

1 Solving Nonlinear Equations [by Root Finding $y = 0$]

Root Multiplicity, m : $0 = f(\bar{x}) = f'(\bar{x}) = \dots = f^{(m-1)}(\bar{x})$ (Simple Root: $m = 1$)

k -th Iteration Error: $e_k = x_k - \bar{x}$ Convergence Rate, r : $\lim_{k \rightarrow \infty} \frac{\|e_{k+1}\|}{\|e_k\|^r} = C$ ($0 < C < 1$ if $r = 1$)

1.1 One Dimension/Equation skipped a lot

Interval Bisection (Finding $y = 0$): $[f(a) < 0], [f(b) > 0], [f \text{ is cont.}] \Rightarrow \exists m \text{ s.t. } f(m) = 0$

Fixed-Point Iteration (Finding $y = x$): $\text{cont. } f(x) = 0 \Rightarrow \text{Find } g(x) = x \rightarrow x_{k+1} = g(x_k)$

\sim Banach-Fixed Point Theorem (there are many FP theorems)

- g is Contractive (over a domain): $\text{dist}(g(x), g(y)) \leq q \cdot \text{dist}(x, y) \quad q \in [0, 1)$
- $e_{k+1} = [x_{k+1} - \bar{x}] = [g(x_k) - g(\bar{x})] = g'(\xi_k)(x_k - \bar{x}) = g'(\xi_k)e_k$
- $\forall |g'(\xi_k)| < G < 1 \Rightarrow (|e_{k+1}| \leq G|e_k| \leq \dots \leq G^k|e_0|) \Rightarrow \lim_{k \rightarrow \infty} e_k = 0 \quad (G = \max g' \text{ over domain})$
- $\lim_{k \rightarrow \infty} |g'(\xi_k)| = \boxed{\begin{matrix} (0 < |g'(\bar{x})| < 1) \\ \text{(one contractive condition)} \end{matrix}} = C \quad (r = 1)$
- $\boxed{g'(\bar{x}) = 0} \Rightarrow [g(x_k) - g(\bar{x})] = \frac{g''(\xi_k)}{2}(x_k - \bar{x})^2 \Rightarrow \boxed{\left| \frac{g''(\bar{x})}{2} \right|} = C \quad (r = 2 \text{ if } \bar{x} \text{ is an } m = 2 \text{ root of } g)$

Newton's Method (Finding $y = 0$):

$$f(\bar{x}) = 0 = f(x_k + h_k) \approx f(x_k) + f'(x_k)h_k \Rightarrow \boxed{x_{k+1} = x_k + h_k = x_k - \frac{f(x_k)}{f'(x_k)}}$$

- $\boxed{g(x) \equiv x - \frac{f(x)}{f'(x)}} \Rightarrow g(\bar{x}) = \bar{x}, \boxed{g'(\bar{x}) = \frac{f(\bar{x})f''(\bar{x})}{f'(\bar{x})^2} = 0}, \boxed{r = 2} \quad (\text{if } \bar{x} \text{ is a simple root of } f)$
- $\bar{x} \text{ is an } m > 1 \text{ root of } f \Rightarrow \boxed{r = 1, C = 1 - 1/m} \quad (\text{proof not given})$

Secant Method/Linear Interpolation (Finding $y = 0$):

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \quad \text{Approx. } f'(x_k) \text{ with a secant line's slope} \Rightarrow \boxed{x_{k+1} = x_k + h_k = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k)}$$

- $\boxed{r = r_+ \approx 1.618} : r_+^2 - r_+ - 1 = 0 \quad (\text{proof hard})$
- Lower cost of iter. offsets the larger number of iter. compared to Newton's Method with derivatives

Inverse Parabolic Interpolation: Use 3 pts to approx. an inverse [sideways] parabola

1.2 m Dimensions/System of Equations stuff skipped

Newton's Method (Solving $\vec{y} = 0$):

$$\{J_f(\vec{x})\}_{ij} = \frac{\partial f_i(\vec{x})}{\partial x_j} : \quad J_f(\vec{x}_k)\vec{h}_k = -\vec{f}(\vec{x}_k) \Rightarrow \vec{x}_{k+1} = \vec{x}_k + \vec{h}_k = \vec{x}_k - J_f(\vec{x}_k)^{-1}\vec{f}(\vec{x}_k)$$

- $$\vec{g}(\vec{x}) \equiv \vec{x} - J_f(\vec{x})^{-1}\vec{f}(\vec{x}) \Rightarrow \begin{aligned} J_g(\vec{x}) &= \cancel{I - J_f(\vec{x})^{-1}J_f(\vec{x})} + \sum_{i=1}^n H_i(\vec{x})f_i(\vec{x}) \\ &\quad \text{(if } J_f(\vec{x}) \text{ is nonsingular)} \end{aligned} \quad \begin{array}{l} H_i = \text{component} \\ \text{matrix of the} \\ \text{tensor, } D_x J_f(\vec{x}) \end{array}$$

$$= \mathcal{O} \Rightarrow \boxed{r=2} \quad (\text{uh... idk})$$
- LU fact. of the Jacobian costs $\mathcal{O}(n^3)$

Broyden's [Secant Updating] Method (Solving $\vec{y} = 0$):

$$B_k\vec{h}_k = -\vec{f}(\vec{x}_k) \Rightarrow \vec{x}_{k+1} = \vec{x}_k + \vec{h}_k, \quad B_{k+1} = B_k + \frac{f(\vec{x}_{k+1})h_k^T}{h_k^T h_k} \quad (\text{cost is } \mathcal{O}(n^3))$$

- $B_{k+1}(\vec{x}_{k+1} - \vec{x}_k) = B_{k+1}\vec{h}_k = f(\vec{x}_{k+1}) - f(\vec{x}_k)$
- B_k factorization is updated to factorization of B_{k+1} at cost $\mathcal{O}(n^2)$ instead of directly from the above eq.
- Lower cost of iter. offsets the larger number of iter. compared to Newton's Method with derivatives

2 Optimizing [By Finding $\min f(\vec{x}) = f(\vec{x})$]

2.1 Function Shape and Convexity

Coercive: $\lim_{x \rightarrow \pm\infty} f(x) = \infty$ Unimodal: $\begin{array}{l} a \leq \bar{x} \leq b \\ x_1 < x_2 \end{array} : \quad \begin{array}{l} x_2 < \bar{x} \rightarrow f(x_1) > f(x_2) \\ \bar{x} < x_1 \rightarrow f(x_1) < f(x_2) \end{array}$

\exists global min f if

- cont. f on a closed and bounded set
- cont. f is coercive on a closed, unbounded set
- cont. f on a set and has a nonempty, closed, and bounded sublevel set
- domain set is unbounded: cont. f is coercive \Leftrightarrow all sublevel sets are bounded

f is convex [on a convex set] :

- any sublevel set is convex
- any local min. is a global min

f is strictly convex [on a convex set] :

- any local min. is a unique global min.
- if set is unbounded: f is coercive $\Leftrightarrow f$ has a unique global min.

2.2 Derivative Tests (Gradient, Jacobian, Hessian) and Lagrangians

Req. : $\boxed{\text{cont. } f(\bar{x}) = \min f, \text{ cont. } \vec{\nabla} f(\bar{x}), \text{ cont. } H_f(\bar{x})}$

Taylor's Theorem:
$$\begin{aligned} f(\bar{x} + \vec{s}) - f(\bar{x}) &= \vec{\nabla} f(\bar{x} + \alpha_1 \vec{s}) \cdot \vec{s} = \vec{\nabla} f(\bar{x}) \cdot \vec{s} + \frac{1}{2} \langle \vec{s} | H_f(\bar{x} + \alpha_2 \vec{s}) | \vec{s} \rangle \geq 0 \\ f(\bar{x} + s\hat{u}) - f(\bar{x}) &= \vec{\nabla} f(\bar{x} + \alpha_1 s\hat{u}) \cdot s\hat{u} = \vec{\nabla} f(\bar{x}) \cdot \vec{s} + \frac{s^2}{2} \langle \hat{u} | H_f(\bar{x} + \alpha_2 \vec{s}) | \hat{u} \rangle \end{aligned}$$

- $\lim_{s \rightarrow 0} \left(\frac{f(\bar{x} + \vec{s}) - f(\bar{x})}{s} = \vec{\nabla} f(\bar{x} + \alpha_1 s\hat{u}) \cdot \hat{u} \right) \Rightarrow \left(\vec{\nabla} f(\bar{x}) \cdot \hat{u} \geq 0 \rightarrow \boxed{\vec{\nabla} f(\bar{x}) \cdot \vec{s} \geq 0} \right)$, $\boxed{\text{Cauchy-Schwarz} \rightarrow \max \vec{\nabla} f(\bar{x}) \cdot \hat{u} \text{ if } \vec{u} = \vec{\nabla} f(\bar{x})}$
- $\boxed{\vec{u} = \mp \vec{\nabla} f(\bar{x})} \Rightarrow \lim_{s \rightarrow 0} \left(\frac{f(\bar{x} + \vec{s}) - f(\bar{x})}{s} = \mp \frac{\vec{\nabla} f(\bar{x} + \alpha_1 s\hat{u}) \cdot \vec{\nabla} f(\bar{x})}{\|\vec{\nabla} f(\bar{x})\|} \right) = \mp \|\vec{\nabla} f(\bar{x})\| \leq 0$ $\boxed{\text{if } \pm \vec{\nabla} f(\bar{x}) \neq 0, \text{ its dir. is an ascent/descent.}}$
- $\lim_{s \rightarrow 0} \left(\frac{f(\bar{x} + \vec{s}) - f(\bar{x}) + f(\bar{x} - \vec{s}) - f(\bar{x})}{s^2} = \frac{\langle \hat{u} | H_f(\bar{x} + \alpha_2 \vec{s}) + H_f(\bar{x} - \alpha_3 \vec{s}) | \hat{u} \rangle}{2} \right) = \langle \hat{u} | H_f(\bar{x}) | \hat{u} \rangle \Rightarrow \boxed{\langle \vec{s} | H_f(\bar{x}) | \vec{s} \rangle \geq 0}$

2.2.1 Unconstrained Optimization Conditions

- $\boxed{f(\bar{x}) = \min f} \Leftrightarrow \left(\begin{array}{l} \vec{\nabla} f(\bar{x}) \cdot \vec{s} \geq 0, \vec{\nabla} f(\bar{x}) \cdot -\vec{s} \geq 0 \\ \Rightarrow \boxed{\vec{\nabla} f(\bar{x}) = 0} \end{array} \right), \quad \vec{u} = -\vec{\nabla} f(\bar{x}) \Rightarrow \boxed{\vec{\nabla} f(\bar{x}) = 0}, \quad \boxed{\begin{array}{l} \text{(for strict convexity)} \\ \langle \vec{s} | H_f(\bar{x}) | \vec{s} \rangle > 0 \end{array}} \right)$

Optimization $f: \mathbb{R}^n \rightarrow \mathbb{R} \quad \boxed{\min f(\vec{x}) = y}$

$$\boxed{\mathcal{L}(\vec{x}) = f(\vec{x})}, \quad \boxed{\nabla \mathcal{L}(\bar{x}) = 0}, \quad \boxed{H_{\mathcal{L}} = \nabla_{xx} \mathcal{L}: \langle s | H_{\mathcal{L}}(\bar{x}) | s \rangle > 0} \Rightarrow \boxed{y = f(\bar{x})}$$

2.2.2 Constrained Optimization Conditions

- $\boxed{\vec{s} = \text{feasible direction}} \Leftrightarrow \left(\boxed{\vec{\nabla} f(\bar{x}) \cdot \vec{s} \geq 0}, \boxed{\langle \vec{s} | H_f(\bar{x}) | \vec{s} \rangle \geq 0} \right)$

Optimization $\begin{array}{l} f: \mathbb{R}^n \rightarrow \mathbb{R} \\ g: \mathbb{R}^n \rightarrow \mathbb{R}^m \\ h: \mathbb{R}^n \rightarrow \mathbb{R}^p \end{array} \quad \min f(\vec{x}) = y \quad \text{w/} \quad \left(\begin{array}{l} \vec{g}(\vec{x}) = 0 \\ \vec{h}(\vec{x}) \leq 0 \end{array} \right) \quad \begin{array}{l} \text{active: } h_i(\bar{x}) = 0 \\ \text{inactive: } h_i(\bar{x}) < 0 \rightarrow \bar{\mu}_i = 0 \end{array}$ (see KKT)

$$\begin{aligned} \mathcal{L}(\bar{x}, \bar{\lambda}, \bar{\mu}) &= f(\bar{x}) + \bar{\lambda} \cdot \vec{g}(\bar{x}) + \bar{\mu} \cdot \vec{h}(\bar{x}) \\ &= f + \sum_i^m \lambda_i g_i + \sum_i^p \cancel{\mu_i h_i} \quad \text{(KKT) if } \bar{x} = \bar{x} \end{aligned}, \quad \nabla \mathcal{L}(\bar{x}, \bar{\lambda}, \bar{\mu}) = \begin{pmatrix} \nabla_x \mathcal{L} = 0 \\ \nabla_{\lambda} \mathcal{L} = 0 \\ \nabla_{\mu} \mathcal{L} \leq 0 \end{pmatrix} = \begin{pmatrix} \nabla f(\bar{x}) + J_g^T(\bar{x}) \bar{\lambda} + J_h^T(\bar{x}) \bar{\mu} \\ \vec{g}(\bar{x}) \\ \vec{h}(\bar{x}) \end{pmatrix}$$

$$H_{\mathcal{L}}(\bar{x}, \bar{\lambda}, \bar{\mu}) = \begin{pmatrix} \nabla_{xx} \mathcal{L} & \nabla_{x\lambda} \mathcal{L} & \nabla_{x\mu} \mathcal{L} \\ \nabla_{\lambda x} \mathcal{L} & \nabla_{\lambda\lambda} \mathcal{L} & \nabla_{\lambda\mu} \mathcal{L} \\ \nabla_{\mu x} \mathcal{L} & \nabla_{\mu\lambda} \mathcal{L} & \nabla_{\mu\mu} \mathcal{L} \end{pmatrix} = \begin{pmatrix} \nabla_{xx} \mathcal{L} & J_g^T & J_h^T \\ J_g & 0 & 0 \\ J_h & 0 & 0 \end{pmatrix}, \quad \nabla_{xx} \mathcal{L}(\bar{x}, \bar{\lambda}, \bar{\mu}) = H_f + \sum_i^m \bar{\lambda}_i H_{g_i} + \sum_i^{\text{act} \leq p} \bar{\mu}_i H_{h_i}$$

(can't be pos. def.)

- Assume $m \leq n$ (not overdetermined)

- $y = f(\bar{x}) : \nabla \mathcal{L}(\bar{x}, \bar{\lambda}, \bar{\mu}) \dots$, $p = 0 : Z^T (\nabla_{xx} \mathcal{L}) Z > 0$ col. of Z = basis of $\text{null}(J_g)$

- Assume h_i don't contradict each other? Assume full rank($J_{h_{\text{act}}}$)

- $y = f(\bar{x}) : \nabla \mathcal{L}(\bar{x}, \bar{\lambda}, \bar{\mu}) \dots$, $p > 0$, Karush-Kuhn-Tucker (KKT) : $\bar{\mu}_i \geq 0$, $\bar{\mu}_i h_i(\bar{x}) = 0$ (2nd deriv. cond. not given)

2.3 Unconstrained One Dimension/Independent Variable

[Interval] Golden-Section Search (if Unimodal): $\tau^2 = 1 - \tau = .382$, $r = 1$, $C = \tau$

$$[a < x_1 < x_2 < b] : \begin{cases} f(x_1) > f(x_2) \rightarrow [x_1 < x_2 < x_1 + \tau(b - x_1) < b] \\ f(x_1) \leq f(x_2) \rightarrow [a < a + (1 - \tau)(x_2 - a) < x_1 < x_2] \end{cases}$$

Newton's Method: $f(\bar{x}) = f(x + h) \approx f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 = g(h)$

$$g\left(\frac{-b}{2a}\right) = \min g \text{ (or max)} \Rightarrow x_{k+1} = x_k + h_k = x_k - \frac{b}{2a} = x_k - \frac{f'(x)}{f''(x)} , r = 2$$

Successive Linear Interpolation [Secant Method]: Not useful, since lines have no unique minimum

Successive Parabolic Interpolation: Use 3 pts to approx. a parabola w/ $r = 1.324$ (not guaranteed)

2.4 Unconstrained m -Dimensions/Independent Variables

Steepest [Gradient] Descent/Line Search (go down $-\nabla f(\vec{x}_k)$):

$$\phi(\alpha) = f(\vec{x} - \alpha \vec{\nabla} f(\vec{x})) , \phi(\alpha_k) = \min \phi \Rightarrow \vec{x}_{k+1} = \vec{x}_k - \alpha_k \vec{\nabla} f(\vec{x}_k) , r = 1 , C_{\text{varies}}$$

- $\vec{\nabla} f(\vec{x}_k) \cdot \vec{\nabla} f(\vec{x}_{k+1}) = 0 \Rightarrow$ Path will zig-zag to the min. (not too efficient)

Newton's Method: $f(\bar{x}) = f(\vec{x} + \vec{h}) \approx f(\vec{x}) + \vec{\nabla} f(\vec{x}) \cdot \vec{h} + \frac{1}{2} \langle \vec{h} | H_f(\vec{x}) | \vec{h} \rangle$

$$H_f(\vec{x}_k) \vec{h}_k = -\vec{\nabla} f(\vec{x}_k) \Rightarrow \vec{x}_{k+1} = \vec{x}_k + \vec{h}_k , r = 2$$

BFGS [Secant Updating] Method: $B_k \vec{h}_k = -\vec{\nabla} f(\vec{x}_k) , \vec{y}_k = \vec{\nabla} f(x_{k+1}) - \vec{\nabla} f(x_k)$

$$\Rightarrow \vec{x}_{k+1} = \vec{x}_k + \vec{h}_k , B_{k+1} = B_k + \frac{|y_k\rangle\langle y_k|}{\langle y_k | h_k \rangle} - \frac{B_k |h_k\rangle\langle h_k| B_k}{\langle h_k | B_k | H_k \rangle} \quad (\text{cost is } \mathcal{O}(n^3))$$

- Preserves symmetry and pos. def.
- B_k factorization is updated to factorization of B_{k+1} at cost $\mathcal{O}(n^2)$ instead of directly from the above eq.
- Lower cost of iter. offsets the larger number of iter. compared to Newton's Method with derivatives

Conjugate Gradient [Line Search] :

$$\boxed{\vec{h}_{k+1} = \vec{\nabla} f(\vec{x}_{k+1}) - \frac{\vec{\nabla} f(\vec{x}_{k+1}) \cdot \vec{\nabla} f(\vec{x}_{k+1})}{\vec{\nabla} f(\vec{x}_k) \cdot \vec{\nabla} f(\vec{x}_k)} \vec{h}_k} \quad (\text{Fletcher and Reeves}) \Rightarrow \boxed{\vec{x}_{k+1} = \vec{x}_k - \alpha_k \vec{h}_k}$$

- Seq. of conj. (where $(a, b) = \langle a | H_f | b \rangle$) search directions implicitly accumulates info. about H_f .
- Better for nonlin. to use
$$\boxed{\vec{h}_{k+1} = \vec{\nabla} f(\vec{x}_{k+1}) - \frac{\vec{\nabla} f(\vec{x}_{k+1}) \cdot \vec{\nabla} f(\vec{x}_{k+1}) - \vec{\nabla} f(\vec{x}_k) \cdot \vec{\nabla} f(\vec{x}_{k+1})}{\vec{\nabla} f(\vec{x}_k) \cdot \vec{\nabla} f(\vec{x}_k)} \vec{h}_k} \quad (\text{Polak and Ribiere})$$
- Restart algorithm after n iter. using last point as the new initial; a quadratic func. finishes after at most n iter.

2.4.1 Nonlinear Least Squares, $\{ \min \|\vec{r}(\vec{x})\|^2 : \vec{f}(\vec{a}, \vec{x}) + \vec{r}(\vec{x}) = \vec{b} \}$

Linear Least Squares	Nonlinear Least Squares
$\begin{pmatrix} \vdots \\ -\vec{a}_i \\ \vdots \end{pmatrix} \begin{pmatrix} \\ \vec{x} \\ \end{pmatrix} + \begin{pmatrix} \\ \vec{r} \\ \end{pmatrix} = \begin{pmatrix} \\ \vec{b} \\ \end{pmatrix} \Rightarrow \begin{pmatrix} \\ \vec{f}(\vec{a}, \vec{x})_i \\ \end{pmatrix} + \begin{pmatrix} \\ \vec{r} \\ \end{pmatrix} = \begin{pmatrix} \\ \vec{b} \\ \end{pmatrix}$	

$\boxed{\phi(\vec{x}) \equiv \frac{1}{2} \vec{r} \cdot \vec{r}}, \quad \boxed{-\vec{\nabla} \phi(\vec{x}) = -J_r^T \vec{r}}$	<p>Newton's Method</p>	$\boxed{\vec{x}_{k+1} = \vec{x}_k + \vec{h}_k}$
$\boxed{H_\phi(\vec{x}) = J_r^T J_r + \sum_i H_{r_i} \vec{r}_i}$	$: \quad \boxed{H_\phi(\vec{x}_k) \vec{h}_k = -\vec{\nabla} \phi(\vec{x}_k)} \Rightarrow$ <p>(usually expensive to compute)</p>	

Gauss-Newton Method: If \vec{r} is small $\Rightarrow H_\phi \approx J_r^T J_r \Rightarrow \boxed{J_r^T (J_r \vec{h}_k) = -J_r^T \vec{r}(\vec{x}_k)}$ System of Normal Equations

Levenberg-Marquardt Method (Gauss-Newton + Line Search):

$\boxed{(J_r^T J_r + \mu_k I) \vec{h}_k = -J_r^T \vec{r}(\vec{x}_k) \Rightarrow \vec{x}_{k+1} = \vec{x} + \vec{h}_k}$ $\Rightarrow \boxed{\begin{pmatrix} J_r^T(\vec{x}) & \sqrt{\mu_k} I \end{pmatrix} \begin{pmatrix} J_r(\vec{x}) \\ \sqrt{\mu_k} I \end{pmatrix} \vec{h}_k = \begin{pmatrix} J_r^T(\vec{x}) & \sqrt{\mu_k} I \end{pmatrix} \begin{pmatrix} -\vec{r}(\vec{x}_k) \\ 0 \end{pmatrix}}$	<p><u>Regularization</u></p> <ul style="list-style-type: none"> • Replacing $H_{r_i} \vec{r}_i$ terms with a scalar mult. of I. • Shifting the Gauss-Newton Hessian to make it pos. def (or boosting its rank).
---	---

2.5 Constrained m -Dimensions/Independent Variables

<p>Newton's Method</p> $\boxed{H_{\mathcal{L}} \vec{h}_k = -\vec{\nabla} \mathcal{L}}$	<table border="0" style="width: 100%;"> <tr> <td style="width: 50%; text-align: center;"> <p>KKT Matrix (Eq. Constr)</p> $\begin{pmatrix} \nabla_{xx} \mathcal{L} & J_g^T \\ J_g & 0 \end{pmatrix} \begin{pmatrix} \vec{s}_k \\ \vec{\delta}_k \end{pmatrix} = - \begin{pmatrix} \nabla f(\vec{x}) + J_g^T(\vec{x}) \bar{\lambda} \\ \vec{g}(\vec{x}) \end{pmatrix}$ $\boxed{\begin{pmatrix} B & J^T \\ J & 0 \end{pmatrix} \begin{pmatrix} s \\ \delta \end{pmatrix} = - \begin{pmatrix} w \\ g \end{pmatrix}}$ </td> <td style="width: 50%; text-align: center;"> <p>[Sequential] Quadratic Programming (SQP) Problem</p> $\min_s \left(\vec{s}_k \cdot \vec{\nabla}_x \mathcal{L} + \frac{1}{2} \langle \vec{s}_k \vec{\nabla}_{xx} \mathcal{L} \vec{s}_k \rangle \right)$ <p>s.t. $J_g(\vec{x}_k) \vec{s}_k + \vec{g}(\vec{x}_k) = 0$</p> </td> </tr> </table>	<p>KKT Matrix (Eq. Constr)</p> $\begin{pmatrix} \nabla_{xx} \mathcal{L} & J_g^T \\ J_g & 0 \end{pmatrix} \begin{pmatrix} \vec{s}_k \\ \vec{\delta}_k \end{pmatrix} = - \begin{pmatrix} \nabla f(\vec{x}) + J_g^T(\vec{x}) \bar{\lambda} \\ \vec{g}(\vec{x}) \end{pmatrix}$ $\boxed{\begin{pmatrix} B & J^T \\ J & 0 \end{pmatrix} \begin{pmatrix} s \\ \delta \end{pmatrix} = - \begin{pmatrix} w \\ g \end{pmatrix}}$	<p>[Sequential] Quadratic Programming (SQP) Problem</p> $\min_s \left(\vec{s}_k \cdot \vec{\nabla}_x \mathcal{L} + \frac{1}{2} \langle \vec{s}_k \vec{\nabla}_{xx} \mathcal{L} \vec{s}_k \rangle \right)$ <p>s.t. $J_g(\vec{x}_k) \vec{s}_k + \vec{g}(\vec{x}_k) = 0$</p>
<p>KKT Matrix (Eq. Constr)</p> $\begin{pmatrix} \nabla_{xx} \mathcal{L} & J_g^T \\ J_g & 0 \end{pmatrix} \begin{pmatrix} \vec{s}_k \\ \vec{\delta}_k \end{pmatrix} = - \begin{pmatrix} \nabla f(\vec{x}) + J_g^T(\vec{x}) \bar{\lambda} \\ \vec{g}(\vec{x}) \end{pmatrix}$ $\boxed{\begin{pmatrix} B & J^T \\ J & 0 \end{pmatrix} \begin{pmatrix} s \\ \delta \end{pmatrix} = - \begin{pmatrix} w \\ g \end{pmatrix}}$	<p>[Sequential] Quadratic Programming (SQP) Problem</p> $\min_s \left(\vec{s}_k \cdot \vec{\nabla}_x \mathcal{L} + \frac{1}{2} \langle \vec{s}_k \vec{\nabla}_{xx} \mathcal{L} \vec{s}_k \rangle \right)$ <p>s.t. $J_g(\vec{x}_k) \vec{s}_k + \vec{g}(\vec{x}_k) = 0$</p>		

Direct Solution: KKT Matrix is sym. and sparse \rightarrow solve for \vec{h}_k using sym. indef. factorization w/ some pivoting

(Column-Space)

Range-Space Method: $Bs = -w - J^T \delta$, $Js = -g \rightarrow JB^{-1}(-w - J^T \delta) = -g$
 $\rightarrow (JB^{-1}J^T)\delta = g - JB^{-1}w$

- Solve for δ , then for s .
- Forming $(JB^{-1}J^T)_{m \times m}$ leads to issues similar to forming $A^T A$ (loss of info. and degrades conditioning).
- B must be nonsingular and J full rank.
- Useful if m is small.

Null-Space Method: $J^T = (Q_{\parallel} \ Q_{\perp}) \begin{pmatrix} R \\ 0 \end{pmatrix} \quad (Q_{\parallel} \in \mathbb{R}^{n \times m}) \Rightarrow \begin{cases} JQ_{\parallel} = R^T \\ JQ_{\perp} = 0 \end{cases}$

Find u_{\parallel} : $Js \equiv (JQ_{\parallel}u_{\parallel} + \cancel{JQ_{\perp}u_{\perp}}) = \boxed{R^T u_{\parallel} = -g}$

Find u_{\perp} : $Q_{\perp}^T(Bs + J^T \delta = -w) \rightarrow (Q_{\perp}^T BQ_{\parallel})u_{\parallel} + (Q_{\perp}^T BQ_{\perp})u_{\perp} = -Q_{\perp}^T w - \cancel{(JQ_{\perp})^T} \delta$
 $\boxed{(Q_{\perp}^T BQ_{\perp})u_{\perp} = -Q_{\perp}^T w - (Q_{\perp}^T BQ_{\parallel})u_{\parallel}}$

Find δ : $Q_{\parallel}^T(J^T \delta = -w - Bs) \rightarrow \boxed{R\delta = -Q_{\parallel}^T w - Q_{\parallel}^T B(Q_{\parallel}u_{\parallel} - Q_{\perp}u_{\perp})}$

- Near a min., $(Q_{\perp}^T BQ_{\perp})$ can be Cholesky factored.
- Avoids issues with loss of info. and degraded conditioning.
- J must be full rank and R nonsingular.
- Useful if m is large, so $n - m$ is small.

Decent Initial $\vec{\lambda}_0$ Guess Given an \vec{x}_0 : $J_g^T(\vec{x}_0)\vec{\lambda}_0 + \vec{r} = -\vec{\nabla} f(\vec{x}_0) \quad (\text{Linear Least Sq.})$

Penalty Func. Method

$\lim_{\rho \rightarrow \infty} \vec{x}_{\rho} = \vec{x}$ (not explained)

(“Under approp. conds.”)

One Simple Function
(Ill-conditioned $\rho \gg 1$) : $\min_{\vec{x}} \phi_{\rho}(\vec{x}) = f(\vec{x}) + \frac{1}{2}\rho \|g(\vec{x})\|^2$

Augmented Lagrangian
(Less Ill-conditioned) : $\min_{\vec{x}} \mathcal{L}_{\rho}(\vec{x}) = f(\vec{x}) + \vec{\lambda}_0 \cdot \vec{g}(\vec{x}) + \frac{1}{2}\rho \|g(\vec{x})\|^2$

Barrier Func. Method

$\lim_{\rho \rightarrow 0} \vec{x}_{\rho} = \vec{x}$

(“Under approp. conds.”)

Inverse : $\min_{\vec{x}} \phi_{\rho}(\vec{x}) = f(\vec{x}) - \rho \sum_i^p \frac{1}{h_i(\vec{x})}$

Logarithmic : $\min_{\vec{x}} \phi_{\rho}(\vec{x}) = f(\vec{x}) - \rho \sum_i^p \log(-h_i(\vec{x}))$

(For Ineq. Constr.)

- Along with line search and trust region (not explained), a merit func. - using perhaps a penalty func. - can be used to make an algorithm more robust.
- An active set strategy (not explained) can be used with an SQP method for ineq.-constr. problems.
- A penalty method penalizes points that violates constraints, but doesn't avoid them. Barrier methods do.

3 [Polynomial] Interpolation, $f(t_i) = \hat{f}(t_i) = \sum_j x_j \phi_j(t_i)$

$$\hat{f}(t_i) = \sum_j x_j \phi_j(t_i) \quad \left| \begin{array}{l} \det(A) \neq 0 \\ \text{Given } \vec{\phi}, \\ \text{solve for } \vec{x} \end{array} \right| \quad A\vec{x} = \begin{pmatrix} \vdots \\ -\vec{\phi}(t_i) \\ \vdots \end{pmatrix} \begin{pmatrix} | \\ \vec{x} \\ | \end{pmatrix} = \vec{y} = \begin{pmatrix} \vdots \\ f(t_i) \\ \vdots \end{pmatrix}$$

- Runge Phenom.: As n increases, evenly-spaced t_i could produce a high-dimensional polynomial $\hat{f}(t)$ that tends to be extremely wavy near the endpoints (like Gibbs phenom.). Choosing t_i to be Chebyshev nodes between the two endpoints mitigates this.
- Interpolation w/ other func. like rationals are possible.
- Error: $\max_{t \in [t_1, t_n]} \left| \hat{f} - f = \frac{f^{(n)}(\xi)}{n!} \prod_i (t - t_i) \right| \leq \left| \max_{t \in [t_1, t_n]} \frac{f^{(n)}(t)}{n!} \right| \left| \frac{(n-1)! h^n}{4} \right| = \boxed{\max_{t \in [t_1, t_n]} \left| f^{(n)}(t) \frac{h^n}{4n} \right|} \rightarrow \text{error decreases if } f^{(n)} \text{ is well behaved}$

3.1 Taylor Series Polynomial Interpolation

$$\begin{aligned} \hat{f}_n(t) &= f(t_0) + f'(t_0)(t - t_0) + \frac{f''(t_0)}{2}(t - t_0)^2 + \dots + \frac{f^{(n-1)}(t_0)}{(n-1)!}(t - t_0)^{n-1} \\ \hat{f}_n(t + h) &= f(t) + f'(t)h + \frac{f''(t)}{2}h^2 + \dots + \frac{f^{(n-1)}(t)}{(n-1)!}h^{n-1} \end{aligned}$$

- Can interpolate an n -polynomial from $n + 1$ points/derivatives/info.

3.2 Monomial Basis Functions \rightarrow Vandermonde Matrix

$$\begin{aligned} \vec{\phi}(t) &= (1, t, t^2, \dots, t^{n-1})^T \\ \hat{f}(t) &= x_1 + x_2 t + \dots + x_n t^{n-1} \end{aligned}$$

(Full, Dense
Vandermonde Matrix)

$$\begin{pmatrix} 1 & t_1 & \dots & t_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & t_n & \dots & t_n^{n-1} \end{pmatrix} \begin{pmatrix} \vdots \\ x_i \\ \vdots \end{pmatrix} = \vec{y}$$

- Solved with $\mathcal{O}(n^3)$ work using Gauss. Elim. ($\mathcal{O}(n^2)$ is possible with other tech.).
- Ill-conditioned since successive t^j look the same at higher j .

3.3 Lagrange Basis Functions (Fund. Polynomials) \rightarrow Identity Matrix

$$\begin{aligned} l(t) &= (t - t_1)(t - t_2) \dots (t - t_n) \\ w_j &= (t_j - t_j)/l(t_j) \quad (\text{barycentric weights}) \end{aligned}$$

(Diag. Iden. Matrix)

$$\begin{pmatrix} 1 & 0 & \dots \\ 0 & 1 & \ddots \\ \vdots & \ddots & \ddots \end{pmatrix} \vec{x} = \vec{y}$$

- Finding w_j is $\mathcal{O}(n^2)$ work.
- Finding $\hat{f}(t)$ from w_j 's is $\mathcal{O}(n)$ work.
- Updating with an extra point (t_{n+1}, y_{n+1}) is $\mathcal{O}(n)$ work by changing $w_j = w_j/(t_j - t_{n+1})$ and finding w_{n+1} .
- Basis func. are more varied \rightarrow better-conditioned.

$$\hat{f}(t) = \vec{x} \cdot \vec{\phi}(t) = l(t) \left[x_1 \frac{w_1}{t - t_1} + \dots + x_n \frac{w_n}{t - t_n} \right]$$

$$\hat{f}(t_j) = x_j = y_i$$

$$\int_{t_1}^{t_n} \hat{f}(t) dt = \sum_{i=1}^n y_i \int_{t_1}^{t_n} \phi_i(t) dt$$

3.4 Newton Basis Functions → Low. Triang. Matrix

$$\left[\begin{array}{l} \phi_j(t) = (t - t_1)(t - t_2) \dots (t - t_{j-1}) \\ \vec{\phi}(t) = [1, (t - t_1), (t - t_1)(t - t_2), \dots]^T \\ \hat{f}(t) = x_1 + x_2(t - t_1) + \dots + x_n \phi_n(t) \end{array} \right] \quad \left| \quad \begin{array}{l} \text{(Low. Triang. Matrix)} \\ \begin{pmatrix} 1 & 0 & 0 & \dots \\ 1 & t_1 - t_2 & 0 & \dots \\ 1 & t_3 - t_2 & (t_3 - t_1)(t_3 - t_2) & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ x_i \end{pmatrix} = \vec{y} \end{array} \right.$$

- For. sub. is $\mathcal{O}(n^2)$.
- Cond. of A depends on ordering of points → best to order points from their dist. to their mean/other num.
- Basis func. are more varied → better-conditioned.

Incremental Updating Newton Interpolation:

$$\hat{f}_{n+1}(t) = \hat{f}_n(t) + x_{n+1} \phi_{n+1}(t)$$

$$y_{n+1} = \hat{f}_{n+1}(t_{n+1})$$

$$= \hat{f}_n(t_{n+1}) + x_{n+1} \phi_{n+1}(t_{n+1})$$

$$\Rightarrow \boxed{\hat{f}_{j+1}(t) = \hat{f}_j(t) + \frac{y_{j+1} - \hat{f}_j(t_{j+1})}{\phi_{j+1}(t_{j+1})} \phi_{j+1}(t)}$$

Divided Differences Newton Interpolation:

$$g[t_1, \dots, t_k] \equiv \frac{g[t_2, \dots, t_k] - g[t_1, \dots, t_{k-1}]}{t_k - t_1}$$

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} g[t_1] \\ g[t_1, t_2] \\ g[t_1, t_2, t_3] \\ \vdots \end{pmatrix}$$

- Also costs $\mathcal{O}(n^2)$.
- Less prone to over/underflow.

3.5 Orthogonal Polynomial Basis (no method given)

Inner Product: $\langle \vec{u} | \vec{v} \rangle_{ab}^w = \int_a^b [u(t)v(t)] w(t) dt$

Orthogonal Polynomials: $\langle u_i | u_j \rangle = \delta_{ij}$

Three-Term Recurrence: $\boxed{\hat{f}_{k+1}(t) = [A(k)t + B(k)] \hat{f}_k(t) - C(k) \hat{f}_{k-1}(t)}$ ($A(k) \neq 0$)

3.6 Piecewise [Hermite] Cubic Interpolation

Piecewise Cubic:

n knots/pts. $\Rightarrow n - 1$ cubics

$$\Rightarrow \boxed{4(n - 1) \text{ param./eq.}}$$

Hermite Interpolation:

Using k -th derivatives as info.

Extra equations can be used for monotonicity/convexity.

Hermite Cubic Interpolation:

Continuous 0th and 1st derivatives; $n - 1$ cubics

$$\Rightarrow [2(n - 1)]_{1\text{st deriv. eq}} + [n - 2]_{2\text{nd deriv. eq.}}$$

$$= \boxed{3n - 4 \text{ eq.} \Rightarrow n \text{ free/extra param./eq.}}$$

3.7 Piecewise Cubic [Spline] Interpolation

Spline:

A piecewise func. of n -polynomials that is n -differentiable (of differentiability class C^{n-1} , or $n - 1$ cont. differentiable).

Cubic Spline Interpolation:

Cont. 0th, 1st, and 2nd derivatives; $n - 1$ cubics

$$\Rightarrow [2(n - 1)]_{1\text{st}} + [n - 2]_{2\text{nd}} + [n - 2]_{3\text{rd}}$$

$$= \boxed{4n - 6 \text{ eq.} \Rightarrow 2 \text{ free/extra param./eq.}}$$

B -splines (basis func.):

Orthog. $\{\phi_j(t)\}$ are j -poly. splines w/ local compact support and look like bells. (not much detail here).

4 Numerical Integration/Quadrature, $I(f) \equiv \int_a^b f(x) dx$

4.1 ∞ -Norm and Condition Number

Function ∞ -Norm:

[Abs.] Integration Condition Number if \hat{b} :

$$\|f(x)\|_{\infty} = \max_{x \in [a,b]} f(x)$$

$$\left| \int_a^{\hat{b}} f(x) dx - \int_a^b f(x) dx \right| = \left| \int_b^{\hat{b}} f(x) dx \right| \leq (\hat{b} - b) \|f(x)\|_{\infty}$$

[Abs.] Integration Condition Number if \hat{f} :

[Rel.] Integration Condition Number if \hat{f} :

$$\begin{aligned} \left| \int_a^b \hat{f}(x) - f(x) dx \right| &\leq \int_a^b |\hat{f}(x) - f(x)| dx \\ &\leq (b-a) \|\hat{f}(x) - f(x)\|_{\infty} \\ \left| \frac{\Delta I}{\Delta f} \right| &\leq \boxed{b-a} \end{aligned}$$

$$\begin{aligned} \left| \frac{\Delta I/I}{\Delta f/f} \right| &\leq \frac{(b-a) / \left| \int_a^b f(x) dx \right|}{1/\|f(x)\|_{\infty}} \\ &= \frac{(b-a) \|f(x)\|_{\infty}}{\left| \int_a^b f(x) dx \right|} \end{aligned}$$

4.2 1-D [Interpolary] Quadrature Rule for $f \approx \hat{f}$

$$\hat{f} \in P_{n-1} : \hat{f}(x) = \left(\begin{array}{c} \vec{y} \cdot \vec{\phi}(x) = \sum_{i=1}^n f(x_i) \phi_i(x) \\ \text{(Lagrange Basis Vectors)} \end{array} \right) = \left(\begin{array}{c} \sum_{j=1}^n c_j x^{j-1} \\ \text{(Monomial Basis Vectors)} \end{array} \right) \quad \begin{array}{l} \bullet x_1 < \dots < x_n \\ \bullet f(x_i) = \hat{f}(x_i) \end{array}$$

$$\Rightarrow \boxed{Q_n(f) \equiv I(\hat{f}) = \int_a^b \hat{f}(x) dx = \sum_{i=1}^n f(x_i) \int_a^b \phi_i(x) dx = \sum_{i=1}^n f(x_i) w_i}$$

- $x_i, w_i \rightarrow 2n$ max param.
- $a \leq x_1 < \dots < x_n \leq b$
- closed if equality, open if not

Method of Undetermined Coefficients (MUC) / System of Moment Equations

$$\begin{aligned} \int_a^b \left(\sum_{j=1}^n c_j x^{j-1} \right) dx &= \sum_{i=1}^n \left(\sum_{j=1}^n c_j x_i^{j-1} \right) w_i \\ \sum_{j=1}^n c_j \left(\int_a^b x^{j-1} dx \right) &= \sum_{j=1}^n c_j \left(\sum_{i=1}^n x_i^{j-1} w_i \right) \Rightarrow \sum_{i=1}^n x_i^{j-1} w_i = \frac{b^j}{j} - \frac{a^j}{j} \\ &\equiv z_j \\ \bullet \quad z_1 &= \sum w_i = b-a \end{aligned}$$

(maybe some dot product to isolate terms)

(Vandermode Matrix)

$$\begin{bmatrix} 1 & 1 & 1 & \dots \\ x_1 & x_2 & x_3 & \dots \\ x_1^2 & x_2^2 & x_3^2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \vec{w} = \vec{z}$$

$$\begin{bmatrix} 1 & 0 & 0 & \dots \\ a & 1 & 1 & \dots \\ \frac{a^2}{2} & x_1 & x_2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ \vec{w} \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 \\ b \\ \frac{b^2}{2} \\ \vdots \end{bmatrix}$$

Error I : $|\Delta I| \leq (b-a) \|f - \hat{f}\|_{\infty} \leq \frac{b-a}{4n} h^n \|f^{(n)}\|_{\infty} \leq \frac{h^{n+1}}{4} \|f^{(n)}\|_{\infty} \rightarrow$ error decreases if $f^{(n)}$ is well behaved

Error Q_n : $g \approx f \rightarrow |Q_n(f) - Q_n(g)| \leq \boxed{\sum |w_i| \|f-g\|_{\infty}} \Rightarrow \boxed{\forall w_i \geq 0 \rightarrow \text{cond}(Q_n) = b-a}$
 (otherwise using Q_n might be unstable.)

[Rule] Degree, d : $\forall p(x) \in P_d$, rule $Q(p) = I(p)$, but not $\forall p \in P_{d+1}$

Newton-Cotes Quadrature [Rule]: n evenly-spaced $x_i \rightarrow n$ param. for w_i

Midpoint Rule (Q_1) :	$M(f) = \frac{b-a}{1} f(\frac{a+b}{2})$	$\vec{w} = (b-a)[1]^T$
Trapezoidal Rule (Q_2) :	$T(f) = \frac{b-a}{2} [f(a) + f(b)]$	$\vec{w} = (b-a)[\frac{1}{2}, \frac{1}{2}]^T$
Simpsons's Rule (Q_3) :	$S(f) = \frac{b-a}{6} [f(a) + 4f(\frac{a+b}{2}) + f(b)]$	$\vec{w} = (b-a)[\frac{1}{6}, \frac{4}{6}, \frac{1}{6}]^T$

• Taylor Expansion and Error

$f(x) = \sum_{m=0} \frac{f^{(m)}(\frac{a+b}{2})}{m!} (x - \frac{a+b}{2})^m$	$I(f) = \sum_{m=0} \frac{f^{(m)}(\frac{a+b}{2})}{(m+1)!} \frac{(2x-a-b)^{m+1}}{2^{m+1}} \Big _a^b$	
$T(f) = \frac{b-a}{2} \sum_{m=0} \frac{f^{(m)}(\frac{a+b}{2})}{m!} \frac{(b-a)^m}{2^m} [(-1)^m + 1]$	$= \sum_{m=0}^{\text{even}} \frac{f^{(m)}(\frac{a+b}{2})}{2^m(m+1)!} (b-a)^{m+1}$	
$= \sum_{m=0}^{\text{even}} \left[\frac{f^{(m)}(\frac{a+b}{2})}{2^m m!} \right] (b-a)^{m+1}$	$= M(f) + \sum_{m=2}^{\text{even}} \frac{E_m(f)}{m+1} h^{m+1}$	Q_1 error is $f^{(2)}$ derivative, not $f^{(1)}$!
$= M(f) + \sum_{m=2}^{\text{even}} \left[E_m(f) \right] h^{m+1}$	$= T(f) - \sum_{m=2}^{\text{even}} m \frac{E_m(f)}{m+1} h^{m+1}$	Q_2 error is $f^{(2)}$ & twice as large as Q_1
$S(f) = \left[\frac{2}{3}M(f) + \frac{1}{3}T(f) \right]$	$= S(f) - \sum_{m=4}^{\text{even}} \frac{m-2}{3} \frac{E_m(f)}{m+1} h^{m+1}$	Q_3 error is $f^{(4)}$ derivative, not $f^{(3)}$!

- n is even : Q_n error is expected $f^{(n)}$ derivative $Q(p_{n-1}) = I(p_{n-1}) \rightarrow \boxed{d = n - 1}$
- n is odd : Q_n error is $f^{(n+1)}$ derivative $Q(p_n) = I(p_n) \rightarrow \boxed{d = n}$
- 2 Rule Error: Est. diff. between $T(f)$ and $M(f)$ can be used to est. $I(f)$ error in using either.
- Can use subinterval, so can be progressive.
- Evenly-spaced x_i exhibit the Runge Phenom. $\rightarrow \boxed{Q_\infty(f) \text{ isn't always } I(f)}$
- Ill-conditioned and unstable : $(n \geq 11 \Rightarrow \exists w_i < 0), (\sum_i^\infty |w_i| \rightarrow \infty)$

Curtis-Clenshaw Quadrature [Rule]: n Chebyshev Nodes, $x_i \rightarrow n$ param. for w_i

- | | |
|--|---|
| <ul style="list-style-type: none"> • $\forall n : \forall w_i > 0 \Rightarrow \text{cond}(Q) = b - a$ • $\lim_{n \rightarrow \infty} C_n(f) = I(f)$ • $\boxed{d_n = n - 1}$ | <ul style="list-style-type: none"> • \exists an algorithm w/ Chebyshev polynomials to find integrand w/o solving for w_i. • Using Chebyshev polynomial zeroes is the classical CCQ. • Using Chebyshev extrema leads to a progressive rule [practical CCQ]. |
|--|---|

Gaussian Quadrature [Rule]: $\boxed{2n \text{ free param. for } x_i, w_i} \Rightarrow \boxed{d_n = 2n - 1}$

- $x_i, w_i : x_{n < i \leq 2n} = w_{n < i \leq 2n} = 0 \rightarrow \boxed{\begin{pmatrix} 1 & \dots & 1 & 0 & \dots \\ x_1 & \dots & x_n & 0 & \dots \\ x_1^2 & \dots & x_n^2 & 0 & \dots \\ \vdots & & \vdots & \vdots & \end{pmatrix} \begin{pmatrix} \vdots \\ w_n \\ 0 \\ \vdots \end{pmatrix} = \vec{z}(a, b)}$ usually $x_i \notin \mathbb{Q}$

- Ortho. Poly. : $\langle p_n(x) | x^k \rangle_{ab} = 0 \Rightarrow \begin{matrix} x_i : & p_n(x_i) = 0, & x_i \in \mathbb{R}, \\ & x_i \neq x_{j \neq i}, & x_i \in (a, b) \end{matrix}$ [−1,1] – Legendre
(−∞,∞) – Hermite
[0,∞) – Laguerre

- Interval Transform : $\boxed{\int_a^b f(t) dt = \frac{b-a}{\beta-\alpha} \int_\alpha^\beta f(t) dx \quad t = \frac{(b-a)x + a\beta - b\alpha}{\beta - \alpha}}$

- $\forall n : \forall w_i > 0 \Rightarrow \text{cond}(Q) = b - a$ • $\lim_{n \rightarrow \infty} G_n(f) = I(f)$

- $n = 2m + 1 \rightarrow \frac{a+b}{2} \in \{x_i\}_n$; otherwise usually $\{x_i\}_n \cup \{x_i\}_{\neq n} = 0 \rightarrow \boxed{\text{Not progressive}}$

- Progressive Gauss-Kronrod, K_{2n+1} : $n \text{ from } G_n \rightarrow \begin{matrix} n+1 \text{ param for } x_{i>n} \\ 2n+1 \text{ param for } w_i \end{matrix} \Rightarrow \boxed{d_{2n+1} = 3n + 1 < 4n + 1}$
GK 2-Rule Error : $\boxed{\Delta I(f) \approx (200|G_n - K_{2n+1}|)^{1.5}}$

Progressive Gauss-Patterson, P_{4n+3} : $2n + 1 \text{ from } K_{2n+1} \rightarrow \begin{matrix} 2n+2 \text{ param for } x_{i>n} \\ 4n+3 \text{ param for } w_i \end{matrix} \Rightarrow \boxed{d_{4n+3} = 6n + 4 < 8n + 5}$

- Closed Gauss-Randau : $x_i \in [a, b) \text{ or } (a, b] \rightarrow \boxed{d = 2n - 2}$
Closed Gauss-Lobatto : $x_i \in [a, b] \rightarrow \boxed{d = 2n - 3}$

Composite [k -Subintervals] Quadrature for Rule Q_n : $Q_n \rightarrow Q_{kn} \text{ or } Q_{kn-(k-1)},$

- $\lim_{k \rightarrow \infty} C_{k,n} = \sum_{j=1}^{k \rightarrow \infty} \left[\sum_{i=1}^n w_i f(x_{ji}) \right] = \sum_{i=1}^n \frac{w_i}{h_k} \left[\sum_{j=1}^{k \rightarrow \infty} h_k f(x_{ji}) \right] = I(f) \sum_{i=1}^n \frac{w_i}{h_k} = I(f)$ $h_k = (b-a)/k$
 $\geq (x_{jn} - x_{j1})$
 $\boxed{d \geq 0} \Rightarrow \sum w_i = h_k$
- Error : $\mathcal{O}(h^{m+1}) \rightarrow \mathcal{O}(kh_k^{m+1}) = \boxed{\mathcal{O}(h_k^m)} \quad (k > 1)$

Adaptive Quadrature for Rule Q_n : Divide subinterval until a tolerance is met.

4.3 n -D Integration

Double Integral: Use a pair of 1-D routines for the inner/outer integral.

($n > 2$)-Dimension Integral: Monte Carlo is best (error $1/\sqrt{n} \rightarrow 0$).

4.4 Other Integrals

Tabular Data: Integrate a piecewise interpolant.

Improper Integral: Separate the integral, do a variable change,
or add/subtract a term to remove singularities.

(Fredholm) Integral Equations: skipped

4.5 Richardson Extrapolation [for Integration]

$$\begin{aligned} F(h) &= I(f) + a_1 h^p + \mathcal{O}(h^{q>p}) \\ F(\frac{h}{k}) &= I(f) + a_1 (\frac{h}{k})^p + \mathcal{O}(h^{r\geq q}) \end{aligned} \Rightarrow \boxed{I(f) = \frac{k^p F(\frac{h}{k}) - F(h)}{k^p - 1} + \mathcal{O}(h^{q>p})}$$

- Romberg Integration [Quadratic Extrapolation for Comp. Trapezoidal Rule] :

$$\begin{aligned} T(f, \frac{h}{2^k}) &= I(f) + 2^k \left[a_1 (\frac{h}{2^k})^3 + \mathcal{O}(\frac{h}{2^k}^5) \right] \\ T_{k,j=0} &= I(f) + h a_1 [\frac{h}{2^k}]^2 + h \mathcal{O}([\frac{h}{2^k}]^4) \end{aligned} \Rightarrow \begin{aligned} T_{k+1,j+1} &\equiv \frac{4^{j+1} T_{k+1,j} - T_{k,j}}{4^{j+1} - 1} \quad (1 \leq j \leq k) \\ \boxed{I(f) &= T_{k,j} + \mathcal{O}(h^{2j+2})} \end{aligned}$$

$$4T_{k+1,0} = 4I(f) + h a_1 [\frac{h}{2^k}]^2 + \frac{h}{4} \mathcal{O}([\frac{h}{2^k}]^4)$$

5 Numerical Differentiation

Conditioning: Inverse of Integration - which smoothes noisy data - so derivatives are inherently sensitive to small changes.

5.1 Finite-Difference Approx

$$\begin{aligned} f'(x) &= \frac{f(x+h)-f(x)}{h} - \sum_{n=2}^{\infty} \frac{f^{(n)}(x)}{n!} h^{n-1} \\ &= \frac{f(x)-f(x-h)}{h} - \sum_{n=2}^{\infty} \frac{f^{(n)}(x)}{n!} (-h)^{n-1} \\ &= \frac{f(x+h)-f(x-h)}{2h} - \sum_{n=3}^{\infty} \frac{f^{(n)}(x)}{n!} h^{n-1} \end{aligned}$$

- Use more points n for higher order approx.

5.2 Deriving Interpolant

$$\begin{aligned} f(x) &\approx \hat{f}_n(x) = p_{n-1}(x) \in P_{n-1} \\ f^{(m)}(x) &\approx \hat{f}_n^{(m)}(x) \end{aligned}$$

- Equivalent but easier than finite-diff. approach.
- Using more points n leads to better accuracy.
- Polynomials, or other interpolants like trig. func. can be used.

5.3 Richardson Extrapolation [for Differentiation]

$$\begin{aligned} F(h) &= D(f) + a_1 h^p + \mathcal{O}(h^{q>p}) \\ F(\frac{h}{k}) &= D(f) + a_1 (\frac{h}{k})^p + \mathcal{O}(h^{r\geq q}) \end{aligned} \Rightarrow \boxed{D(f) = \frac{k^p F(\frac{h}{k}) - F(h)}{k^p - 1} + \mathcal{O}(h^{q>p})}$$

- E.g. $D(f) = \frac{f(x+h)-f(x)}{h} + \mathcal{O}(h)$

$$\begin{aligned} F(h) &= \frac{f(x+h) - f(x)}{h} \\ F(\frac{h}{2}) &= \frac{f(x+\frac{h}{2}) - f(x)}{h/2} \end{aligned} \Rightarrow \boxed{D(f) = \frac{2 \cdot \frac{f(x+h/2)-f(x)}{h/2} - \frac{f(x+h)-f(x)}{h}}{2-1} + \mathcal{O}(h^2)}$$

5.4 Method of Undetermined Coefficients (MUC) / System of Moment Equations

$$\boxed{(D_n(f))(a) \equiv \frac{df}{dx}(a) = \frac{d\hat{f}}{dx}(a) = \sum_{i=1}^n f(x_i) \phi'_i(a) = \sum_{i=1}^n f(x_i) w_i} \quad \bullet \quad \boxed{x_i, w_i \rightarrow 2n \text{ max param.}}$$

$$\begin{aligned} \left(\frac{d}{dx} \sum_{j=1}^n c_j x^{j-1} \right) (a) &= \sum_{i=1}^n \left(\sum_{j=1}^n c_j x_i^{j-1} \right) w_i \\ \sum_{j=1}^n c_j \frac{d(x^{j-1})}{dx}(a) &= \sum_{j=1}^n c_j \left(\sum_{i=1}^n x_i^{j-1} w_i \right) \end{aligned} \Rightarrow \begin{aligned} &\boxed{\sum_{i=1}^n f(x_i) w_i = \frac{d\hat{f}}{dx}(a)} \\ &\downarrow \\ &\boxed{\sum_{i=1}^n x_i^{j-1} w_i = \frac{d(x^{j-1})}{dx}(a)} \\ &\equiv z_j \end{aligned}$$

(maybe some dot product to isolate terms)

$\bullet \quad \boxed{z_1 = \sum w_i = 0}$

(Vandermode Matrix)

$$\begin{bmatrix} 1 & 1 & 1 & \dots \\ x_1 & x_2 & x_3 & \dots \\ x_1^2 & x_2^2 & x_3^2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \vec{w} = \vec{z} = \begin{bmatrix} 0 \\ 1 \\ 2a \\ \vdots \end{bmatrix}$$

6 Initial Value Problems for ODEs: $\vec{y}'(t) = \vec{f}(t, \vec{y}); \vec{y}(t_0)$

k -th Order : $f(t, y, \dots, y^{(k)}) = 0$

Linear : $\vec{a}(t) \cdot [y(t), \dots, y^{(k)}(t)]^T = b(t)$

Autonomous : $f(y, \dots, y^{(k)}) = 0$

Homogeneous : $b(t) = 0$

$$\begin{array}{l} \text{\underline{k-th Order}} \\ \text{\underline{System of } } n \\ \text{\underline{Coupled ODEs}} \end{array} : \vec{y}'_i(t) = \begin{bmatrix} y'_1(t) \\ u'_2(t) \\ \vdots \\ u'_{k-1}(t) \\ u'_k(t) \end{bmatrix} = \begin{bmatrix} u_2(t) \\ u_3(t) \\ \vdots \\ u_k(t) \\ f(t, u_1, \dots, u_k) \end{bmatrix} \Rightarrow \boxed{\vec{y}'(t) = \begin{bmatrix} \vec{y}'_1(t) \\ \vdots \\ \vec{y}'_n(t) \end{bmatrix} = \vec{f}(t, \vec{y}) = \begin{bmatrix} \vec{f}_1(t, \vec{y}) \\ \vdots \\ \vec{f}_n(t, \vec{y}) \end{bmatrix}}$$

$$\text{\underline{Nonautonomous} } \rightarrow \text{\underline{Autonomous}}: \boxed{\vec{y}' = f(t, \vec{y}) \rightarrow \begin{bmatrix} \vec{y}' = f(y_{n+1}, \vec{y}) \\ y'_{n+1} = 1 \end{bmatrix}}$$

6.1 ODE Stability (Conditioning)

Unique/Exists if Lipschitz Continuous:

$$\|\vec{f}(t, \hat{y}) - \vec{f}(t, \vec{y})\|_\infty \leq L \|\hat{y} - \vec{y}\|_\infty^{\alpha=1} = \left(\max \|J_f(t, \vec{y})\| \right) \cdot \|\hat{y} - \vec{y}\| \quad \text{Assured if } \vec{f} \text{ is differentiable}$$

$$\underline{\hat{y}' = \hat{f}(t, \hat{y})} : \|\hat{y} - \vec{y}\|_\infty \leq e^{L(t-t_0)} \|\hat{y}(t_0) - \vec{y}(t_0)\|_\infty + \frac{e^{L(t-t_0)} - 1}{L} \left(\max \|\hat{f} - \vec{f}\| \right) \quad \left(\begin{array}{l} \text{cont./well-posed but} \\ \text{possibly sensitive} \end{array} \right)$$

$$\text{\underline{Stable } } (\epsilon\text{-}\delta): \forall \epsilon > 0, \exists \delta > 0 : \|\hat{y}(t_0) - \vec{y}(t_0)\| < \delta \Rightarrow \|\hat{y}(t) - \vec{y}(t)\| < \epsilon \quad (\text{exp. above ruled out})$$

$$\text{\underline{Asymptotically Stable}}: \lim_{t \rightarrow \infty} \|\hat{y}(t) - \vec{y}(t)\| = 0$$

$$y' = \lambda y \Rightarrow \lim_{t \rightarrow \infty} y = y_0 e^{\lambda t} \quad \left\{ \begin{array}{ll} \text{Re}(\lambda) < 0 & \rightarrow \text{Asymp. Stable [Cond.]} \\ \text{Re}(\lambda) = 0 & \rightarrow \text{Oscillating (Stable Cond.)} \\ \text{Re}(\lambda) > 0 & \rightarrow \text{Unstable [Cond.]} \end{array} \right.$$

• A is diagonalizable / J_f is diagonalizable

$$\Rightarrow \vec{y}_0 = \sum a_i \vec{v}_i, \quad \vec{y}(t) = \sum a_i \vec{v}_i e^{\lambda_i t}$$

$$\rightarrow \left\{ \begin{array}{ll} \forall \lambda_i \text{ Re}(\lambda_i) < 0 & \rightarrow \text{Asymp. Stable [Cond.]} \\ \forall \lambda_i \text{ Re}(\lambda_i) = 0 & \rightarrow \text{Stable [Cond.]} \\ \exists \lambda_i \text{ Re}(\lambda_i) > 0 & \rightarrow \text{Unstable [Cond.]} \end{array} \right.$$

• A isn't diagonalizable / J_f isn't diagonalizable

$$\rightarrow \text{Stable [Cond.]} \text{ if } \forall \lambda_i \quad \left\{ \begin{array}{l} \bullet \text{Re}(\lambda_i) \leq 0 \\ \bullet \text{Re}(\lambda_i) < 0 \text{ if } \lambda_i \text{ isn't simple.} \end{array} \right.$$

• $A = A(t)$ / \vec{f} isn't autonomous $\rightarrow J_f = J_f(t, \vec{y})$

\rightarrow Might not be long-term stable

Linear System of ODEs

$$\boxed{\vec{y}' = A \vec{y}}$$

$$\vec{y}(0) = \vec{y}_0$$

\rightarrow

Nonlinear System of ODEs

$$\boxed{\vec{y}' = \vec{f}(t, \vec{y})}$$

$$\rightarrow \vec{y}' \approx J_f(t, \vec{y}) \vec{y}$$

6.2 Algorithm Stability and Error

Local [Trunc.] Error (Accuracy): $\vec{l}_k = \vec{y}_k(t_k) - \vec{y}_{k-1}(t_k) = \mathcal{O}(h_k^{p+1}) \rightarrow \frac{\vec{l}_k}{h_k} = \mathcal{O}(h_k^p)$ (Order p)

Global [Trunc.] Error (Stability): $\vec{e}_k = \vec{y}_k - \vec{y}(t_k) = \mathcal{O}(\hat{h}_k^p)$ (under “reasonable” conditions)

Growth/Amplification Factor, g • $y' = \lambda y \Rightarrow y_k = g^k y_0 \begin{cases} |g| \leq 1 \rightarrow \text{Stable} \\ |g| > 1 \rightarrow \text{Unstable} \end{cases}$

• $\vec{e}_{k+1} = g\vec{e}_k + \vec{l}_{k+1} \begin{cases} \rho(g) \leq 1 \rightarrow \text{Stable} \\ \rho(g) > 1 \rightarrow \text{Unstable} \end{cases} \quad \left(\begin{array}{c} \text{Spectral Radius, } \rho(\mathbb{R}^{n \times n}), \\ \text{may vary with } t \end{array} \right)$

Unconditionally Stable: If stable alg. when $(\forall h, h > 0), (\forall \lambda_i, \text{Re}(\lambda_i) < 0 \Rightarrow \text{Stable [Cond.]})$

Implicit Method: $y_{k+1} = y_{k+1}(t_k, t_{k+1}, \dots)$ (usually more stable than explicit methods, $y_{k+1} = y_{k+1}(t_k)$)

6.3 ODE Stiffness

[Asymptotic] Stiffness:

- Rapid asymp. decay to convergence; $\text{Re}(\lambda_i(J_f)) \ll \sim 0$ and differ greatly in magnitude.
- Normally small h_k required; even w/ an alg. with no local error, a perturbation of an initial value may cause a step to overshoot to neighboring solutions/level sets.
- Implicit methods with greater range of stability allow larger h_k for stiff ODEs than explicit ones.

[Oscillatory] Stiffness: Rapid oscillation stiffness; $|\text{Im}(\lambda_i(J_f))| \gg \sim 0$ and differ greatly in magnitude. Treatment not given.

6.4 Taylor Series Algorithms, $\vec{y}(t+h) = \vec{y}(t) + \sum_i \frac{h^i}{i!} \vec{y}^{(i)}(t) + \mathcal{O}(h^{p+1})$

Local Error Tolerance : $\frac{h^{p+1}}{(p+1)!} \|\vec{y}^{(p+1)}(t)\| \leq \text{tol} \Rightarrow h_k \lesssim \sqrt[p+1]{(p+1)! \cdot \text{tol} / \|\vec{y}_k^{(p+1)}\|}$ (Could use finite diff for $\|\vec{y}_k''\|$)

[Explicit] Forward Euler's Method (1st Order): $\vec{y}_{k+1} = \vec{y}_k + h_k \vec{y}'_k = \vec{y}_k + h_k \vec{f}(t_k, \vec{y}_k)$

- $g : y_k = (1 + h_k \lambda)^k y_0 \Rightarrow e^{\lambda h} = g + \mathcal{O}(h^2)$ $p=1$, $|1 + h\lambda| \leq 1 \rightarrow$ Stable if $\lambda : |1/h + \lambda| \leq 1/h$
- $g : \vec{e}_{k+1} = \vec{y}_{k+1} - \vec{y}(t_k + h_k) = [\vec{y}_k + h_k \vec{y}'_k] - [\vec{y}(t_k) + h_k \vec{y}'(t_k) + \mathcal{O}(h_k^2)]$
 $= [\vec{y}_k - \vec{y}(t_k)] + h_k [\vec{f}(t_k, \vec{y}_k) - \vec{f}(t_k, \vec{y}(t_k))] - \mathcal{O}(h_k^2)$ $p=1$
 $= \vec{e}_k + h_k \bar{J}_f \vec{e}_k - \mathcal{O}(h_k^2)$, (From Mean Value Theorem : $\bar{J}_f \vec{e}_k = \int_0^1 J_f(t_k, \alpha \vec{y}_k + (1-\alpha) \vec{y}(t_k)) d\alpha \cdot \vec{e}_k$)
 $= [I + h_k \bar{J}_f] \vec{e}_k + \vec{l}_{k+1} \Rightarrow$ $\rho(I + h_k \bar{J}_f) \leq 1 \rightarrow \text{Stable}$
- Stiffness Tolerance : $h_k \cdot \min(\text{Re}(\lambda_i(J_f))) \ll -1$ (small tolerance, not uncond. stable)

[Imp.] Backwards Euler's Method: $\vec{y}_{k+1} = \vec{y}_k + h_k \vec{f}(t_{k+1}, \vec{y}_{k+1})$ (Solve Nonlin. Eq.; use init. guess from an explicit method)

- $g : y_k = (\frac{1}{1-h_k \lambda})^k y_0 \Rightarrow e^{\lambda h} = 1/(1-h\lambda) + \mathcal{O}(h^2)$ $p=1$, $|1 - h\lambda| \geq 1 \rightarrow \text{Unconditionally Stable}$
- $g : \rho((I - hJ_f)^{-1}) \leq 1$ (sic hJ_f) \rightarrow Unconditionally Stable \rightarrow Greater Stiffness Tol.

[Exp.] $p = 2$: $\vec{y}_{k+1} = \vec{y}_k + h_k \vec{y}'_k + \frac{h_k^2}{2} \vec{y}''_k = \vec{y}_k + h_k \vec{f}(t_k, \vec{y}_k) + \frac{h_k^2}{2} \left(\vec{f}_t(t_k, \vec{y}_k) + \vec{f}_y(t_k, \vec{y}_k) \vec{f}(t_k, \vec{y}_k) \right)$

6.5 [Exp.] Runge-Kutta Algorithms

$$\vec{y}_{k+1} = \vec{y}_k + h_k \sum_{i=1}^s b_i k_i \quad , \quad k_i = \vec{f}(t_k + c_i h_k, \vec{y}_k + h_k \sum_{j=1}^{i-1} a_{ij} k_j) \quad , \quad \sum_i b_i = 1$$

Heun's Method/
Trapezoid Rule :
($p = 2, s = 2$)

$$\vec{y}_{k+1} \approx \vec{y}_k + \frac{h_k}{2} [\vec{f}(t_k, \vec{y}_k) + \vec{f}(t_{k+1}, \vec{y}_{k+1})] \quad (\text{See Multistep Trap. Rule Below})$$

$$= \vec{y}_k + \frac{h_k}{2} [\vec{f}(t_k, \vec{y}_k) + \vec{f}(t_k + h_k, \vec{y}_k + h_k \vec{f}(t_k, \vec{y}_k))]$$

$$= \vec{y}_k + \frac{h_k}{2} [k_1 + k_2]$$

$$\vec{c} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} = a$$

$$, [\frac{1}{2}, \frac{1}{2}]^T = \vec{b}$$

RK4/
Simpson's Rule :
($p = 4, s = 4$)

$$\vec{y}_{k+1} = \vec{y}_k + \frac{h_k}{6} [k_1 + 2k_2 + 2k_3 + k_4]$$

$$\vec{c} = \begin{bmatrix} 0 \\ 1/2 \\ 1/2 \\ 1 \end{bmatrix}, \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = a$$

$$, [\frac{1}{6} \quad \frac{2}{6} \quad \frac{2}{6} \quad \frac{1}{6}]^T = \vec{b}$$

- Explicit Runge-Kutta have no error estimates to base step-size on.
- Embedded [paired] RK methods have pair-diff. error estimates.
- Implicit RK methods exist for stiff ODEs: $k_i = \vec{f}(t_k + c_i h_k, \vec{y}_k + h_k \sum_{j=1}^s a_{ij} k_j)$

6.6 [Linear] Multistep Algorithms (May Use Previous Points/Not Self-Starting)

Interpolate
(use MUC for coefficients)
($\forall t_k$, Let $h=1$)

$$\vec{y}_{k+1} = \sum_{i=1}^m \alpha_i y_{k-(i-1)} + h \sum_{i=0}^m \beta_i \vec{y}'_{k+1-i}$$

or

$$\sum_{i=0}^m a_i y_{k+1-i} = h \sum_{i=0}^m b_i y'_{k+1-i}$$

(Explicit if $\beta_0 = 0$)

Adams Methods
Explicit = Adams-Bashforth (AB) :
Implicit = Adams-Moulton (AM)

$$\vec{y}_{k+1} = \vec{y}_k + h \sum_{i=0}^m \beta_i \vec{y}'_{k+1-i}$$

$$\Leftrightarrow \vec{y}_{k+1} - \vec{y}_k = \sum \vec{y}'_{k+1-i} \int_{t_k}^{t_k+h} \phi_{k+1-i}(t) dt$$

$$\frac{t_{k+1}^j}{j} - \frac{t_k^j}{j} = \sum t_{k+1-i}^{j-1} w_{k+1-i}$$

2-Step AB : $t_{k-1}=0, h=1$, $\begin{bmatrix} y_k & y'_k & y'_{k-1} \\ 1 & 0 & 0 \\ t_k & 1 & 1 \\ t_k^2 & 2t_k & 2t_{k-1} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ h\beta_1 \\ h\beta_2 \end{bmatrix} = \begin{bmatrix} y_{k+1} \\ t_{k+1} \\ t_{k+1}^2 \end{bmatrix}$ or $\begin{bmatrix} t_k^{j-1} & t_{k-1}^{j-1} \\ 1 & 1 \\ t_k & t_{k-1} \end{bmatrix} \begin{bmatrix} h\beta_1 \\ h\beta_2 \end{bmatrix} = \begin{bmatrix} (t_{k+1}^j - t_k^j)/j \\ t_{k+1} - t_k \\ (t_{k+1}^2 - t_k^2)/2 \end{bmatrix}$

$$\Rightarrow \vec{y}_{k+1} = \vec{y}_k + h \left(\frac{3}{2} \vec{y}'_k - \frac{1}{2} \vec{y}'_{k-1} \right)$$

2-Step AM/
Trapezoid Rule : $t_k=0, h=1$ $\begin{bmatrix} t_{k+1} & t_k \\ 1 & 1 \\ t_{k+1} & t_k \end{bmatrix} \begin{bmatrix} h\beta_0 \\ h\beta_1 \end{bmatrix} = \begin{bmatrix} (t_{k+1}^j - t_k^j)/j \\ t_{k+1} - t_k \\ (t_{k+1}^2 - t_k^2)/2 \end{bmatrix}$ or $\vec{y}_k + h \vec{y}'_k + \frac{h^2}{2} \left[\frac{\vec{y}'_{k+1} - \vec{y}'_k}{h} \right]$

$$\text{or } \vec{y}_k + \frac{(t_k+h)-t_k}{2} [\vec{y}'_k + \vec{y}'_{k+1}] \Rightarrow \vec{y}_{k+1} = \vec{y}_k + \frac{h}{2} [\vec{y}'_k + \vec{y}'_{k+1}]$$

- $g : y_k = \left(\frac{1+h\lambda/2}{1-h\lambda/2} \right)^k y_0 \Rightarrow e^{\lambda h} = g + \mathcal{O}(h^3)$ $\boxed{p=2}$, $\boxed{|g| < 1 \rightarrow \text{Unconditionally Stable}}$

- $g : \rho((I+hJ_f)(I-hJ_f)^{-1}) < 1$ (sic hJ_f) \rightarrow $\boxed{\text{Unconditionally Stable} \rightarrow \text{Greater Stiffness Tol.}}$

- Local Error Tolerance : $\boxed{\frac{2}{2+1} \frac{1}{2^2 2!} \|\vec{f}''(t_{k+h/2}, \vec{y}(t_{k+h/2}))\| h^3 \lesssim tol}$

Backwards Differentiation
Formula (BDF) Methods :

$$\vec{y}_{k+1} = \sum_{i=1}^m \alpha_i \vec{y}_{k-(i-1)} + h\beta_0 \vec{y}'_{k+1}, \quad \vec{y}'_{k+1} = \sum \vec{y}_{k+1-i} \phi'_{k+1-i}(t_{k+1}), \quad \frac{d(t^{j-1})}{dt}(t_{k+1}) = \sum t_{k+1-i}^{j-1} w_{k+1-i}$$

2-Step BDF : $t_{k-1}=0, h=1$, $\begin{bmatrix} y_k & y_{k-1} & y'_{k+1} \\ 1 & 1 & 0 \\ t_k & t_{k-1} & 1 \\ t_k^2 & t_{k-1}^2 & 2t_{k+1} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ h\beta_0 \end{bmatrix} = \begin{bmatrix} y_{k+1} \\ t_{k+1} \\ t_{k+1}^2 \end{bmatrix}$ or $\begin{bmatrix} t_{k+1}^{j-1} & t_k^{j-1} & t_{k-1}^{j-1} \\ 1 & 1 & 1 \\ t_{k+1} & t_k & t_{k-1} \\ t_{k+1}^2 & t_k^2 & t_{k-1}^2 \end{bmatrix} \begin{bmatrix} 1/h\beta_0 \\ -\alpha_1/h\beta_0 \\ -\alpha_2/h\beta_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2t_{k+1} \end{bmatrix} \quad (j-1)(t_{k+1}^{j-2})$

$$\Rightarrow \vec{y}_{k+1} = \frac{4}{3} \vec{y}_k - \frac{1}{3} \vec{y}_{k-1} + h \left(\frac{2}{3} \vec{y}'_{k+1} \right)$$

Kinda-Generalized Adams (Interpolating $y'(t)$ then Integrating)

$$\sum_{i=0}^m \left[\hat{\mathcal{I}} \phi_{k+1-i}(t) \right] y'_{k+1-i} = \hat{\mathcal{I}} y'(t) = \sum_{i=0}^m a_{k+1-i} y_{k+1-i}$$

$$\sum_{i=0}^m [b_{k+1-i}] \frac{d(t^{j-1})}{dt} \Big|_{k+1-i} = \hat{\mathcal{I}} \frac{d(t^{j-1})}{dt} = \sum_{i=0}^m a_{k+1-i} t_{k+1-i}^{j-1}$$

- $\hat{\mathcal{I}} f(t) = \int_{t_k}^{t_{k+1}} f(t) dt$ (Adams)
- $\hat{\mathcal{I}} f(t) = (\vec{a} \cdot [1, e^{-h\nabla}, \dots, e^{-mh\nabla}] F)(t_{k+1})$

Kinda-Generalized BDF (Interpolating $y(t)$ then Deriving)

$$\sum_{i=0}^m \left[\hat{\mathcal{D}} \phi_{k+1-i}(t) \right] y_{k+1-i} = \hat{\mathcal{D}} y(t) = \sum_{i=0}^m b_{k+1-i} y'_{k+1-i}$$

$$\sum_{i=0}^m [a_{k+1-i}] t_{k+1-i}^{j-1} = \hat{\mathcal{D}} t^{j-1} = \sum_{i=0}^m b_{k+1-i} \frac{d(t^{j-1})}{dt} \Big|_{k+1-i}$$

- $\hat{\mathcal{D}} f(t) = \frac{df}{dt}(t_{k+1})$ (BDF)
- $\hat{\mathcal{D}} f(t) = (\vec{b} \cdot [1, e^{-h\nabla}, \dots, e^{-mh\nabla}] f')(t_{k+1})$

PECE - Predict[or], Evaluate
Correct[or], Evaluate :

A set of previous point values is used in an explicit multistep algorithm as a *predictor* to find the next value, y_{k+1} . The derivative is then *evaluated* at this next time as $y'_{k+1} = f(t_{k+1}, y_{k+1})$. With this derivative, an improved value for y_{k+1} is found with an implicit multistep algorithm as a *corrector*. The derivative y'_{k+1} can then be improved by *evaluating* it again with the improved y_{k+1} from the corrector. The implicit corrector can be repeated to re-evaluate y_{k+1} and y'_{k+1} until convergence. PECE is explicit.

- Mult. methods must be used to get previous values.
- Changing step-size h is hard since interpolation is most convenient for equal-spaced points.
- Relatively hard to code.
- Not all imp. methods are unconditionally stable.
- Method pairs can be used for error estimates.

6.7 Multivalue Methods

$$y' = f(t, y) \quad , \quad \vec{y}_k = \left[y_k, h y'_k, \frac{h^2}{2} y''_k, \frac{h^3}{3!} y'''_k \right]^T, \quad \vec{y}_{k+1} = \left[y_{k+1}, h y'_{k+1}, \frac{h^2}{2} y''_{k+1}, \frac{h^3}{3!} y'''_{k+1} \right]^T$$

$$\vec{y}_{k+1} \equiv B \vec{y}_k = \begin{matrix} \text{(Pascal's)} \\ \text{Triangle} \end{matrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_k \\ h y'_k \\ (h^2/2!) y''_k \\ (h^3/3!) y'''_k \end{bmatrix} \Rightarrow \vec{y}_{k+1} = \vec{\hat{y}}_{k+1} + \alpha \vec{r}$$

$$\begin{bmatrix} \hat{y}_{k+1} \\ h \hat{y}'_{k+1} \\ (h^2/2!) \hat{y}''_{k+1} \\ (h^3/3!) \hat{y}'''_{k+1} \end{bmatrix} = \begin{bmatrix} [y_k + h y'_k + (h^2/2!) y''_k + (h^3/3!) y'''_k] \\ h[y'_k + h y''_k + (h^2/2!) y'''_k] \\ (h^2/2!)[y''_k + h y'''_k] \\ (h^3/3!)[y'''_k] \end{bmatrix} \Rightarrow \begin{bmatrix} y_{k+1} \\ h y'_{k+1} \\ \frac{h^2}{2!} y''_{k+1} \\ \frac{h^3}{3!} y'''_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{y}_{k+1} \\ h \hat{y}'_{k+1} \\ \frac{h^2}{2!} \hat{y}''_{k+1} \\ \frac{h^3}{3!} \hat{y}'''_{k+1} \end{bmatrix} + h(y'_{k+1} - \hat{y}'_{k+1}) \begin{bmatrix} r_1 = \frac{3}{8} \\ 1 \\ r_3 = \frac{3}{4} \\ r_4 = \frac{1}{6} \end{bmatrix}$$

- Equivalent to Multistep Methods.
- Easy to change step size h at \vec{y}_i .
- Easy to change order p from \vec{r} .