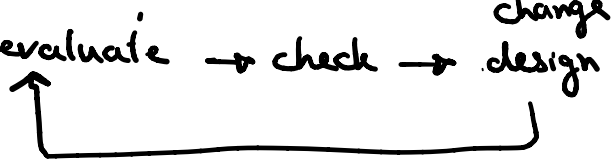
MUMEF41

A typical engineering design optimisation process. → iterative!



User: define a problem specification

- Parameters
- Constants
- Objectives
- Constraints

- define the merits of potential design
- initial design.

There! Automize the process

- cost
- time
- no improvement

Constraints:

Each constraint limits the set of possible soln.

→ feasible design points do not violate any constraints

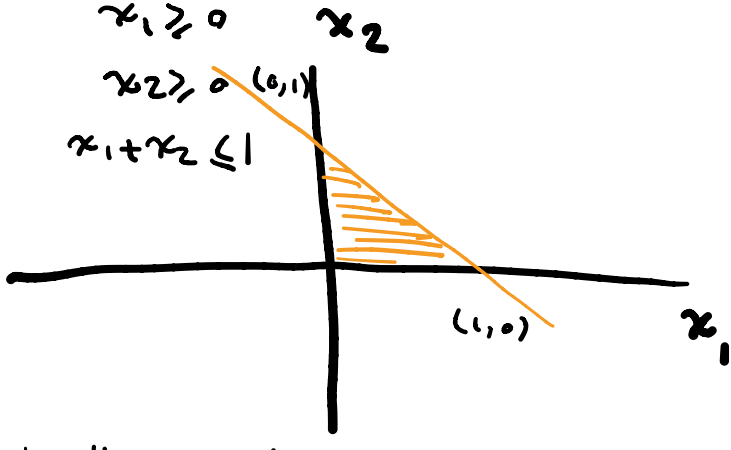
Ex.

$$\underset{x_1, x_2}{\text{minimize}} \quad f(x_1, x_2)$$

subject to $x_1 \geq 0$

$x_2 \geq 0$

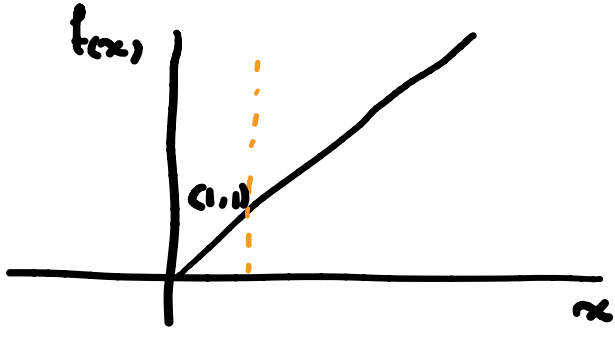
$x_1 + x_2 \leq 1$



Constraints typically $\leq$, $\geq$, $=$

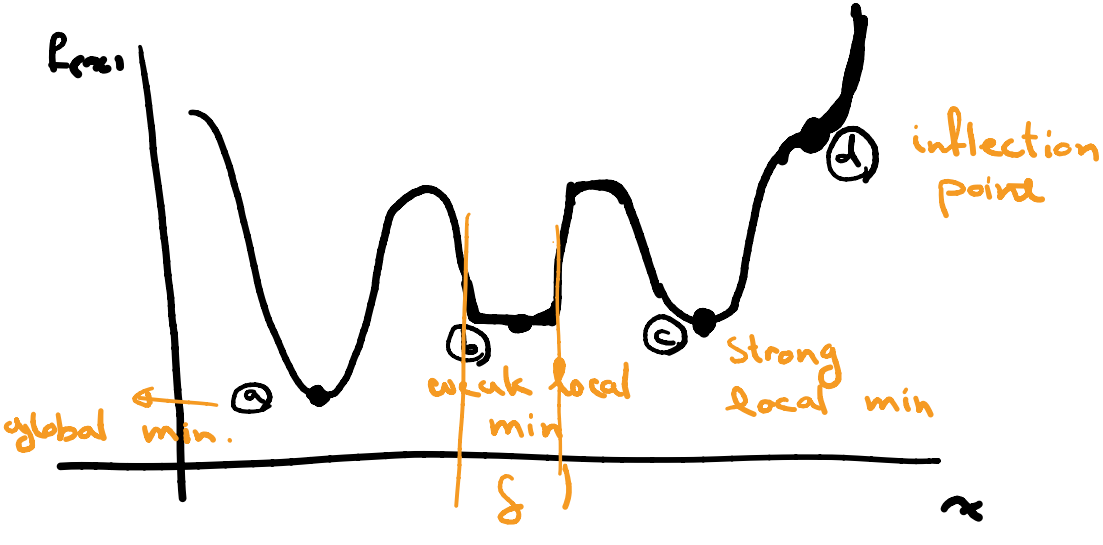
Ex. A potential issue with not including the boundary

minimize x
x
subject to $x > 1$



Critical Points:

Where the derivative is zero.



Global min: find a global minimizer, a value of x for which $f(x)$ is minimized

local min: point $x^* \Rightarrow$

there exist a $\delta > 0$

such that $f(x^*) \leq f(x)$

for all x with $|x - x^*| < \delta$

multivariable $\|x - x^*\| < \delta$

Univariant:

Conditions for a strong local min:

1. $f'(x^*) = 0$ local derivative = zero
2. $f''(x^*) > 0$ Second derivative = positive

Also (local min) first order

1. $f'(x^*) = 0$ necessary condition (FONC)
2. $f''(x^*) \geq 0$ Second order (SONC)

to show 1) Taylor Series Exp.

$$f(x^*+h) = f(x^*) + hf'(x^*) + o(h^2)$$
$$f(x^*-h) = f(x^*) - hf'(x^*) + o(h^2)$$

$$\left.\begin{array}{l} f(x^*+h) \geq f(x^*) \rightarrow hf'(x^*) \geq 0 \\ f(x^*-h) \geq f(x^*) \rightarrow hf'(x^*) \leq 0 \end{array}\right\} \Rightarrow f'(x^*) = 0$$

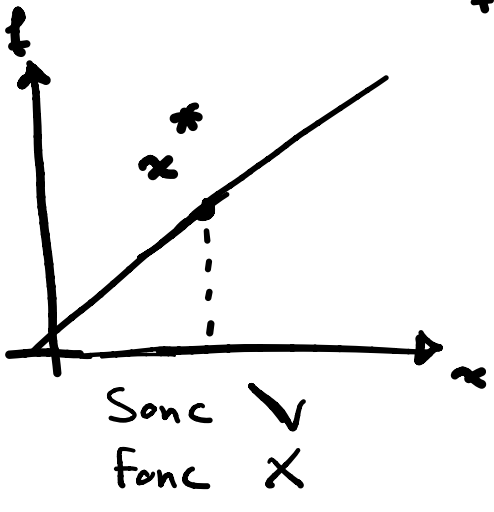
2) Taylor Series Exp.

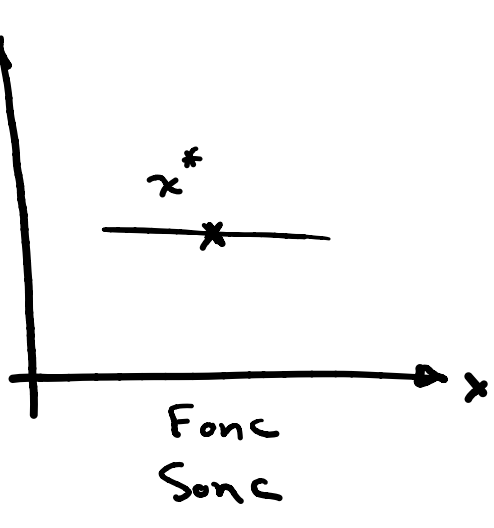
$$f(x^*+h) = f(x^*) + \cancel{hf'(x^*)}^{0} + \frac{h^2}{2}f''(x^*) + o(h^3)$$

$$\Downarrow$$

$$f''(x^*) \geq 0$$

Ex.





Multivariant:

1. $\nabla f(x) = 0$ (FONN)
2. $\nabla^2 f(x)$ Possitive semi definite (SONC)

$$\Rightarrow f(x^* + hy) = f(x^*) + h \cancel{\nabla f(x^*)^T y}^{\,0} + \tfrac{1}{2} h^2 y^T \nabla^2 f(x^*) y + o(h^3)$$

$$\boxed{y^T \nabla^2 f(x^*) y \geq 0}$$

↳ eigenvalues $\lambda_j \geq 0$

Ex. Rosenbrock banana function

$$f(x) = (1 - x_1)^2 + 5(x_2 - x_1^2)^2$$

(a) $(1,1)$ does this satisfy SONN & FONC

$$\nabla f = \begin{pmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\nabla^2 f = \begin{bmatrix} \partial^2 f / \partial x_1 \partial x_1 & \partial^2 f / \partial x_1 \partial x_2 \\ \partial^2 f / \partial x_2 \partial x_1 & \partial^2 f / \partial x_2 \partial x_2 \end{bmatrix} = \begin{bmatrix} 42 & -20 \\ -20 & 10 \end{bmatrix}$$

- Gradient-based

- Derivative-free methods
  - * Population methods
  - * Stochastic methods

- Surrogate space optimisation

$f(x) \longrightarrow \tilde{f}(x)$

Derivatives & Gradients.

Numerical differentiation:

* finite difference Method:

$O(1)$ $f'(x) = \frac{f(x+h) - f(x)}{h}$ forward diff.

$O(2)$ $f'(x) = \frac{f(x+h/2) - f(x-h/2)}{h}$ Central diff

$O(1)$ $f'(x) = \frac{f(x) - f(x-h)}{h}$ backward diff

* Complex Step Method:

bypass the effect of subtractive cancelation

→ Single function evaluation (step in Imag. dir)

$$f(x+ih) = f(x) + ihf'(x) - h^2\frac{f''(x)}{2!} - ih^3\frac{f'''(x)}{3!} + \dots$$

$$\mathrm{Im}\left(f(x+ih)\right) = hf'(x) - h^3\frac{f'''(x)}{3!} + \dots$$

$$f'(x) = \frac{\mathrm{Im}\left(f(x+ih)\right)}{h} + \underbrace{h^2\frac{f''(x)}{3!} + \dots}_{O(h^2)}$$

as $h \to 0$

Dual numbers:

using an abstract quantity ε, where ε^2 is defined as 0.

like complex numbers, a dual number is written as $a + \varepsilon b$

if $\begin{matrix} a \\ b \end{matrix} \Rightarrow$ real values

$(a+\varepsilon b) + (c + \varepsilon d) = (a+c) + \varepsilon(b+d)$

$(a+\varepsilon b) \times (c + \varepsilon d) = ac + \varepsilon(ad+bc)$

- by passing a dual number in any smooth function f, get $f(x)$ $f'(x)$

* Automatic differentiation :

→ Numerical evaluation of derivative through algorithms

⇒ key : Application of chain rule

⇒ Program : Composed of opperation lik addition, substration, multip. & division

$$f(a,b) = \ln(ab + \max(a,2))$$

to compute ; partial derivative w.r.t a @ point

$$\frac{\partial f}{\partial a} = \frac{\partial}{\partial a} \ln(ab + \max(a,2))$$

$$= \frac{1}{ab + \max(a,2)} \frac{\partial}{\partial a}(ab + \max(a,2))$$

$$= \sim \left(\frac{\partial(ab)}{\partial a} + \frac{\partial}{\partial a}(\max(a,2)) \right)$$

$$= \frac{1}{ab + \max(a,2)} \left[b + (2 < a) \right]$$

→ Automated ⇒ use complitational graph

Computational graph → A function where the nodes are operations and eadges are input-output relation

$(\partial f/\partial a)$

$\dot{b}=0$

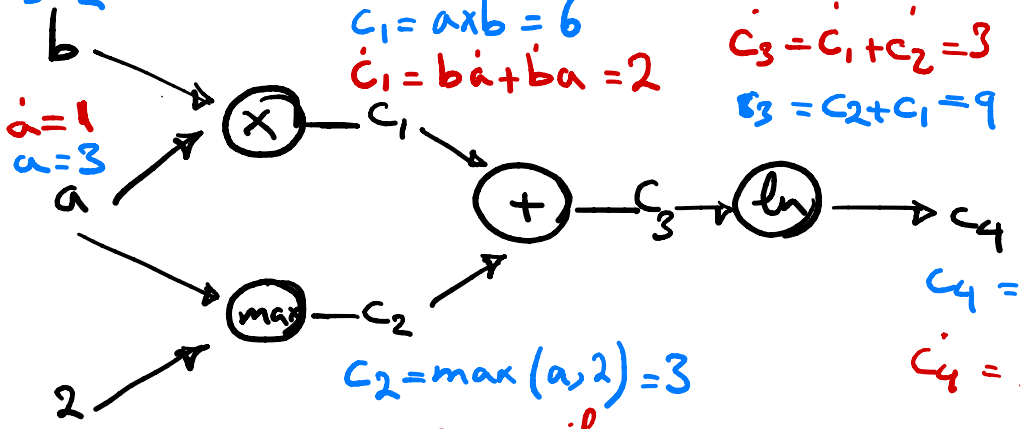
$b=2$

$\dot{a}=1$

$a=3$

$f(3,2)=\ln 9$

$\partial f/\partial a\,(3,2)=1/3$ → forward accumulation



$c_1=a\times b=6$

$\dot{c}_1=\dot{b}a+b\dot{a}=2$

$c_2=\max(a,2)=3$

$\dot{c}_2=\begin{cases}0 & \text{if } 2>a\\ \dot{a} & \text{if } 2<a\end{cases}=1$

$c_3=c_1+c_2=9$

$\dot{c}_3=\dot{c}_1+\dot{c}_2=3$

$c_4=\ln c_3=\ln 9$

$\dot{c}_4=\dfrac{\dot{c}_3}{c_3}=1/3$

Chain-rule:

$$\frac{df}{dx}=\frac{\partial f}{\partial c_4}\frac{dc_4}{dx}=\frac{\partial f}{\partial c_4}\left(\frac{\partial c_4}{\partial c_3}\frac{dc_3}{\partial x}\right)$$

$$=\frac{df}{\partial c_4}\left(\frac{\partial c_4}{\partial c_3}\left(\frac{\partial c_3}{\partial c_2}\frac{\partial c_2}{\partial x}+\frac{\partial c_3}{\partial c_1}\frac{\partial c_1}{\partial x}\right)\right)$$

(a) $a=3$, $b=2$

1. the value of the function and partial derivative at each node
2. Proceed the tree (one node at a time)

Gradient-based Optimisation algorithms

First Order Methods:

Rely on gradients information to help direct the search for a minimum.

* Gradient Descent

for descent direction "d" → direction of steepest descent

→ Progress guaranteed (if) * ∇f smooth

* Stepsize sufficiently small

→ direction d descent is $-\nabla f$ → gradient

here: $g^{(k)} = \nabla f(x^{(k)})$ — iteration k; design point

Usually normalized:

$$d^{(k)} = -\frac{g^{(k)}}{\|g^{(k)}\|}$$

Ex. $f(x_1, x_2) = x_1 x_2^2$

$$\nabla f_{(x_1, x_2)} = \left[x_2^2,\ 2x_1x_2\right]$$

for $x^{(k)} = [1, 2]$

gradient: $g = [-4, -4]$

direction "d": $d = \left[-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right]$

Direction ↘

size ?

* Optimise the step-size at each step

$$\alpha^{(k)} = \underset{\alpha}{\operatorname{argmin}} \left(f(x^{(k)} + \alpha d^{(k)}) \right)$$

---

line search

Ex. Consider conducting a linesearch on $f(x_1, x_2, x_3) = \sin(x_1 x_2) + \exp(x_2 + x_3) - x_3$

from $x = [1, 2, 3]$ in the direction

$$d = [0, -1, -1]$$

resulting optimisation problem, line search:

$$\underset{\alpha}{\text{minimize}} \left((\sin(1 + 0\alpha)(2 - \alpha) + \exp((2-\alpha) + (3-\alpha)) - (3-\alpha) \right)$$

$$\underset{\alpha}{\text{minimize}} \left(\sin(2-\alpha) + \exp(5 - 2\alpha) + \alpha - 3 \right)$$

$\alpha_{min} = 3.127$ with $x \approx [1, -1.126, -0.126]$

One can show that using line-search each direction is orthogonal to the next search direction

$$\alpha^{(k)} = \underset{\alpha}{\operatorname{argmin}}\, f\left(x^{(k)} + \alpha d^{(k)}\right)$$

$\rightarrow$ directional derivative equals zero

using,

$$\nabla_s f(x) = \nabla f(x)^T s$$

we have:

$$\nabla f\left(x^{(k)} + \alpha d^{(k)}\right)^T d^{(k)} = 0$$

therefore

$$d^{(k+1)} = -\frac{\nabla f(x^k + \alpha d^k)}{\|\nabla f(x^k + \alpha d^k)\|}$$

$$d^{k+1\,T} d^k = 0 \Longrightarrow \text{Orthogonal}$$

Vally's floor $\rightarrow$ lots of iterations!

Conjugate Gradient:

Overcomes this issue by borrowing inspiration from methods for optimizing quadratic functions;

$$\underset{x}{\text{minimize}}\ f(x) = \tfrac{1}{2} x^T A x + b^T x + c$$

A symmetric positive definite $\Rightarrow$ f has a unique local minima

→ Can optimise n-dimention quadratic functions in n-steps. The directions are mutually conjugate with respect to A:

$$d^{(i)T} A\, d^{(j)} = 0 \quad \text{for all } i \neq j$$

$d^{(i)}$ — $d^{(j)}$ → basis vectors of A.

The algorithm.

1. Start with the direction of steepest descent

$$d^{(1)} = -g^{(1)}$$

2. Use line-search for the next design point.

⇒ for quadratic functions α can be computed exactly

$$x^{(2)} = x^{(1)} + \alpha^{(1)} d^{(1)}$$

3. for subsequent iteration choose $d^{(k+1)}$ based on the next gradient and contribution from the current descent

$$d^{k+1} = -g^{(k+1)} + \beta^{(k)} d^{(k)}$$

$\beta \uparrow$ → previous descent direction contributes more

Ex: line-search for quadratic functions:

derive the optimal step factor for a line search on a quadratic function:

$$\underset{\alpha}{\text{minimise}} \quad f(x+\alpha d)$$

compute the derivative w.r.t α

$$\frac{\partial f(x+\alpha d)}{\partial \alpha} = \frac{\partial}{\partial \alpha}\left[\frac{1}{2}(x+\alpha d)^T A (x+\alpha d) + b^T(x+\alpha d) + c \right]$$

$$= d^T A (x+\alpha d) + d^T b$$

$$= d^T A x + b + \alpha d^T A d$$

Setting $\frac{\partial f(x+\alpha d)}{\partial \alpha} = 0$

$$\boxed{\alpha = -\frac{d^T(Ax+b)}{d^T A d}}$$

A is known

best value for β using, $d^{(k+1)}$ is conjugate to $d^{(k)}$

$$d^{(k+1)T} A d^{(k)} = 0$$

$$\Rightarrow \left(-g^{(k+1)} + \beta^{(k)} d^{(k)}\right)^T A d^{(k)} = 0$$

$$\Rightarrow -g^{(k+1)T} A d^{(k)} + \beta^{(k)} d^{(k)T} A d^{(k)} = 0$$

$$\Rightarrow \boxed{\beta^{(k)} = \frac{g^{(k+1)T} A d^{(k)}}{d^{(k)T} A d^{(k)}}}$$

Unfortunately we don't have access to A $\Rightarrow$

* Fletcher-Reeves: $\beta^{(k)} = \frac{g^{T(k)} g^{(k)}}{g^{T(k-1)} g^{(k-1)}}$

* Polak_Ribière :

$$\beta^{(k)} = \frac{g^{T(k)}\left(g^{(k)} - g^{(k-1)}\right)}{g^{T(k-1)} g^{(k-1)}}$$

Convergence for PR method can be guaranteed if we modify

$$\beta \to \max(\beta, 0)$$

automatic reset