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

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

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

1.1 One Dimension/Equation **skipped a lot**

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

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

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

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

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

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

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

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

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

- $\boxed{r = r_+ \approx 1.618}$: $r_+^2 - r_+ - 1 = 0$ (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 $\vec{y} = 0$):

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

- $\boxed{\vec{g}(\vec{x}) \equiv \vec{x} - J_f(\vec{x})^{-1}\vec{f}(\vec{x})} \Rightarrow$ $J_g(\bar{x}) = \cancel{I - J_f(\bar{x})^{-1} J_f(\bar{x})} + \sum_{i=1}^{n} H_i(\bar{x}) f_i(\bar{x})$ (if $J_f(\bar{x})$ is nonsingular) H_i = component matrix of the tensor, $D_x J_f(\bar{x})$

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

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

$$\boxed{B_k\vec{h}_k = -\vec{f}(x_k)} \quad \Rightarrow \quad \boxed{\vec{x}_{k+1} = \vec{x}_k + \vec{h}_k} , \quad \boxed{B_{k+1} = B_k + \frac{f(x_{k+1})h_k^T}{h_k^T h_k}}$$

(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 $\min f(\vec{x}) = f(\bar{x})$]

2.1 Function Shape and Convexity

Coercive: $\boxed{\lim_{x \to \pm\infty} f(x) = \infty}$

Unimodal: $\begin{matrix} a \le \bar{x} \le b \\ x_1 < x_2 \end{matrix} : \boxed{\begin{matrix} x_2 < \bar{x} & \to & f(x_1) > f(x_2) \\ \bar{x} < x_1 & \to & f(x_1) < f(x_2) \end{matrix}}$

$\exists$ 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:
$$\boxed{\begin{aligned} f_{(\bar{x}+\vec{s})} - f_{(\bar{x})} &= \overset{\alpha_1 \in (0,1)}{\vec{\nabla} f_{(\bar{x}+\alpha_1\vec{s})}} \cdot \vec{s} = \vec{\nabla} f_{(\bar{x})} \cdot \vec{s} + \tfrac{1}{2}\langle \vec{s} | \overset{\alpha_2 \in (0,1)}{H_{f(\bar{x}+\alpha_2\vec{s})}} | \vec{s} \rangle \geq 0 \\ f_{(\vec{x}+s\hat{u})} - f_{(\vec{x})} &= \vec{\nabla} f_{(\vec{x}+\alpha_1 s\hat{u})} \cdot s\hat{u} = \vec{\nabla} f_{(\vec{x})} \cdot \vec{s} + \tfrac{s^2}{2}\langle \hat{u} | H_{f(\vec{x}+\alpha_2\vec{s})} | \hat{u} \rangle \end{aligned}}$$

- $\lim\limits_{s\to 0}\left(\frac{f(\vec{x}+\vec{s})-f(\vec{x})}{s} = \vec{\nabla} f_{(\vec{x}+\alpha_1 s\hat{u})} \cdot \cancel{s}\hat{u}\right) \Rightarrow \left(\vec{\nabla} f(\bar{x})\cdot\hat{u} \geq 0 \to \boxed{\vec{\nabla} f(\bar{x})\cdot\vec{s} \geq 0}\right)$, $\boxed{\begin{array}{l}\text{Cauchy-Schwarz} \to \\ \max \vec{\nabla} f_{(\vec{x})} \cdot \hat{u} \text{ if } \vec{u} = \vec{\nabla} f(\vec{x})\end{array}}$
- $\boxed{\vec{u} = \mp\vec{\nabla} f_{(\vec{x})}} \Rightarrow \lim\limits_{s\to 0}\left(\frac{f(\vec{x}+\vec{s})-f(\vec{x})}{s} = \mp\cancel{s}\frac{\vec{\nabla} f(\vec{x}+\alpha_1 s\hat{u})\cdot\vec{\nabla} f(\vec{x})}{\|\vec{\nabla} f(\vec{x})\|}\right) = \mp\|\vec{\nabla} f_{(\vec{x})}\| \lesseqgtr 0$ $\boxed{\begin{array}{l}\text{if } \pm\vec{\nabla} f(\vec{x}) \neq 0\text{, its dir.} \\ \text{is an ascent/descent.}\end{array}}$
- $\lim\limits_{s\to 0}\left(\frac{f(\vec{x}+\vec{s})-f(\vec{x})+f(\vec{x}-\vec{s})-f(\vec{x})}{s^2} = \frac{\langle\hat{u}|H_f(\vec{x}+\alpha_2\vec{s})+H_f(\vec{x}-\alpha_3\vec{s})|\hat{u}\rangle}{2}\right) = \langle\hat{u}|H_f(\vec{x})|\hat{u}\rangle \Rightarrow \boxed{\langle\vec{s}|H_f(\bar{x})|\vec{s}\rangle \geq 0}$

2.2.1 Unconstrained Optimization Conditions

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

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

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

2.2.2 Constrained Optimization Conditions

- $\boxed{\begin{array}{c}\vec{s} = \text{feasable direction} \\ f(\bar{x}) = \min f \text{ given } g,\ h\end{array}} \Leftrightarrow \left(\boxed{\vec{\nabla} f(\bar{x})\cdot\vec{s} \geq 0} , \boxed{\langle\vec{s}|H_f(\bar{x})|\vec{s}\rangle \geq 0}\right)$

Optimization $\begin{array}{l} f: \mathbb{R}^n \to \mathbb{R} \\ g: \mathbb{R}^n \to \mathbb{R}^m \\ h: \mathbb{R}^n \to \mathbb{R}^p \end{array}$ $\boxed{\min f(\vec{x}) = y \quad \text{w/} \quad \begin{pmatrix}\vec{g}(\vec{x}) = 0 \\ \vec{h}(\vec{x}) \leq 0\end{pmatrix}}$

active : $h_i(\bar{x}) = 0$

inactive : $h_i(\bar{x}) < 0 \to \overset{\text{(see KKT)}}{\bar{\mu}_i = 0}$

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

$$H_{\mathcal{L}(\bar{x},\bar{\lambda},\bar{\mu})} = \begin{pmatrix}\nabla_{xx}\mathcal{L} & \nabla_{x\lambda}\mathcal{L} & \nabla_{x\mu}\mathcal{L} \\ \nabla_{\lambda x}\mathcal{L} & \nabla_{\lambda\lambda}\mathcal{L} & \nabla_{\lambda\mu}\mathcal{L} \\ \nabla_{\mu x}\mathcal{L} & \nabla_{\mu\lambda}\mathcal{L} & \nabla_{\mu\mu}\mathcal{L}\end{pmatrix} = \underset{\text{(can't be pos. def.)}}{\begin{pmatrix}\nabla_{xx}\mathcal{L} & J_g^T & J_h^T \\ J_g & 0 & 0 \\ J_h & 0 & 0\end{pmatrix}} , \quad \boxed{\nabla_{xx}\mathcal{L}_{(\bar{x},\bar{\lambda},\bar{\mu})} = H_f + \sum_i^m \bar{\lambda}_i H_{g_i} + \sum_i^{\text{act}\leq p} \bar{\mu}_i H_{h_i}}$$

- Assume $m \leq n$ (not overdetermined)
- $y = f(\bar{x})$: $\nabla\mathcal{L}(\bar{x},\bar{\lambda},\bar{\mu})$ ... , $\boxed{p = 0 : \quad Z^T(\nabla_{xx}\mathcal{L})Z > 0 \quad \text{col. of } Z = \text{basis of null}(J_g)}$
- Assume h_i don't contradict each other? Assume full rank($J_{h_{\text{act}}}$)
- $y = f(\bar{x})$: $\nabla\mathcal{L}(\bar{x},\bar{\lambda},\bar{\mu})$ ... , $\boxed{p > 0, \text{ Karush-Kuhn-Tucker (KKT)}: \quad \bar{\mu}_i \geq 0,\ \bar{\mu}_i h_i(\bar{x}) = 0}$ (2nd deriv. cond. not given)

2.3 Unconstrained One Dimension/Independent Variable

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

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

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

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

Sucessive Linear Interpolation [Secant Method]: Not useful, since lines have no unique minimun

Sucessive Parabolic Interpolation: Use 3 pts to approx. a parabola w/ $\boxed{r = 1.324}$ (not guarenteed)

2.4 Unconstrained m-Dimensions/Independent Variables

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

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

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

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

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

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

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

- 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 $\boxed{\vec{h}_{k+1} = \vec{\nabla} f(\vec{x}_{k+1}) - \frac{\vec{\nabla} f(\vec{x}_{k+1})\cdot\vec{\nabla} f(\vec{x}_{k+1}) - \vec{\nabla} f(\vec{x}_k)\cdot\vec{\nabla} f(\vec{x}_{k+1})}{\vec{\nabla} f(\vec{x}_k)\cdot\vec{\nabla} f(\vec{x}_k)}\ \vec{h}_k}$ (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, $\left\{ \min \|\vec{r}(\vec{x})\|^2 :\ \vec{f}_{(\vec{a},\vec{x})} + \vec{r}_{(\vec{x})} = \vec{b} \right\}$

Linear Least Squares — Nonlinear Least Squares

$$\begin{pmatrix} \vdots \\ -\vec{a}_i- \\ \vdots \end{pmatrix} \begin{pmatrix} | \\ \vec{x} \\ | \end{pmatrix} + \begin{pmatrix} | \\ \vec{r} \\ | \end{pmatrix} = \begin{pmatrix} | \\ \vec{b} \\ | \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} | \\ \vec{f}_{(\vec{a},\vec{x})_i} \\ | \end{pmatrix} + \begin{pmatrix} | \\ \vec{r} \\ | \end{pmatrix} = \begin{pmatrix} | \\ \vec{b} \\ | \end{pmatrix}$$

$$\boxed{\phi(\vec{x}) \equiv \tfrac{1}{2}\vec{r}\cdot\vec{r}}\ ,\ \boxed{-\vec{\nabla}\phi(\vec{x}) = -J_r^T\vec{r}}$$

$$\boxed{H_\phi(\vec{x}) = J_r^T J_r + \sum_i H_{r_i}\vec{r}_i}$$

Newton's Method

$$:\ \boxed{H_\phi(\vec{x}_k)\vec{h}_k = -\vec{\nabla}\phi(\vec{x}_k)} \Rightarrow \boxed{\vec{x}_{k+1} = \vec{x}_k + \vec{h}_k}$$

(usually expensive to compute)

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

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

$$\boxed{(J_r^T J_r + \mu_k I)\vec{h}_k = -J_r^T\vec{r}_{(\vec{x}_k)} \;\Rightarrow\; \vec{x}_{k+1} = \vec{x} + \vec{h}_k}$$

$$\Rightarrow \boxed{\begin{pmatrix} J_r^T{}_{(\vec{x})} & \sqrt{\mu_k} I \end{pmatrix} \begin{pmatrix} J_{r(\vec{x})} \\ \sqrt{\mu_k} I \end{pmatrix} \vec{h}_k = \begin{pmatrix} J_r^T{}_{(\vec{x})} & \sqrt{\mu_k} I \end{pmatrix} \begin{pmatrix} -\vec{r}_{(\vec{x}_k)} \\ 0 \end{pmatrix}}$$

Regularization

- 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

Newton's Method

$$\boxed{H_{\mathcal{L}}\vec{h}_k = -\vec{\nabla}\mathcal{L}}$$

KKT Matrix (Eq. Constr)

$$\begin{pmatrix} \nabla_{xx}\mathcal{L} & J_g^T \\ J_g & 0 \end{pmatrix} \begin{pmatrix} \vec{s}_k \\ \vec{\delta}_k \end{pmatrix} = -\begin{pmatrix} \nabla f(\bar{x}) + J_g^T(\bar{x})\bar{\lambda} \\ \vec{g}(\bar{x}) \end{pmatrix}$$

$$\boxed{\begin{pmatrix} B & J^T \\ J & 0 \end{pmatrix} \begin{pmatrix} s \\ \delta \end{pmatrix} = -\begin{pmatrix} w \\ g \end{pmatrix}}$$

$\Rightarrow$ [Sequential] Quadratic Programming (SQP) Problem

$$\min_s \left(\vec{s}_k \cdot \vec{\nabla}_x\mathcal{L} + \tfrac{1}{2}\langle \vec{s}_k|\vec{\nabla}_{xx}\mathcal{L}|\vec{s}_k\rangle \right)$$
$$\text{s.t.} \quad J_{g(\vec{x}_k)}\vec{s}_k + \vec{g}(\vec{x}_k) = 0$$

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

(Column-Space)

<u>Range-Space Method</u>: $\boxed{Bs = -w - J^T\delta}$, $Js = -g \;\to\; JB^{-1}(-w - J^T\delta) = -g$

$$\to \boxed{(JB^{-1}J^T)\delta = g - JB^{-1}w}$$

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

<u>Null-Space Method</u>: $J^T = \begin{pmatrix} Q_\parallel & Q_\perp \end{pmatrix}\begin{pmatrix} R \\ 0 \end{pmatrix}$ $\quad (Q_\parallel \in \mathbb{R}^{n\times m}) \quad \Rightarrow \quad \boxed{\begin{matrix} JQ_\parallel = R^T \\ JQ_\perp = 0 \end{matrix}}$

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

$$\text{Find } u_\perp : \quad Q_\perp^T\left(Bs + J^T\delta = -w\right) \;\to\; (Q_\perp^T B Q_\parallel)u_\parallel + (Q_\perp^T B Q_\perp)u_\perp = -Q_\perp^T w - (\cancel{JQ_\perp})^T\delta$$

$$\boxed{(Q_\perp^T B Q_\perp)u_\perp = -Q_\perp^T w - (Q_\perp^T B Q_\parallel)u_\parallel}$$

$$\text{Find } \delta : \quad Q_\parallel^T\left(J^T\delta = -w - Bs\right) \;\to\; \boxed{R\delta = -Q_\parallel^T w - Q_\parallel^T B(Q_\parallel u_\parallel - Q_\perp u_\perp)}$$

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

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

<u>Penalty Func. Method</u>

$\boxed{\lim_{\rho\to\infty} \vec{x}_\rho = \bar{x}}$ (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) : $\min_{\vec{x}} \mathcal{L}_\rho(\vec{x}) = f(\vec{x}) + \vec{\lambda}_0 \cdot \vec{g}(\vec{x}) + \frac{1}{2}\rho\|g(\vec{x})\|^2$

<u>Barrier Func. Method</u>

$\boxed{\lim_{\rho\to 0} \vec{x}_\rho = \bar{x}}$

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

(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, $f(t_i) = \hat{f}(t_i) = \sum_j x_j \phi_j(t_i)$

$$\hat{f}(t_i) = \sum_j x_j \phi_j(t_i) = \vec{\phi}(t_i) \cdot \vec{x}$$

$\det(A) \neq 0$

Given $\vec{\phi}$, solve for $\vec{x}$

$$A\vec{x} = \begin{pmatrix} \vdots \\ - \ \vec{\phi}(t_i) \ - \\ \vdots \end{pmatrix} \begin{pmatrix} | \\ \vec{x} \\ | \end{pmatrix} = \vec{y} = \begin{pmatrix} \vdots \\ f(t_i) \\ \vdots \end{pmatrix}$$

- Runge Phenom.: As n increases, evenly-spaced t_i could produce a high-dimensional polynomial $\hat{f}(t)$ that tends to be extremely wavey 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: $\max_{t \in [t_1, t_n]} \left| \hat{f} - f = \frac{f^{(n)}(\xi)}{n!} \prod_i (t - t_i) \right| \leq \left| \max \frac{f^{(n)}(t)}{n!} \right| \left| \frac{(n-1)! h^n}{4} \right| = \boxed{\max_{t \in [t_1, t_n]} \left| f^{(n)}(t) \frac{h^n}{4n} \right|} \rightarrow$ error decreases if $f^{(n)}$ 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 → Vandermonde Matrix

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

(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 sucessive t^j look the same at higher j.

3.3 Lagrange Basis Functions (Fund. Polynomials) → Identity Matrix

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

$$\boxed{\phi_j(t) = \frac{l(t)/(t-t_j)}{l(t_j)/(t_j-t_j)} = l(t) \frac{w_j}{t - t_j}}$$

$$\phi_j(t_i) = \delta_{ij} \ \Rightarrow \ \boxed{\vec{\phi}(t_i) = \vec{e}_i}$$

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

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

(Diag. Iden. Matrix)

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

- 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 → better-conditioned.
- $\boxed{\int_{t_1}^{t_n} \hat{f}(t) dt = \sum_{i=1}^{n} y_i \int_{t_1}^{t_n} \phi_i(t) dt}$

3.4 Newton Basis Functions → Low. Triang. Matrix

$$\boxed{\begin{aligned}\phi_j(t) &= (t-t_1)(t-t_2)\dots(t-t_{j-1})\\ \vec{\phi}(t) &= \big[1,(t-t_1),(t-t_1)(t-t_2),\ \dots\big]^T\end{aligned}}$$

$$\boxed{\hat{f}(t) = x_1 + x_2(t-t_1) + \cdots + x_n\phi_n(t)}$$

(Low. Triang. Matrix)

$$\begin{pmatrix} 1 & 0 & 0 & \cdots \\ 1 & t_1-t_2 & 0 & \cdots \\ 1 & t_3-t_2 & (t_3-t_1)(t_3-t_2) & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}\begin{pmatrix}\vdots\\ x_i\\ \vdots\end{pmatrix} = \vec{y}$$

- 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:

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

$$\begin{aligned} y_{n+1} &= \hat{f}_{n+1}(t_{n+1})\\ &= \hat{f}_n(t_{n+1}) + x_{n+1}\phi_{n+1}(t_{n+1})\end{aligned}$$

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

Divided Differences Newton Interpolation:

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

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

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

3.5 Orthogonal Polynomial Basis (no method given)

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

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

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

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

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

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

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

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

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

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:

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

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

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

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

$$\left|\int_a^b \hat{f}(x) - f(x)\ dx\right| \ \le\ \int_a^b \left|\hat{f}(x) - f(x)\right| dx$$

$$\le\ (b-a)\|\hat{f}(x) - f(x)\|_\infty$$

$$\left|\frac{\Delta I}{\Delta f}\right| \ \le\ \boxed{b-a}$$

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

$$\left|\frac{\Delta I/I}{\Delta f/f}\right| \ \le\ \frac{(b-a)/\left|\int_a^b f(x)dx\right|}{1/\|f(x)\|_\infty}$$

$$=\ \boxed{\frac{(b-a)\|f(x)\|_\infty}{\left|\int_a^b f(x)dx\right|}}$$

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

$\hat{f} \in P_{n-1}$: $\quad \hat{f}(x) = \left(\underset{\text{(Lagrange Basis Vectors)}}{\vec{y}\cdot\vec{\phi}(x) = \sum_{i=1}^{n} f(x_i)\,\phi_i(x)}\right) = \left(\underset{\text{(Monomial Basis Vetors)}}{\sum_{j=1}^{n} c_j\, x^{j-1}}\right)$

- $x_1 < \ldots < x_n$
- $f(x_i) = \hat{f}(x_i)$

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

- $\boxed{x_i,\ w_i\ \to\ 2n \text{ max param.}}$
- $a \le x_1 < \ldots < x_n \le b$
- closed if equality, open if not

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

$$\int_a^b \left(\sum_{j=1}^{n} c_j\, x^{j-1}\right) dx\ =\ \sum_{i=1}^{n}\left(\sum_{j=1}^{n} c_j\, x_i^{j-1}\right) w_i$$

$$\sum_{j=1}^{n} c_j\left(\int_a^b x^{j-1}dx\right)\ =\ \sum_{j=1}^{n} c_j\left(\sum_{i=1}^{n} x_i^{j-1}\, w_i\right)$$

(maybe some dot product to isolate terms)

$$\Rightarrow$$

$$\boxed{\sum_{i=1}^{n} f(x_i)\,w_i = \hat{F}(b) - \hat{F}(a)}$$

$$\downarrow$$

$$\boxed{\sum_{i=1}^{n} x_i^{j-1}\, w_i = \frac{b^j}{j} - \frac{a^j}{j}}$$

$$\equiv z_j$$

- $\boxed{z_1\ =\ \sum w_i\ =\ b-a}$

(Vandermode Matrix)

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

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

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

Error Q_n : $\quad g \approx f\ \to\ |Q_n(f) - Q_n(g)|\ \le\ \boxed{\sum |w_i|\cdot\|f-g\|_\infty}\ \Rightarrow\ \boxed{\forall w_i \ge 0\ \to\ \text{cond}(Q_n) = b-a}$

$= \left|\sum w_i\left[f(x_i) - g(x_i)\right]\right|$

(otherwise using Q_n might be unstable.)

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

Newton-Cotes Quadrature [Rule]: $\boxed{n \text{ evenly-spaced } x_i \;\rightarrow\; n \text{ param. for } w_i}$

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

$$f(x) = \boxed{\sum_{m=0} \frac{f^{(m)}(\frac{a+b}{2})}{m!}(x - \frac{a+b}{2})^m}$$

$$T(f) = \frac{b-a}{2} \sum_{m=0} \frac{f^{(m)}(\frac{a+b}{2})}{m!} \frac{(b-a)^m}{2^m} [(-1)^m + 1]$$

$$= \sum_{m=0}^{\text{even}} \boxed{\frac{f^{(m)}(\frac{a+b}{2})}{2^m m!}}(b-a)^{m+1}$$

$$= \boxed{M(f) + \sum_{m=2}^{\text{even}} \boxed{E_m(f)} h^{m+1}}$$

$$S(f) = \boxed{\tfrac{2}{3}M(f) + \tfrac{1}{3}T(f)}$$

$$I(f) = \sum_{m=0} \frac{f^{(m)}(\frac{a+b}{2})}{(m+1)!} \frac{(2x-a-b)^{m+1}}{2^{m+1}}\Big|_a^b$$

$$= \sum_{m=0}^{\text{even}} \frac{f^{(m)}(\frac{a+b}{2})}{2^m(m+1)!}(b-a)^{m+1}$$

$$= \boxed{M(f) + \sum_{m=2}^{\text{even}} \frac{E_m(f)}{m+1} h^{m+1}}$$

Q_1 error is $f^{(2)}$ derivative, not $f^{(1)}$!

$$= \boxed{T(f) - \sum_{m=2}^{\text{even}} m \frac{E_m(f)}{m+1} h^{m+1}}$$

Q_2 error is $f^{(2)}$ & twice as large as Q_1

$$= \boxed{S(f) - \sum_{m=4}^{\text{even}} \frac{m-2}{3} \frac{E_m(f)}{m+1} h^{m+1}}$$

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 exibit the Runge Phenom. $\longrightarrow$ $\boxed{Q_\infty(f) \text{ isn't always } I(f)}$
- $\boxed{\text{Ill-conditioned and unstable}}$: $(n \geq 11 \Rightarrow \exists w_i < 0)$, $(\sum_i^\infty |w_i| \rightarrow \infty)$

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

- $\forall n : \forall w_i > 0 \;\Rightarrow\; \text{cond}(Q) = b - a$
- $\lim_{n\to\infty} C_n(f) = I(f)$
- $\boxed{d_n = n - 1}$
- $\exists$ 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].

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

- $x_i, w_i:\quad x_{n<i\le 2n} = w_{n<i\le 2n} = 0 \ \rightarrow\ \boxed{\begin{pmatrix} 1 & \dots & 1 & 0 & \dots \\ x_1 & \dots & x_n & 0 & \dots \\ x_1^2 & \dots & x_n^2 & 0 & \dots \\ \vdots & & \vdots & \vdots & \end{pmatrix}\begin{pmatrix} \vdots \\ w_n \\ 0 \\ \vdots \end{pmatrix} = \vec{z}(a,b)}$ $\quad\boxed{\text{usually } x_i \notin \mathbb{Q}}$
- Ortho. Poly. : $\langle p_n(x)|x^k\rangle_{ab} = 0 \Rightarrow \boxed{\begin{aligned} x_i:\quad & p_n(x_i)=0, \quad x_i \in \mathbb{R}, \\ & x_i \neq x_{j\neq i}, \quad x_i \in (a,b)\end{aligned}}$ $(k=0,\ \dots,\ n-1)$

  $[-1,1]$ — Legendre
  $(-\infty,\infty)$ — Hermite
  $[0,\infty)$ — Laguerre
- Interval Transform : $\boxed{\int_a^b f(t)\,dt = \frac{b-a}{\beta-\alpha}\int_\alpha^\beta f(t)\,dx \qquad t = \frac{(b-a)x+a\beta-b\alpha}{\beta-\alpha}}$
- $\forall n : \forall w_i > 0 \Rightarrow \text{cond}(Q) = b-a$
- $\lim_{n\to\infty} G_n(f) = I(f)$
- $n = 2m+1 \ \rightarrow\ \frac{a+b}{2} \in \{x_i\}_n$; otherwise usually $\{x_i\}_n \cup \{x_i\}_{\neq n} = 0 \ \rightarrow\ \boxed{\text{Not progressive}}$
- Progressive Gauss-Kronrod, K_{2n+1} : n 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} : $2n+1$ 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 Gauus-Randau : $x_i \in [a,b)$ or $(a,b]$ $\rightarrow \boxed{d = 2n-2}$

  Closed Gauus-Lobatto : $x_i \in [a,b]$ $\rightarrow \boxed{d = 2n-3}$

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

- $\lim_{k\to\infty} C_{k,n} = \sum_{j=1}^{k\to\infty}\left[\sum_{i=1}^{n} w_i f(x_{ji})\right] = \sum_{i=1}^{n}\frac{w_i}{h_k}\left[\sum_{j=1}^{k\to\infty} h_k f(x_{ji})\right] = I(f)\sum_{i=1}^{n}\frac{w_i}{h_k} = I(f)$

  $h_k = (b-a)/k$
  $\ge (x_{jn} - x_{j1})$
  $\boxed{d \ge 0} \Rightarrow \sum w_i = h_k$
- 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 $1/\sqrt{n} \to 0$).

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]

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

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

$$\begin{aligned} T(f, \tfrac{h}{2^k}) &= I(f) + 2^k \left[a_1(\tfrac{h}{2^k})^3 + \mathcal{O}(\tfrac{h}{2^k}^5)\right] \\ T_{k,j=0} &= I(f) + ha_1[\tfrac{h}{2^k}]^2 + h\mathcal{O}([\tfrac{h}{2^k}]^4) \\ 4T_{k+1,0} &= 4I(f) + ha_1[\tfrac{h}{2^k}]^2 + \tfrac{h}{4}\mathcal{O}([\tfrac{h}{2^k}]^4) \end{aligned} \quad \Rightarrow \quad \begin{aligned} T_{k+1,j+1} &\equiv \frac{4^{j+1}\ T_{k+1,j} - T_{k,j}}{4^{j+1} - 1} \qquad {\scriptstyle (1\leq j\leq k)} \\ &\boxed{I(f) = T_{k,j} + \mathcal{O}(h^{2j+2})} \end{aligned}$$

5 Numerical Differentiation

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

5.1 Finite-Difference Approx

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

- 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]

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

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

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

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

$$
\boxed{\big(D_n(f)\big)(a) \equiv \frac{df}{dx}(a) = \frac{d\hat{f}}{dx}(a) = \sum_{i=1}^{n} f(x_i)\phi_i'(a) = \sum_{i=1}^{n} f(x_i) w_i}
$$

- $\boxed{x_i,\ w_i \ \rightarrow \ 2n \text{ max param.}}$

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

(maybe some dot product to isolate terms)

- $\boxed{z_1 = \sum w_i = 0}$

(Vandermode Matrix)

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

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

$$k\text{-th Order}: \quad f(t, y, \ldots, y^{(k)}) = 0 \qquad \text{Linear}: \quad \vec{a}(t) \cdot [y(t), \ldots, y^{(k)}(t)]^T = b(t)$$

$$\text{Autonomous}: \quad f(y, \ldots, y^{(k)}) = 0 \qquad \text{Homogeneous}: \quad b(t) = 0$$

<u>k-th Order System of n Coupled ODEs</u> :

$$\vec{y}_i'(t) = \begin{bmatrix} y_i'(t) \\ u_2'(t) \\ \vdots \\ u_{k-1}'(t) \\ u_k'(t) \end{bmatrix} = \begin{bmatrix} u_2(t) \\ u_3(t) \\ \vdots \\ u_k(t) \\ f(t, u_1, \ldots, u_k) \end{bmatrix} \Rightarrow \boxed{\vec{y}'(t) = \begin{bmatrix} \vec{y}_1'(t) \\ \vdots \\ \vec{y}_n'(t) \end{bmatrix} = \vec{f}(t, \vec{y}) = \begin{bmatrix} \vec{f}_1(t, \vec{y}) \\ \vdots \\ \vec{f}_n(t, \vec{y}) \end{bmatrix}}$$

<u>Nonautonomous → Autonomous</u>: $\boxed{\vec{y}' = f(t, \vec{y}) \ \rightarrow \begin{bmatrix} \vec{y}' = f(y_{n+1}, \vec{y}) \\ y_{n+1}' = 1 \end{bmatrix}}$

$\boxed{\vec{y}(t_0) \text{ for an IVP like } \{u''(t) = f(t),\ u(t_0) = \alpha,\ u'(t_0) = \beta\} \text{ contains the IC.}}$

6.1 ODE Stability (Conditioning)

<u>Unique/Exists if Lipschitz Continuous:</u>

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

<u>$\hat{y}' = \hat{f}(t, \hat{y})$</u> : $\|\hat{y} - \vec{y}\|_\infty \leq e^{L(t-t_0)}\|\hat{y}(t_0) - \vec{y}(t_0)\|_\infty + \frac{e^{L(t-t_0)}-1}{L}\left(\max \|\hat{f} - \vec{f}\|\right)$ (cont./well-posed but possibly sensitive)

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

<u>Asymptotically Stable</u>: $\lim_{t\to\infty} \|\hat{y}(t) - \vec{y}(t)\| = 0$

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

<u>Linear System of ODEs</u>

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

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

<u>Noninear System of ODEs</u>

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

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

$\rightarrow$

- A is diagonalizable / J_f is diagonalizable

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

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

- A isn't diagonalizable / J_f isn't diagonalizable

$\rightarrow$ Stable [Cond.] if $\forall\lambda_i$
- $\text{Re}(\lambda_i) \ \leq \ 0$
- $\text{Re}(\lambda_i) \ < \ 0$ if λ_i isn't simple.

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

$\rightarrow$ Might not be long-term stable

6.2 Algorithm Stability and Error

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

Global [Trunc.] Error (Stability): $\vec{e}_k = \vec{y}_k - \vec{y}(t_k) \;=\; \mathcal{O}(\widehat{h_k}^{\,p})$ (under "reasonable" conditions)

Growth/ Amplification : Factor, g

- $y' = \lambda y \;\Rightarrow\; y_k = g^k y_0 \begin{cases} |g| \le 1 \;\rightarrow\; \text{Stable} \\ |g| > 1 \;\rightarrow\; \text{Unstable} \end{cases}$
- $\vec{e}_{k+1} = g\vec{e}_k + \vec{l}_{k+1} \begin{cases} \rho(g) \le 1 \;\rightarrow\; \text{Stable} \\ \rho(g) > 1 \;\rightarrow\; \text{Unstable} \end{cases}$ $\left(\begin{array}{c}\text{Spectral Radius, } \rho(\mathbb{R}^{n\times n}), \\ \text{may vary with } t\end{array}\right)$

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

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

6.3 ODE Stiffness

[Asymptotic] Stiffness:

- Rapid asymp. decay to convergence; $\mathrm{Re}(\lambda_i(J_f)) \ll \sim 0$ 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; $|\mathrm{Im}(\lambda_i(J_f))| \gg \sim 0$ and differ greatly in magnitude. Treatment not given.

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

Local Error Tolerance : $\frac{h^{p+1}}{(p+1)!}\|\vec{y}^{(p+1)}(t)\| \le tol \;\Rightarrow\; \boxed{h_k \;\lesssim\; \sqrt[p+1]{(p+1)! \cdot tol / \|\vec{y}_k^{(p+1)}\|}}$ $\left(\begin{array}{c}\text{Could use finite} \\ \text{diff for } \|\vec{y}_k''\|\end{array}\right)$

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

- $g:\; y_k = (1 + h_k\lambda)^k y_0 \;\Rightarrow\; e^{\lambda h} = g + \mathcal{O}(h^2)$ $\boxed{p{=}1}$, $|1 + h\lambda| \le 1 \;\rightarrow$ $\boxed{\text{Stable if } \lambda : |1/h + \lambda| \le 1/h}$
- $g:\; \vec{e}_{k+1} = \vec{y}_{k+1} - \vec{y}(t_k+h_k) = \left[\vec{y}_k + h_k\vec{y}'_k\right] - \left[\vec{y}(t_k) + h_k\vec{y}'(t_k) + \mathcal{O}(h_k^2)\right]$

$$= \left[\vec{y}_k - \vec{y}(t_k)\right] + h_k\left[\vec{f}(t_k,\vec{y}_k) - \vec{f}(t_k,\vec{y}(t_k))\right] - \mathcal{O}(h_k^2) \quad \boxed{p{=}1}$$

$$= \vec{e}_k + h_k\bar{J}_f\vec{e}_k - \mathcal{O}(h_k^2) \quad , \quad \left(\begin{array}{c}\text{From Mean} \\ \text{Value Theorem}\end{array} \;:\; \bar{J}_f\vec{e}_k = \int_0^1 J_f\big(t_k,\; \alpha\vec{y}_k + (1-\alpha)\vec{y}(t_k)\big)\,d\alpha \cdot \vec{e}_k\right)$$

$$= \left[I + h_k\bar{J}_f\right]\vec{e}_k + \vec{l}_{k+1} \;\Rightarrow\; \boxed{\rho(I + h_k\bar{J}_f) \le 1 \;\rightarrow\; \text{Stable}}$$

- Stiffness Tolerance : $\boxed{h_k \cdot \min(\mathrm{Re}(\lambda_i(J_f))) \ll -1}$ (small tolerance, not uncond. stable)

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

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

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

6.5 [Exp.] Runge-Kutta Algorithms

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

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

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

$$= \boxed{\vec{y}_k + \tfrac{h_k}{2}\left[\vec{f}(t_k,\vec{y}_k) + \vec{f}\big(t_k+h_k,\; \vec{y}_k + h_k \vec{f}(t_k,\vec{y}_k)\big)\right]}$$

$$= \vec{y}_k + \tfrac{h_k}{2}\left[k_1 + k_2\right]$$

$$\vec{c} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} & \\ 1 & \end{bmatrix} = a \quad , \left[\tfrac{1}{2}, \tfrac{1}{2}\right]^T = \vec{b}$$

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

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

$$\vec{c} = \begin{bmatrix} 0 \\ 1/2 \\ 1/2 \\ 1 \end{bmatrix}, \begin{bmatrix} & & & \\ \frac{1}{2} & & & \\ 0 & \frac{1}{2} & & \\ 0 & 0 & 1 & \end{bmatrix} = a \quad , \left[\; \tfrac{1}{6} \quad \tfrac{2}{6} \quad \tfrac{2}{6} \quad \tfrac{1}{6} \;\right]^T = \vec{b}$$

- 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: $k_i = \vec{f}(t_k + c_i h_k,\; \vec{y}_k + h_k \sum_{j}^{s} a_{ij} k_j)$

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

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

$$\boxed{\vec{y}_{k+1} \;=\; \sum_{i=1}^{m} \alpha_i y_{k-(i-1)} \;+\; h \sum_{i=0}^{m} \beta_i \vec{y'}_{k+1-i}} \quad \text{or} \quad \boxed{\sum_{i=0} a_i y_{k+1-i} = h \sum_{i=0} b_i y'_{k+1-i} \quad \text{(Explicit if } \beta_0 = 0\text{)}}$$

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

$$\boxed{\vec{y}_{k+1} \;=\; \vec{y}_k + h \sum_{i=0}^{m} \beta_i \vec{y'}_{k+1-i}} \;\Leftrightarrow\; \begin{aligned} \vec{y}_{k+1} - \vec{y}_k &= \sum \vec{y'}_{k+1-i} \int_{t_k}^{t_k+h} \phi_{k+1-i}(t)\, dt \\ \frac{t^j_{k+1}}{j} - \frac{t^j_k}{j} &= \sum t^{j-1}_{k+1-i} w_{k+1-i} \end{aligned}$$

2-Step AB : $\begin{matrix} t_{k-1}=0 \\ h=1 \end{matrix}$,

$$\begin{array}{ccc} y_k & y'_k & y'_{k-1} \end{array} \qquad\qquad y_{k+1}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ t_k & 1 & 1 \\ t_k^2 & 2t_k & 2t_{k-1} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ h\beta_1 \\ h\beta_2 \end{bmatrix} = \begin{bmatrix} 1 \\ t_{k+1} \\ t_{k+1}^2 \end{bmatrix} \quad \text{or} \quad \begin{array}{cc} t_k^{j-1} & t_{k-1}^{j-1} \end{array} \begin{bmatrix} 1 & 1 \\ t_k & t_{k-1} \end{bmatrix} \begin{bmatrix} h\beta_1 \\ h\beta_2 \end{bmatrix} = \overset{(t^j_{k+1} - t^j_k)/j}{\begin{bmatrix} t_{k+1} - t_k \\ (t_{k+1}^2 - t_k^2)/2 \end{bmatrix}}$$

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

2-Step AM/ Trapezoid Rule : $\begin{matrix} t_k=0 \\ h=1 \end{matrix}$

$$\begin{array}{cc} t_{k+1} & t_k \end{array} \begin{bmatrix} 1 & 1 \\ t_{k+1} & t_k \end{bmatrix} \begin{bmatrix} h\beta_0 \\ h\beta_1 \end{bmatrix} = \overset{(t^j_{k+1} - t^j_k)/j}{\begin{bmatrix} t_{k+1} - t_k \\ (t_{k+1}^2 - t_k^2)/2 \end{bmatrix}} \quad \text{or} \quad \vec{y}_k + h\vec{y'}_k + \frac{h^2}{2}\left[\frac{\vec{y'}_{k+1} - \vec{y'}_k}{h}\right]$$

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

- $g:\; y_k = \left(\frac{1+h\lambda/2}{1-h\lambda/2}\right)^k y_0 \;\Rightarrow\; e^{\lambda h} = g + \mathcal{O}(h^3)$ $\boxed{p{=}2}$, $\boxed{|g| < 1 \;\rightarrow\; \text{Unconditionally Stable}}$
- $g:\; \rho\big((I+hJ_f)(I-hJ_f)^{-1}\big) < 1$ $(sic\ hJ_f)$ $\rightarrow$ $\boxed{\text{Unconditionally Stable} \rightarrow \text{Greater Stiffness Tol.}}$
- Local Error Tolerance : $\boxed{\frac{2}{2+1}\frac{1}{2^2 2!}\left\| \vec{f}''\big(t_k+h/2,\; \vec{y}(t_k+h/2)\big) \right\| h^3 \;\lesssim\; tol}$

Backwards Differentiation Formula (BDF) Methods : $\boxed{\vec{y}_{k+1} \;=\; \sum_{i=1}^{m} \alpha_i\, \vec{y}_{k-(i-1)} + h\beta_0 \vec{y}'_{k+1}}$, $\vec{y}'_{k+1} = \sum \vec{y}_{k+1-i}\phi'_{k+1-i}(t_{k+1})$, $\frac{d(t^{j-1})}{dt}(t_{k+1}) = \sum t^{j-1}_{k+1-i} w_{k+1-i}$

2-Step BDF : $\begin{matrix} t_{k-1}=0 \\ h=1 \end{matrix}$,

$$\begin{array}{ccc} y_k & y_{k-1} & y'_{k+1} \end{array} \qquad \begin{bmatrix} 1 & 1 & 0 \\ t_k & t_{k-1} & 1 \\ t_k^2 & t_{k-1}^2 & 2t_{k+1} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ h\beta_0 \end{bmatrix} = \begin{array}{c} y_{k+1} \\ \begin{bmatrix} 1 \\ t_{k+1} \\ t_{k+1}^2 \end{bmatrix} \end{array} \quad \text{or} \quad \begin{array}{ccc} t^{j-1}_{k+1} & t^{j-1}_{k} & t^{j-1}_{k-1} \end{array} \begin{bmatrix} 1 & 1 & 1 \\ t_{k+1} & t_k & t_{k-1} \\ t_{k+1}^2 & t_k^2 & t_{k-1}^2 \end{bmatrix} \begin{bmatrix} 1/h\beta_0 \\ -\alpha_1/h\beta_0 \\ -\alpha_2/h\beta_0 \end{bmatrix} = \begin{array}{c} (j-1)(t^{j-2}_{k+1}) \\ \begin{bmatrix} 0 \\ 1 \\ 2t_{k+1} \end{bmatrix} \end{array}$$

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

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

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

$$\sum_{i=0}^{m}\left[b_{k+1-i}\right] \frac{d(t^{j-1})}{dt}_{k+1-i} = \widehat{\mathcal{I}}\frac{d(t^{j-1})}{dt} = \sum_{i=0}^{m} a_{k+1-i}\, t^{j-1}_{k+1-i}$$

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

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

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

$$\sum_{i=0}^{m}\left[a_{k+1-i}\right] t^{j-1}_{k+1-i} = \widehat{\mathcal{D}}t^{j-1} = \sum_{i=0}^{m} b_{k+1-i}\, \frac{d(t^{j-1})}{dt}_{k+1-i}$$

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

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

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

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

$$\begin{bmatrix} \hat{y}_{k+1} \\ h\hat{y}'_{k+1} \\ (h^2/2!)\hat{y}''_{k+1} \\ (h^3/3!)\hat{y}'''_{k+1} \end{bmatrix} = \begin{bmatrix} [y_k + hy'_k + (h^2/2!)y''_k + (h^3/3!)y'''_k] \\ h[y'_k + hy''_k + (h^2/2!)y'''_k] \\ (h^2/2!)[y''_k + hy'''_k] \\ (h^3/3!)[y'''_k] \end{bmatrix}$$

$$\Rightarrow \quad \vec{y}_{k+1} = \vec{\hat{y}}_{k+1} + \alpha\vec{r}$$

$$\begin{bmatrix} y_{k+1} \\ hy'_{k+1} \\ \frac{h^2}{2!}y''_{k+1} \\ \frac{h^3}{3!}y'''_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{y}_{k+1} \\ h\hat{y}'_{k+1} \\ \frac{h^2}{2!}\hat{y}''_{k+1} \\ \frac{h^3}{3!}\hat{y}'''_{k+1} \end{bmatrix} + h\left(y'_{k+1} - \hat{y}'_{k+1}\right) \begin{bmatrix} r_1 = \frac{3}{8} \\ 1 \\ r_3 = \frac{3}{4} \\ r_4 = \frac{1}{6} \end{bmatrix}$$

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

7 Boundary Value Equations for ODEs

$$\vec{y}'(t) = \vec{f}_{(t,\vec{y})}; \quad \vec{g}\Big(\vec{y}_{(a)},\ \vec{y}_{(b)}\Big) = 0; \quad \text{E.g., } u''(t) = f(t),\ u(a) = \alpha,\ u(b) = \beta$$

$\boxed{\vec{y}(t_0) \text{ isn't known for a BVP like } \{u''(t) = f(t),\ u(a) = \alpha,\ u(b) = \beta\} \text{ since } u'(a) \text{ isn't given.}}$

Separated [Boundary Conditions]: $\vec{g}$ s.t. $g_1 = g_1(\vec{y}_{(a)}),\quad g_2 = g_2(\vec{y}_{(b)}),\quad \ldots \quad g_{2n}$

Linear [Boundary Conditions]: $\vec{g}$ s.t. $B_a\vec{y}_{(a)} + B_b\vec{y}_{(b)} = \begin{bmatrix} & | & \\ - & B_a & - \\ & | & \end{bmatrix}\begin{bmatrix} | \\ \vec{y}_{(a)} \\ | \end{bmatrix} + \begin{bmatrix} & | & \\ - & B_b & - \\ & | & \end{bmatrix}\begin{bmatrix} | \\ \vec{y}_{(b)} \\ | \end{bmatrix} = \vec{c}$

Linear BVP: Linear ODE + Linear BC $\Rightarrow \vec{y}' = A(t)\vec{y} + \vec{b}(t)$

- $\underbrace{Y(t)}_{\text{Fund. Sol. Matrix}} = \begin{bmatrix} \ldots & \overset{|}{y_i(t)} & \ldots \\ & | & \end{bmatrix}$: $\underbrace{\Big(\vec{y}_i' = A(t)\vec{y}_i\ ,\quad \vec{y}_i(a) = \vec{e}_i\Big)}_{\text{Sol. Modes}} \Rightarrow$ $\boxed{Q \equiv B_aY(a) + B_bY(b)}$ $\boxed{\exists Q^{-1} \Leftrightarrow \text{exists a unique solution to BVP}}$
- $$\begin{aligned} \Phi(t) &\equiv Y(t)Q^{-1} = Y(t)Y^{-1}(a)B_a^{-1} + Y(t)Y^{-1}(b)B_b^{-1} \\ \Phi^{-1}(s) &\equiv QY^{-1}(s) = B_aY(a)Y^{-1}(s) + B_bY(b)Y^{-1}(s) \end{aligned}$$
- $G(s,t) = \begin{cases} \Phi(t)B_a\Phi(a)\Phi^{-1}(s) & s \in [a,t] \\ -\Phi(t)B_b\Phi(b)\Phi^{-1}(s) & s \in (t,b] \end{cases} \Rightarrow \boxed{\vec{y}(t) = \Phi(t)\vec{c} + \int_a^b G(s,t)\vec{b}(s)\,ds}$
- $\vec{y}(t) \le \kappa\left(\|\vec{c}\| + \int_a^b \|\vec{b}(s)\|\,ds\right) \Rightarrow \hat{y}(t) - \vec{y}(t) = \boxed{\Delta\vec{y}(t) \le \kappa\left(\|\Delta\vec{c}\| + \int_a^b \|\Delta\vec{b}(s)\|\,ds\right)}$
- Conditioning/Stability depends on both the growth of the solution modes and BC. A BVP's solution is determined at all the points simultaneously. [dictonomy skipped]

7.1 Intro Methods

Shooting Method: Guess the IVP init. cond. as $\hat{u}'(a)$ and use an IVP method to approx. $u(t)$. Iterate with a better guess by comparing the end BC to $\hat{u}(b)$.

- E.g. $u'', u(a), u(b)$: $y_0 = \begin{bmatrix} u(a) \\ \hat{u}'(a) \end{bmatrix} \rightarrow y_k = \begin{bmatrix} \hat{u}(t) \\ \hat{u}'(t) \end{bmatrix} \rightarrow y_n = \begin{bmatrix} \hat{u}(b) \\ \hat{u}'(b) \end{bmatrix}$, and compare $\hat{u}(b)$ to $u(b)$.
- IVP might be unstable even if BVP is stable, or the IVP for an init. guess might not be integrable over the interval.
- Multiple shooting (over subintervals) improves conditioning but is a larger system to solve.
- The approx. sol. isn't cont. or differentiable, since the points are discrete.

Finite Difference Method: Solve a system to approx. $u(t_i)$ from the BC and a set of mesh points, t_i, by replacing derivatives with a finite diff. approx.

- E.g. $u'', u(a), u(b)$: $y_0 = u_{(a)},\ y_{n+1} = u_{(b)};\ u'(t_{1\le i\le n}) = \frac{y_{i+1}-y_{i-1}}{2h},\ u''(t_i) = \frac{y_{i+1}-2y_i+y_{i-1}}{h^2}$
- The finite diff. means the system matrix (or Jacobian if nonlinear) is typically sparse/banded.
- Infinite mesh points converge to the solution if the finite diff. method is consistent (truncation error goes to 0 as h does) and stable (small perturb. are bounded).
- The approx. sol. isn't cont. or differentiable, since the points are discrete.

7.2 Weighted Residual Methods (Interpolation)

E.g., $u''(t) = f(t) \;\approx\; v''(t,\vec{x}) = \sum_i^n x_i \phi_i''(t)$

$u(a), u(b)$

$r(t,\vec{x}) = \vec{x} \cdot \vec{\phi}''(t) - f(t)$

If $\int_a^b r(t,\vec{x})\, w_i(t)\, dt = 0$ $\Rightarrow A\vec{x} = \vec{b}$:

$$\boxed{A = \int_a^b \vec{\phi}''(t)\, \vec{w}^T(t)\, dt = 0 \qquad \vec{b} = \int_a^b f(t)\, \vec{w}(t)\, dt}$$

$$\boxed{\text{Solve a system for } \vec{x} \text{ from the BC, } v(a) = u(a),\ v(b) = u(b), \text{ and } 1 < i < n:\ b_i = A_i \vec{x}}$$

<u>Collocation Method</u>: Solve a system for $\vec{x}$ from the BC and the points interpolated at $u''(t_{1<i<n}) = \sum x_i \phi_i''(t_i) = v''(t_i)$; $w_i(t) = \delta(t - t_i)$

$r(t_i,\vec{x}) = \boxed{\vec{x} \cdot \vec{\phi}''(t_i) - f(t_i) = 0}$

- E.g., $t_1 = a < ... < t_n = b:\ v(a) = u(a),\ v(b) = u(b),\ v''(t_{1<i<n}) = \sum x_i \phi_i''(t_i) = f(t)$
- System doesn't necessarily converge or is exact at the interpolated points, since the derivative (u'') is interpolated, not the function itself (u).
- Basis func. w/ global support (e.g., poly. or trig. func.) yield a spectral/pseudospectral method. They're very accurate for the number of points used but non-orthog. bases require solving a dense system, and some are ill-conditioned (like monomials). An orthog. basis can be solved efficiently with a FFT.
- Basis func. w/ compact support (e.g., B-splines) yield a finite element method. The basis functions are near-orthog, so the system is usually well-conditioned and often sparse.

<u>Least Squares Residual Method</u>: $w_i(t) = \frac{\partial r}{\partial x_i} = \phi_i''(t)$

$$\min \tfrac{1}{2} \int_a^b \|r\|^2\, dt \;\Rightarrow\; 0 \;=\; \int_a^b r(t,\vec{x})\, \phi_i''(t)\, dt \;=\; \boxed{\sum_{j=1}^n \left(\int_a^b \phi_j''(t) \phi_i''(t)\, dt \right) x_j - \int_a^b f(t) \phi_i''(t)\, dt}$$

$$0 \;=\; A\vec{x} - \vec{b}$$

- A usually isn't symmetric, and entries involve 2nd derivatives.

<u>Galerkin Method</u>: $w_i(t) = \phi_i(t)$; $\phi_i(t)$ satisfy the relevant BC/homogeneous BC (HBC).

$$\int_a^b r(t,\vec{x})\, \phi_i(t)\, dt = 0 \;\Rightarrow\; \int_a^b f(t) \phi_i(t)\, dt = \int_a^b v''(t,\vec{x}) \phi_i(t)\, dt = \cancelto{0}{v'(t,\vec{x}) \phi_i(t) \Big|_a^b} - \int_a^b v'(t,\vec{x}) \phi_i'(t)\, dt$$

$$\boxed{\text{(Load vector)} \quad b_i = \int_a^b f(t) \phi_i(t)\, dt = \sum_{j=1}^n \left(- \int_a^b \phi_j'(t) \phi_i'(t)\, dt \right) x_i = A_i \vec{x} \quad \text{(Stiffness matrix } A \text{ is sym.)}}$$

- E.g., $v(a) = u(a),\ v(b) = u(b),\ b_{1<i<n} = A_i \vec{x}$ (All BC/HBC are satisfied)
- The approx. solution using a finite number of basis functions. best approx. the true sol. when the residual is orthog. to the span of all the basis functions.
- Bases may have global support or compact local support.
- Approx. sol. might have a lower order differentiability.
- Approx. sol. is integrable, but need not be cont./differentiable [like pointwise-interpolation].

8 Partial Differential Equations (PDEs)

$$\begin{aligned}
\text{Transport Eq. (Linear)}: \quad u_t = cu_x &\ + \ f(t,x) \\
\text{Diffusion Eq. (Parabolic)}: \quad u_t = cu_{xx} &\ + \ f(t,x) \\
\text{Wave Eq. (Hyperbolic)}: \quad u_{tt} - cu_{xx} &\ = \ f(t,x) \\
\text{Laplace/Poisson Eq. (Elliptic)}: \quad u_{yy} + cu_{xx} &\ = \ f(x,y)
\end{aligned}$$

- Time-Dependant Functions (Diffusion/Wave) can use IVP techniques to solve.
- Time-Independant Functions (Laplace) can use BVP techniques to solve.
- Everything skipped.