition: Ill conditioned system means small change in coefficents greatly affects result. Ill condtd system has det near 0. A det of 0 means no (or infinite) solutions.

Singular Systems: When two equations are identical, we may not have enough equations to solve. This can be seen from a determinant of 0.

Since triangular matrices det is equal to the product of their diagonal elements, we just need to look for 0 on the diagonal.

Source of Error: The source of error is $\frac{2n^3}{3} + n^2$. The first term is from forward elimination. The second from backward substitution. As n->∞ the second term disappears. So error comes from forward sub.

LU Decomposition:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ \frac{a_{21}}{a_{11}} & 1 & 0 \\ \frac{a_{31}}{a_{11}} & \frac{\bar{a}_{32}}{\bar{a}_{22}} & 1 \end{pmatrix} * \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & \bar{a}_{22} & \bar{a}_{23} \\ 0 & 0 & \bar{\bar{a}}_{33} \end{pmatrix}$$

Crout's Method:

$L_{11} \mid L_{11}U_{12} \mid L_{11}U_{13}$

$L_{21} \mid L_{21}U_{12} + L_{22} \mid L_{21}U_{13} + L_{22}U_{23}$

$L_{31} \mid L_{31}U_{12} + L_{32} \mid L_{31}U_{13} + L_{32}U_{23} + L_{33}$

Now instead of [A]{x} = {b} we have [L][U]{x} = {b}

We can FIRST solve: [L]{y} = {b} then [U]{x} = {y}

With partial pivotting this becomse: PA = LUx = Pb

Inverse from LU: Solve Ax = LUx = {1 0 0} for first column of inverse etc.

Jacobi Iteration:

Solve each equation so we have x_1, x_2, etc. Isolated for equation 1, 2 etc. We make initial guesses for each. We go and solve each equation using our guesses to get new guesses.

Convergence: This will converge if the absolute value of the diagonal element is greater than the absolute values of the other row elements for each row.

Gauss-Seidel Method: Same as Jacobi but we immediately update our guesses

Norms and Conditions: Norms: ||a|| = 0 if and only if {a} = 0

||ca||=|c|||a||,||ab||≤||a|| ||b||,||a+b||≤||a||+||b||

Vector Norms: $||a||_\infty$ = max absolute value, $||a||_1$ = sum of abs values, $||a||_2$ = $||a||_{Euclidean}$ = sqrt(sum(values squared))

Matrix Norms: $||A||_\infty$ = max(row sum(abs value)), $||A||_1$ = max(column sum(abs value)), $||A||_{Euclidean}$ = sqrt(sum(all elements squared))

Conditions: $1/(||A||||A^{-1}||*||r||/||b||) \leq e/x_{ts} \leq ||A||||A^{-1}||\ (||r||/||b||)$

Cond(A) = $||A||||A^{-1}||$ if cond(A) is close to 1 problem is easy

Else if cond(A) much larger than 1, large err. or pos. no solution

Newton's Method:

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - [J(x,y)]^{-1} \begin{pmatrix} f_1(x_0, y_0) \\ f_2(x_0, y_0) \end{pmatrix}$$

$$J(x,y) = \begin{bmatrix} f_1^1(x,y)dx & f_1^1(x,y)dy \\ f_2^1(x,y)dx & f_2^1(x,y)dy \end{bmatrix}, [M]^{-1} = 1/\det(M) \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

4. Curve Fitting – Regression and Interpolation

Least Square: Make a grid with x, y, xy, x^2 then sum each row, and find the means of x and y. Our best fit line $y = b_0 + b_1 x$ where:

$b_1 = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sum x^2 - \frac{(\sum x)^2}{n}}$, b_0 = mean(y) – b_1 * mean(x)

Error Quantification:

S_t = total sum of squares = $\sum_{i=1}^{n}(y_i - \bar{y})^2$

S_y = std dev = $\sqrt{\frac{S_r}{n-2}}$, S_r = residual = $(y_i - a_0 - a_1x_i)^2$

Correlation Coefficient r: $r^2 = (S_t - S_r)/S_t$

Linearization:

$Y = bx^m$	$\ln(y) = m \ln(x) + \ln(b)$	y=ln(y), x=ln(x)
$Y = be^{mx}$	$\ln(y) = m x + \ln(b)$	y=ln(y), x = x
$Y = b10^{mx}$	$\log(y) = m x + \log(b)$	y=log(y), x = x
$Y = 1/(mx + b)$	$1/y = mx + b$	y=1/y, x=x
$Y = mx/(b+x)$	$1/y = b/m\ 1/x + 1/m$	y=1/y, x = 1/x

Polynomial Regression:

$$\text{Solve: } \begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \{a\} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \\ \sum y_i x_i^2 \end{bmatrix}$$

$S_r = \sum(y_i - (a_0 + a_1x_i + a_2x_i^2))^2$, $S_{y/x} = \sqrt{\frac{S_r}{n-(m+1)}}$ where m is order

Residual: $e_i = y_i - (a_0 + a_1x_i + a_2x_i^2)$

Interpolating Polynomials:

$F_n(x) = f(x_0) + f[x_1,x_0](x-x_0) + f[x_2,x_1,x_0](x-x_1)(x-x_0) + \ldots$

$F[x_i, x_j] = \frac{f(x_i) - f(x_j)}{x_i - x_j}$, $f[x_i,x_j,x_k] = \frac{f[x_i,x_j] - f[x_j,x_k]}{x_i - x_k}$

DDT Headers: i, x_i, $f(x_i)$, $d_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$, $dd_i = \frac{dd_{i+1} - dd_i}{x_{i+2} - x_i}$

Error:$R_n = f^{(n+1)}(sqgl)/(n+1)! * (x-x_0)(x-x_1)...(x-x_n)$
If we have n+1 points we can use: $R_n = f[x_{n+1}, x_n, ... x_0](x-x_0)...(x-x_n)$
In general, the error for nth order polynomial is:
$R_n = P_{n+1}(x) – P_n(x)$ or the diff between n and n+1 order guesses
LaGrange Polynomial Interpolating:
$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$ where $L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x-x_j}{x_i-x_j}$
Error:$f(x) = P(x) + f^{(n+1)}(sqgl)/(n+1)! * (x-x_0)...(x-x_n)$
Error for Taylor Series Polynomial: $f^{(n+1)}(sqgl)/(n+1)! * (x-x_0)^n$
Splines: Take cubic lines between each point. Make them fit boundary conditions, all data points, continuious, 1st deriv equal, 2nd deriv equal
$S_j = a_0 + b_0(x-x_0) + c_0(x-x_0)^2 + d_0(x-x_0)^3$
$S_{j+1} = a_1 b_1(x-x_1) + c_1(x-x_1)^2 + d_1(x-x_1)^3$

5. Numerical Integration and Differentiation

Finite Difference: $f^1(x) \approx (f(x+h) – f(x))/h$ where h is step size
Backward Divided Difference Tables:

First Derivative	Error	Second Derivative	Error
$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h}$	$O(h)$	$f''(x_i) = \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{h^2}$	$O(h)$
$f'(x_i) = \frac{3f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{2h}$	$O(h^2)$	$f''(x_i) = \frac{2f(x_i) - 5f(x_{i-1}) + 4f(x_{i-2}) - f(x_{i-3})}{h^2}$	$O(h^2)$

Third Derivative	Error
$f'''(x_i) = \frac{f(x_i) - 3f(x_{i-1}) + 3f(x_{i-2}) - f(x_{i-3})}{h^3}$	$O(h)$
$f'''(x_i) = \frac{5f(x_i) - 18f(x_{i-1}) + 24f(x_{i-2}) - 14f(x_{i-3}) + 3f(x_{i-4})}{2h^3}$	$O(h^2)$

Fourth Derivative	Error
$f''''(x_i) = \frac{f(x_i) - 4f(x_{i-1}) + 6f(x_{i-2}) - 4f(x_{i-3}) + f(x_{i-4})}{h^4}$	$O(h)$
$f''''(x_i) = \frac{3f(x_i) - 14f(x_{i-1}) + 26f(x_{i-2}) - 24f(x_{i-3}) + 11f(x_{i-4}) - 2f(x_{i-5})}{h^4}$	$O(h^2)$

Forward Divided Difference Table:

First Derivative	Error	Second Derivative	Error
$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$	$O(h)$	$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2}$	$O(h)$
$f'(x_i) = \frac{-f(x_{i+2}) + 4f(x_{i+1}) - 3f(x_i)}{2h}$	$O(h^2)$	$f''(x_i) = \frac{-f(x_{i+3}) + 4f(x_{i+2}) - 5f(x_{i+1}) + 2f(x_i)}{h^2}$	$O(h^2)$

Third Derivative	Error
$f'''(x_i) = \frac{f(x_{i+3}) - 3f(x_{i+2}) + 3f(x_{i+1}) - f(x_i)}{h^3}$	$O(h)$
$f'''(x_i) = \frac{-3f(x_{i+4}) + 14f(x_{i+3}) - 24f(x_{i+2}) + 18f(x_{i+1}) - 5f(x_i)}{2h^3}$	$O(h^2)$

Fourth Derivative	Error
$f''''(x_i) = \frac{f(x_{i+4}) - 4f(x_{i+3}) + 6f(x_{i+2}) - 4f(x_{i+1}) + f(x_i)}{h^4}$	$O(h)$

Centered Divided Difference Table:

First Derivative	Error	Second Derivative	Error
$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$	$O(h^2)$	$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}$	$O(h^2)$
$f'(x_i) = \frac{-f(x_{i+2}) + 8f(x_{i+1}) - 8f(x_{i-1}) + f(x_{i-2})}{12h}$	$O(h^4)$	$f''(x_i) = \frac{-f(x_{i+2}) + 16f(x_{i+1}) - 30f(x_i) + 16f(x_{i-1}) - f(x_{i-2})}{12h^2}$	$O(h^4)$

Third Derivative	Error
$f'''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + 2f(x_{i-1}) - f(x_{i-2})}{2h^3}$	$O(h^2)$
$f'''(x_i) = \frac{-f(x_{i+3}) + 8f(x_{i+2}) - 13f(x_{i+1}) + 13f(x_{i-1}) - 8f(x_{i-2}) + f(x_{i-3})}{8h^3}$	$O(h^4)$

Fourth Derivative	Error
$f''''(x_i) = \frac{f(x_{i+2}) - 4f(x_{i+1}) + 6f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{h^4}$	$O(h^2)$
$f''''(x_i) = \frac{-f(x_{i+3}) + 12f(x_{i+2}) + 39f(x_{i+1}) + 56f(x_i) - 39f(x_{i-1}) + 12f(x_{i-2}) + f(x_{i-3})}{6h^4}$	$O(h^4)$

Higher Accuracy From Taylor Series: start off with Taylor series for $f(x_{i+1})$ = series, then rearrange to solve for $f^{(n)}$ as desired, and show that $O(h^n)$ is the order of terms dropped.
Romberg Integration (Richardson Extrapolation): TV = $AV_{2n} + (AV_{2n} – AV_n)/3$
$f^1(x_0) = N_j(h) + O(h^j)$... $N_1(h) = 1/h\ [f(x_0+h)-f(x_0)]$...
$N_2(h) = \frac{1}{h}(4f\left(x_0 + \frac{h}{2}\right) - f(x_0 + h) - 3f(x_0))$...
$N_3(h) = \frac{1}{12h}(f(x_0 + 4h) - 12f(x_0 + 2h) + 32f(x_0 + h) - 21f(x_0))$
Unequally Spaced Data: Use second order Lagrange interpolating Polynomial:
$f^1(x) = f(x_{i-1})\frac{2x-x_i-x_{i+1}}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} + f(x_i)\frac{2x-x_{i-1}-x_{i+1}}{(x_i-x_{i-1})(x_i-x_{i+1})} + f(x_{i+1})\frac{2x-x_{i-1}-x_i}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}$
Numerical Integration: Quadrature Formula: $\int_a^b f(x)dx = Q[f] + E[f]$
E[f] = truncation error, $Q[f] = \sum_{i=0}^{n} C_i f(x_i)$ where C_i depends on method.
Degree of Precision: equal to the highest order of polynomial for E[f] = 0
Newton-Cotes: Class of formulas that replace $f(x_i)$ with $f_n(x)$ = nth order polynomial
Trapezoidal Rule: Newton-Cotes $f_1(x)$, I=(b-a)*(f(a)-f(b))/2, $E_t = -1/12 * f^2(sqgl)(b-a)^3$
Multiple Application: We subdivide interval into smaller ones. Error becomes:
$E_a = -(b-a)^3/12n^2 * \bar{f}^2$, $\sum f^2(sqgl) \approx n\bar{f}^2$
Simpson's 1/3 Rule: I = h/3 [$f(x_0) + f(x_n) + 4(f(x_{odd}) +...) + 2(f(x_{even}) +...)$], h=(b-a)/2 and n must be an odd number to give even number of subintervals.
Error: $\sum f^{(4)}(sgl) \approx n\bar{f}^{(4)}$, $E_a = -(b-a)^5/180n^4 * \bar{f}^{(4)}$
Simpson's 3/8 Rule: I = 3h/8 [$f(x_0) + f(x_n) + 2(f(x_{divisible\ by\ 3}) +...) + 3(f(x_{remaining}) + ...)$]
h=(b-a)/n, n must be multiple of 3
Method of Undetermined Coefficients: $\int_a^b f(x)dx = c_0f(a) + c_1f(b) + Kf^2(sqgl)$, $K=-h^3/12$
$E_t = -1/12\ (f^2(sqgl)(b-a)^3)$
Newton-Cotes Formulas: number of segments n is even, degree of precision = n+1, else n
Gauss Quadrature: Two points so errors cancel out: Step 1 convert to interval [-1,1]
$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}x + \frac{b+a}{2}\right)dx$ ... Step 2: Use Quadrature for n points:
$\int_{-1}^{1} f(x)dx = c_1f(x_1) + c_2f(x_2) + c_3f(x_3)$ ... where c and x values are all given from book.
Make both sides of above equal.

Points	Weighting Factors	Function Arguments
2	$c_0 = 1.0000000$	$x_0 = -0.577350269$
	$c_1 = 1.0000000$	$x_1 = 0.577350269$
3	$c_0 = 0.5555556$	$x_0 = -0.774596669$
	$c_1 = 0.8888889$	$x_1 = 0.0$
	$c_2 = 0.5555556$	$x_2 = 0.774596669$
4	$c_0 = 0.3478548$	$x_0 = -0.861136312$
	$c_1 = 0.6521452$	$x_1 = -0.339981044$
	$c_2 = 0.6521452$	$x_2 = 0.339981044$
	$c_3 = 0.3478548$	$x_3 = 0.861136312$
5	$c_0 = 0.2369269$	$x_0 = -0.906179846$
	$c_1 = 0.4786287$	$x_1 = -0.538469310$
	$c_2 = 0.5688889$	$x_2 = 0.0$
	$c_3 = 0.4786287$	$x_3 = 0.538469310$
	$c_4 = 0.2369269$	$x_4 = 0.906179846$
6	$c_0 = 0.1713245$	$x_0 = -0.932469514$
	$c_1 = 0.3607616$	$x_1 = -0.661209386$
	$c_2 = 0.4679139$	$x_2 = -0.238619186$
	$c_3 = 0.4679139$	$x_3 = 0.238619186$
	$c_4 = 0.3607616$	$x_4 = 0.661209386$
	$c_5 = 0.1713245$	$x_5 = 0.932469514$

Errors: For n points: $f^{(2n)}(sqgl)$
Romberg Integration: see Richardson. Vary segments n so n, 2n, 4n, 8n, then improve
Integration Properties: $\int \frac{a}{x}dx = a\ lnx + C$, $\int ax^n dx = a\frac{x^{n+1}}{n+1} + C$, $\int e^x dx = e^x + C$
Constants and addition can be pulled out. To opposie derive chain $f^1(g(x))g^1(x)$ us sub.

6. Ordinary Differential Equations

Euler's Method from Taylor Series: $w_0 = y_0$, $w_{i+1} = w_i + h * f(t_i, w_i)$
Heun's Method: improves by averaging f^1 from start and end.
$Y_{i+1} = y_i + ([f(t_i,y_i) + f(t_{i+1}, y^0_{i+1})]/2) * h$
Midpoint or Improved Polygon: $y_{i+0.5} = y_i + f(x_i, y_i) * h/2$ ... $y_{i+1} = y_i + f(x_{i+0.5}, y_{i+0.5})h$
Higher Order (General): $w_0 = y_0$... $w_{i+1} = w_i + h[f(t_i,w_i) + (h^{(n-1)}/n!)f^{(n-1)}(t_i,w_i)]$
Runge Kulta (RK): Given $dy/dt = f(x,y)$, $y(x_0) = y_0$
$K1=hf(x_0, y_0)$... $k2=hf(x_0+a_1h_1, y_0+B_1K1)$... $kn=hf(x_0 + a_{n-1}h, y_0 +b_{n-1}K1 + c_{n-1}K2 + ... d_{n-1}k_{n-1})$
Incremental function: $y_1 = y_0 + u_1k_1 + ... u_nk_n$ ... Once n is chosen, values are evaluated by setting general equation equal to terms of Taylor Series equation. One of unknowns must be assumed: Heun Single Correct: u_2=½, Midpoint:u_2=1, Raltson's: u_2=2/3
RK4: $y_{i+1} = y_i + (1/6)[k_1 + 2K_2 + 2K_3 + k_4)$... $k_1 = hf(x_i,y_i)$... $k_2 = hf(x_i +0.5h, y_i + 0.5k_1)$
$K_3=hf(x_i+0.5h, y_i +0.5k_2)$... $k_4hf(x_i+h, y_i+k_3)$
Euler for Systems:Apply Euler for each sep.: $u^1=f(x,u,v)\ u(x_0) = u_0$... $v^1=g(x,u,v)\ u(x_0)=v_0$
$U_{i+1} = u_i + h * f(x_i, u_i, v_i)$ $v_{i+1} = v_i + h * g(x_i, u_i, v_i)$
RK4 for Systems: (using same notation) create separate recurrence functions k, so k_n and L_n are used in u_{i+1} and v_{i+1} of Rk4 formula respectively.
Systems of Two Equations (RK4): We need two initial points.
$Y_{i+1,1} = y_{i,1} + 1/6 * (k_{1,1} + 2k_{2,1} + 2k_{3,1} + k_{4,1})h$ $y_{i+1,2} = y_{i,2} + 1/6 * (k_{1,2} + 2k_{2,2} + 2k_{3,2} + k_{4,2})\ h$
Translate Higher Order ODE's: use substitution so $u_1 = y$, $u_2 = y^1$, $u_3 = y^3$ etc.
So $u^1_n = f(t,u_1,u_2...u_n)$ (yay first order!)
Adams-Bashforth (Multistep):

$$y_{n+1} = y_n + hf(t_n, y_n), \qquad \text{(This is the Euler method)}$$
$$y_{n+2} = y_{n+1} + h\left(\frac{3}{2}f(t_{n+1}, y_{n+1}) - \frac{1}{2}f(t_n, y_n)\right),$$
$$y_{n+3} = y_{n+2} + h\left(\frac{23}{12}f(t_{n+2}, y_{n+2}) - \frac{4}{3}f(t_{n+1}, y_{n+1}) + \frac{5}{12}f(t_n, y_n)\right),$$
$$y_{n+4} = y_{n+3} + h\left(\frac{55}{24}f(t_{n+3}, y_{n+3}) - \frac{59}{24}f(t_{n+2}, y_{n+2}) + \frac{37}{24}f(t_{n+1}, y_{n+1}) - \frac{3}{8}f(t_n, y_n)\right),$$
$$y_{n+5} = y_{n+4} + h\left(\frac{1901}{720}f(t_{n+4}, y_{n+4}) - \frac{1387}{360}f(t_{n+3}, y_{n+3}) + \frac{109}{30}f(t_{n+2}, y_{n+2}) - \frac{637}{360}f(t_{n+1}, y_{n+1}) + \frac{251}{720}f(t_n, y_n)\right)$$

coefficients b_j can be determined as follows. Use polynomial interpolation to find the polynomial p of degree $s-1$ such that

$$p(t_{n+i}) = f(t_{n+i}, y_{n+i}), \qquad \text{for } i = 0, \ldots, s-1.$$