Numerical Computation Cheat Sheet **by Radin Shahdaei**

Initial-Value Problems for ODEs

$$y'(x) = f(x, y)$$
$$y(x_0) = y_0$$

General Way:

$$x_n = x_0 + nh$$
$$w_{n+1} = w_n + h\Phi(x_n, w_n) \rightarrow \text{difference equation}$$

Local Truncation Error:

$$\tau_{i+1} = \frac{y_{i+1} - (y_i + h\Phi(x_i, y_i))}{h} = \frac{y_{i+1} - y_i}{h} - \Phi(x_i, y_i)$$

Taylor's Method for order k:

$$x_n = x_0 + nh$$
$$w_{n+1} = w_n + hT^k(x_n, w_n)$$
$$T^k(x, y) = f(x, y) + \frac{h}{2}f'(x, y) + \frac{h^2}{6}f''(x, y) + \cdots + \frac{h^{k-1}}{k!}f^{(k-1)}(x, y)$$

$$\tau_{i+1} = \frac{y_{i+1} - y_i}{h} - T^k(x_i, y_i) = \frac{h^k y^{(k+1)}\big(\mu(x_i)\big)}{(k+1)!} = O(h^k)$$
$$\sum \tau_i = \sum \frac{h^k y^{(k+1)}\big(\mu(x_i)\big)}{(k+1)!} = n \times \frac{h^k y^{(k+1)}(\mu)}{(k+1)!} = O(h^{k-1})$$

Euler's Method is Taylor's Method with k=1

$$x_n = x_0 + nh$$
$$w_{n+1} = w_n + hf(x_n, w_n)$$

Runge-Kutta Methods:

$$x_n = x_0 + nh$$
$$w_{n+1} = w_n + \gamma_1 K_1 + \gamma_2 K_2 + \cdots + \gamma_j K_j$$

$$K_1 = hf(x_n, w_n)$$
$$K_l = hf\left(x_i + \alpha_l h, w_i + \sum_{m=1}^{l-1} \beta_{ml} K_m\right), 2 \le l \le j$$

find γ_1 to γ_j such that w_{n+1} has Taylor's Method error for order $k = j$

Solve Runge-Kutta for j = 2:

$$K_1 = hf(x_n, w_n)$$
$$K_2 = \ hf(x_n + \alpha h, w_n + \beta K_1)$$

Recall that Taylor's Method for k=2 is:

$$x_n = x_0 + nh$$
$$w_{n+1} = w_n + hf(x_n, w_n) + \frac{h^2 f'(x_n, y_n)}{2}$$

$$f'(x,y) = \frac{d}{dx} f(x,y) = \frac{\partial f(x,y)}{\partial x} + \frac{\partial f(x,y)}{\partial y} \times \frac{\partial y}{\partial x} = \frac{\partial f(x,y)}{\partial x} + f(x,y)\frac{\partial f(x,y)}{\partial y}$$
$$\rightarrow w_{n+1} = w_n + hf(x_n, w_n) + \frac{h^2}{2}\frac{\partial f(x,y)}{\partial x} + \frac{h^2 f(x,y)}{2}\frac{\partial f(x,y)}{\partial y}$$

Recall that Taylor's Expansion for two variables is:

$$f(x+h, y+k) = f(x,y) + h\frac{\partial f(x,y)}{\partial x} + k\frac{\partial f(x,y)}{\partial y} + \left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right)^2 f(\theta_x, \theta_y)$$
$$\rightarrow \ K_2 = hf(x_n, w_n) + \alpha h^2 \frac{\partial f(x,w)}{\partial x} + \beta h^2 f(x_n, w_n)\frac{\partial f(x,w)}{\partial w} + O(h^3)$$

$$w_{n+1} = w_n + \gamma_1 K_1 + \gamma_2 K_2$$
$$w_{n+1} = w_n + (\gamma_1 + \gamma_2)hf(x_n, w_n) + \gamma_2 \alpha h^2 \frac{\partial f(x,w)}{\partial x} + \gamma_2 \beta h^2 f(x_n, w_n)\frac{\partial f(x,w)}{\partial w}$$

$$\gamma_1 + \gamma_2 = 1, \gamma_2 \alpha = \frac{1}{2}, \gamma_2 \beta = \frac{1}{2}$$

The system of equations gives more than one solution!

1) **Midpoint Method:**

$$\gamma_1 = 0, \gamma_2 = 1, \alpha = \frac{1}{2}, \beta = \frac{1}{2}$$

$$x_n = x_0 + nh$$
$$w_{n+1} = w_n + hf\left(x_n + \frac{h}{2}, w_n + \frac{hf(x_n, w_n)}{2}\right)$$

2) **Modified Euler's Method (AKA Huen's Method):**

$$\gamma_1 = \frac{1}{2}, \gamma_2 = \frac{1}{2}, \alpha = 1, \beta = 1$$

$$x_n = x_0 + nh$$
$$w_{n+1} = w_n + \frac{h}{2}\Big(f(x_n, w_n) + f\big(x_{n+1}, y_n + hf(x_n, w_n)\big)\Big)$$

In both of these methods, because the coefficients were solved by 2nd order Runge-Kutta, the local truncation error is equal to Taylor's 2nd Order Method, which is $O(h^2)$

Higher Order Runge-Kutta are found in a similar way.

3rd Order Runge-Kutta:

$$\begin{cases} K_{١} = hf(x_n, y_n) \\ K_{٢} = hf(x_n + \frac{h}{٢}, y_n + \frac{K_{١}}{٢}) \\ K_{٣} = hf(x_{n+١}, y_n + ٢K_{٢} - K_{١}) \\ y_{n+١} = y_n + \frac{١}{۶}(K_{١} + ۴K_{٢} + K_{٣}) \end{cases}$$

4th Order Runge-Kutta:

$$\begin{cases} K_{١} = hf(x_n, y_n) \\ K_{٢} = hf(x_n + \frac{h}{٢}, y_n + \frac{K_{١}}{٢}) \\ K_{٣} = hf(x_n + \frac{h}{٢}, y_n + \frac{K_{٢}}{٢}) \\ K_{۴} = hf(x_{n+١}, y_n + K_{٣}) \\ y_{n+١} = y_n + \frac{١}{۶}(K_{١} + ٢K_{٢} + ٢K_{٣} + K_{۴}) \end{cases}$$

Solving for a system of equations is no different than solving for a single one, just use vectors as inputs and outputs of the functions. Solving an equation with higher-degree differentials is the same, convert it to a system of equations and use vector calculus.

Solving Equations in One Variable:

Order of Convergence:

Suppose $\{p_n\}_{n=0}^{\infty}$ is a sequence that converges to p, with $p_n \neq p$ for all n. If positive constants λ and α exist with

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^{\alpha}} = \lambda,$$

then $\{p_n\}_{n=0}^{\infty}$ **converges to p of order α, with asymptotic error constant λ.** ■

Bisection:

Binary search on the interval [a,b] to find the solution. An upper bound for the error is shown below:

$$|p_n - p| \leq \frac{|b-a|}{2^n}$$

Fixed-Point Method:

A number p is a fixed point for g if:

$f(p) = 0$

$g(x) = x \pm af(x) \rightarrow g(p) = p$

Fixed-Point Theorems:

Theorem 1)

i) if continuous function g is contained within $[a,b]$ within interval $[a,b]$, there exists at least one fixed point.

ii) if there exists constant k in interval $[0,1)$ such that: $|g'(x)| \leq k \; \forall x \in [a,b]$

$\rightarrow$ There is exactly one fixed point.

Proof:

i) if $g(a) = a$ or $g(b) = b$, one is a fixed point.

otherwise, $g(a) > a$ and $g(b) < b$

$\rightarrow$ let $h(x) = g(x) - x \rightarrow h(a) > 0$ and $h(b) < 0 \rightarrow h(p) = 0$ for some $p \in (a,b) \rightarrow g(p) = p$

ii) suppose there are 2 fixed points p and q $\rightarrow \dfrac{g(p) - g(q)}{p - q} = g'(\mu) = 1 \rightarrow$ contradiction

Theorem 2)

If the conditions hold for the Fixed-Point Theorem, then this series converges: $p_n = g(p_{n-1})$

Proof:

$|p_n - p| = |g(p_{n-1}) - g(p)| = |g'(\mu)||p_{n-1} - p| \leq k|p_{n-1} - p|$

$\rightarrow |p_n - p| \leq k^n |p_0 - p| = 0 \rightarrow |p_n - p| = 0$

Theorem 3)

if $g^{(i)} = 0$ for all $i < k$: the order of convergence for fixed point is k
proof:

$$g(p_n) = g(p) + (p_n - p)g'(p) + \cdots + \frac{(p_n - p)^{k-1} g^{(k-1)}(p)}{(k-1)!} + \frac{(p_n - p)^k g^{(k)}(p)}{k!} + \frac{(p_n - p)^{k+1} g^{(k+1)}(\mu)}{(k+1)!}$$
$$= \frac{(p_n - p)^k g^{(k)}(p)}{k!} + \frac{(p_n - p)^{k+1} g^{(k+1)}(\mu)}{(k+1)!}$$

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^k} = \frac{|p_n - p|^k}{|p_n - p|^k} \lim_{n\to\infty} \left(\frac{g^{(k)}(p)}{k!} + \frac{(p_n - p) g^{(k+1)}(\mu)}{(k+1)!} \right) = \frac{g^{(k)}(p)}{k!} = \lambda$$

The Newton-Raphson Method:

$$f(p) = f(p_i) + (p - p_i)f'(p_i) + \cdots \rightarrow p \approx p_i - \frac{f(p_i)}{f'(p_i)}$$

$$\rightarrow p_n = g(p_{n-1}) \text{ where } g(x) = x - \frac{f(x)}{f'(x)}$$

Convergence Proof: if $f(x)$ is 2 times diffrentiable on $[a, b]$ and there exists $p \in (a, b)$ such that $f(p) = 0$ and $f'(p) \neq 0 \rightarrow$ there exists δ such that the Newton $-$ Raphson sequence converges starting from $p_0 \in [p - \delta, p + \delta]$

$$p_n = g(p_{n-1}) \text{ where } g(x) = x - \frac{f(x)}{f'(x)}$$

Lemma 1: There exists δ_1 such that $f'(x) \neq 0$ for every $x \in [p - \delta_1, p + \delta_1]$
$\rightarrow g(x)$ is defined an continuous on $[p - \delta_1, p + \delta_1]$

$$g'(x) = 1 - \frac{\left(f'(x)\right)^2 - f(x)f''(x)}{\left(f'(x)\right)^2} = \frac{f(x)f''(x)}{\left(f'(x)\right)^2} \rightarrow g'(p) = 0$$

Lemma 2: There exists δ such that $|g'(x)| \leq k$ for some $k < 1$ for every $x \in [p - \delta, p + \delta]$

$|g(x) - p| = |g(x) - g(p)| = |g'(\mu)||x - p| \leq k|x - p| < |x - p| < \delta$
$\rightarrow$ g maps $[p - \delta, p + \delta]$ on to $[p - \delta, p + \delta]$

$\rightarrow$ It satisfies the Fixed $-$ Point Theorem $\rightarrow$ Newton $-$ Raphson converges with $\alpha = 2$ (at least)

The Secant Method:

Use an approximation for $f'(x)$ in Newton-Raphson:

$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1} - p_{n-2})}{f(p_{n-1}) - f(p_{n-2})}$$

Solving a System of Equations:

Direct Methods of solving a system of linear equations involve using matrix algebra, which we have previously learnt in Linear Algebra course!

- For minimizing the error caused by row elimination, choose the row with the biggest $|x_j|$ to eliminate others.

The objective is to solve Ax = b, or better represented:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$
$$\dots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

Iterative Methods:

The general form is like this: $x^{(k)} = Tx^{(k-1)} + c \rightarrow$ converges when $\|T\| < 1$

1) **The Jacobi Method:**

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}x_j^{(k-1)}\right)$$

$$A = L + D + U$$
$$Dx^{(k)} = b - (L + U)x^{(k-1)}$$
$$\rightarrow x^{(k)} = -D^{-1}(L + U)x^{(k-1)} + D^{-1}b$$

2) **The Gauss-Seidel Method:**

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{(k)} - \sum_{i<j} a_{ij}x_j^{(k-1)}\right)$$

$$A = L + D + U$$
$$(D + L)x^{(k)} = b - Ux^{(k-1)}$$
$$\rightarrow x^{(k)} = -(D + L)^{-1}Ux^{(k-1)} + (D + L)^{-1}b$$

Solving Non-Linear System of Equations:

$$f(x, y) = 0$$
$$g(x, y) = 0$$

let (α, β) be the solution of the system where $\alpha = x_0 + h_0$ and $\beta = y_0 + k_0$

$$f(\alpha, \beta) \approx f(x_0, y_0) + h_0 f_x(x_0, y_0) + k_0 f_y(x_0, y_0) = 0$$
$$g(\alpha, \beta) \approx g(x_0, y_0) + h_0 g_x(x_0, y_0) + k_0 g_y(x_0, y_0) = 0$$

$$\rightarrow \begin{bmatrix} f_x(x_i, y_i) & f_y(x_i, y_i) \\ g_x(x_i, y_i) & g_y(x_i, y_i) \end{bmatrix} \begin{bmatrix} h_i \\ k_i \end{bmatrix} = -\begin{bmatrix} f_i \\ g_i \end{bmatrix} \rightarrow \begin{bmatrix} h_i \\ k_i \end{bmatrix} = -J_i^{-1} \begin{bmatrix} f_i \\ g_i \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} h_i \\ k_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - J_i^{-1} \begin{bmatrix} f_i \\ g_i \end{bmatrix}$$

This is also known as Newton's Method for solving non-linear systems. The iterative scheme is very similar for the one-variable version of it where f(x) = 0.

Data Fitting:

The main problem is to fit the data $\{(x_i, y_i)\}_{i=1}^{N}$ via some function.

Polynomial Fitting:

$$\begin{bmatrix} 1 & x_1 & \dots & x_1^k \\ 1 & x_2 & \dots & x_2^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^k \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} \approx \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\rightarrow \min \left|\left|Xa - y\right|\right| \rightarrow \frac{d\left|\left|Xa - y\right|\right|}{da} = \frac{d(Xa - y)^T(Xa - y)}{da} = 2X^TXa - 2X^Ty = 0 \rightarrow a = (X^TX)^{-1}X^Ty$$

In Polynomial Fitting, the basis functions are as: $\{x^0, x^1, \dots, x^k\}$
The columns of matrix X is the basis functions applied to each data vector.

If the basis functions are to change, then matrix X changes and everything else stays the same!

Another way of solving the equations without using vector calculus is shown below:

let the basis functions be $\{\boldsymbol{\phi_0}, \boldsymbol{\phi_1}, \dots, \boldsymbol{\phi_k}\}$

we want to minimize the error of $\sum_{i=0}^{k} c_i \phi_i(x_j) \approx y_j$ by optimizing each c_i

objective:

$$\text{minimize error} = \sum_{j=1}^{N} \left(\sum_{i=0}^{k} c_i \phi_i(x_j) - y_j \right)^2 \rightarrow \frac{\partial \text{error}}{\partial c_m} = 2 \sum_{j=1}^{N} \phi_m(x_j) \left(\sum_{i=0}^{k} c_i \phi_i(x_j) - y_j \right) = 0$$

$$\sum_{j=1}^{N} y_j \phi_m(x_j) = \sum_{j=1}^{N} \sum_{i=0}^{k} c_i \phi_i(x_j) \phi_m(x_j)$$

let Φ_i be the function ϕ_i applied to each data vector.

The equation becomes as below:

$$y^T \Phi_m = \sum_{i=0}^{k} c_i \, \Phi_i^T \Phi_m \text{ for all m}$$

The equations for all m together gives the following:

$$\begin{bmatrix} \Phi_1^T\Phi_1 & \Phi_1^T\Phi_2 & \dots & \Phi_1^T\Phi_k \\ \Phi_2^T\Phi_1 & \Phi_2^T\Phi_2 & \dots & \Phi_2^T\Phi_k \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_k^T\Phi_1 & \Phi_k^T\Phi_2 & \dots & \Phi_k^T\Phi_k \end{bmatrix} \begin{bmatrix} c \\ c_1 \\ \vdots \\ c_k \end{bmatrix} = \begin{bmatrix} y^T\Phi_1 \\ y^T\Phi_2 \\ \vdots \\ y^T\Phi_k \end{bmatrix}$$

If the function vectors are orthogonal, then the matrix becomes diagonal, making solving it much easier. Use Gram-Schmidt to make the basis functions orthogonal.

Gram-Schmidt Procedure:

The objective is to turn basis $\{g_0, g_1, \dots, g_k\}$ into orthonormal basis $\{\phi_0, \phi_1, \dots, \phi_k\}$

1) $\phi_0 = g_0$
2) do the following for k steps

$$\phi_m = g_m - \sum_{i=0}^{m-1} \frac{g_m^T\phi_i}{\phi_i^T\phi_i}\phi_i \ \forall m > 0$$

3) $\phi_i = \dfrac{\phi_i}{\phi_i^T\phi_i} \ \forall m \geq 0$ to normalize

If the basis functions $\{\phi_0, \phi_1, \dots, \phi_k\}$ where orthonormal, then we get $c_i = y^T\phi_i$

In exponential data-fitting, we can change the data points by taking the natural logarithm to turn it into linear regression, which we already know the answer to!