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

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

- $\boxed{\vec{g}(\vec{x}) \equiv \vec{x} - J_f(\vec{x})^{-1}\vec{f}(\vec{x})} \Rightarrow \quad J_g(\bar{x}) = \underbrace{\cancel{I - J_f(\bar{x})^{-1}J_f(\bar{x})}}_{\text{(if } J_f(\bar{x}) \text{ is nonsingular)}} + \sum_{i=1}^{n} H_i(\bar{x})f_i(\bar{x})$ $\quad$ 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:
$$\boxed{\begin{aligned} f(\bar{x}+\vec{s}) - f(\bar{x}) &= \overset{\alpha_1\in(0,1)}{\vec{\nabla} f(\bar{x}+\alpha_1\vec{s})} \cdot \vec{s} = \vec{\nabla} f(\bar{x}) \cdot \vec{s} + \tfrac{1}{2}\langle \vec{s}|\overset{\alpha_2\in(0,1)}{H_{f(\bar{x}+\alpha_2\vec{s})}}|\vec{s}\rangle \geq 0 \\ f(\vec{x}+s\hat{u}) - f(\vec{x}) &= \vec{\nabla} f(\vec{x}+\alpha_1 s\hat{u}) \cdot s\hat{u} = \vec{\nabla} f(\vec{x}) \cdot \vec{s} + \tfrac{s^2}{2}\langle \hat{u}|H_{f(\vec{x}+\alpha_2\vec{s})}|\hat{u}\rangle \end{aligned}}$$

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

2.2.1 Unconstrained Optimization Conditions

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

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

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

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)
- $y = f(\bar{x}) : \ \nabla \mathcal{L}(\bar{x}, \bar{\lambda}, \bar{\mu}) \ \ldots$, $\boxed{p = 0 : \quad Z^T(\nabla_{xx}\mathcal{L})Z > 0 \quad \text{col. of } Z = \text{basis of null}(J_g)}$
- Assume h_i don't contradict each other? Assume full rank($J_{h_{\text{act}}}$)
- $y = f(\bar{x}) : \ \nabla \mathcal{L}(\bar{x}, \bar{\lambda}, \bar{\mu}) \ \ldots$, $\boxed{p > 0, \ \text{Karush-Kuhn-Tucker (KKT)} : \quad \bar{\mu}_i \geq 0, \ \bar{\mu}_i h_i(\bar{x}) = 0}$ (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: $f(\bar{x}) = f(\vec{x} + \vec{h}) \ \approx \ f(\vec{x}) + \vec{\nabla} f(\vec{x}) \cdot \vec{h} + \frac{1}{2}\langle \vec{h} | H_f(\vec{x}) | \vec{h} \rangle$

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

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] :

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

- 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)

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

<u>Levenberg-Marquardt Method (Gauss-Newton + Line Search)</u>:

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

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

<u>Regularization</u>

- 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

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

<u>KKT Matrix (Eq. Constr)</u>

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

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

$\Rightarrow$

<u>[Sequential] Quadratic Programming (SQP) Problem</u>

$$\min_s \left(\vec{s}_k \cdot \vec{\nabla}_x\mathcal{L} + \tfrac{1}{2}\langle \vec{s}_k|\vec{\nabla}_{xx}\mathcal{L}|\vec{s}_k\rangle \right)$$

$$\text{s.t.} \quad J_{g(\vec{x}_k)}\vec{s}_k + \vec{g}_{(\vec{x}_k)} = 0$$

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

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

- 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.

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

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

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

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

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

- 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.

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

Penalty Func. Method

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

Barrier Func. Method

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

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

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 $f(t)$ 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.

3.1 Taylor Series Polynomial Interpolation

$$\begin{aligned} f_n(t) &= f(t_0) + f'(t_0)(t-t_0) + \frac{f''(t_0)}{2}(t-t_0)^2 + \ldots + \frac{f^{(n)}(t_0)}{n!}(t-t_0)^n \\ f_n(t+h) &= f(t) + f'(t)h + \frac{f''(t)}{2}h^2 + \ldots + \frac{f^{(n)}(t)}{n!}h^n \end{aligned}$$

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

3.2 Monomial Basis Functions → Vandermonde Matrix

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

(Full, Dense Vandermonde Matrix)

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

- 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

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

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

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

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

$$f(t_j) = x_j = y_i$$

(Diag. Iden. Matrix)

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

- Finding w_j is $\mathcal{O}(n^2)$ work.
- Finding $f(t)$ 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.
- $\boxed{\displaystyle\int_{t_1}^{t_n} f(t)dt = \sum_{i=1}^{n} y_i \int_{t_1}^{t_n} \phi_i(t)dt}$

3.4 Newton Basis Functions → Low. Triang. Matrix

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

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

(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:

$$f_{n+1}(t) = f_n(t) + x_{n+1}\phi_{n+1}(t)$$

$$\begin{aligned} y_{n+1} &= f_{n+1}(t_{n+1})\\ &= f_n(t_{n+1}) + x_{n+1}\phi_{n+1}(t_{n+1})\end{aligned}$$

$$\Rightarrow \boxed{f_{j+1}(t) = f_j(t) + \tfrac{y_{j+1}-f_j(t_{j+1})}{\phi_{j+1}(t_{j+1})}\phi_{j+1}(t)}$$

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: $\boxed{f_{k+1}(t) = \big[A(k)t + B(k)\big]f_k(t) - C(k)f_{k-1}(t)}$ $(A(k)\neq 0)$

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).