Session 3: 03/02/2021

take an example close to fluid flow applications:

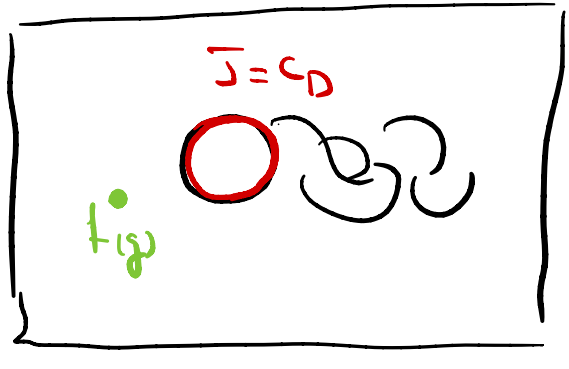
$$\begin{cases} \text{minimize} & J(\varphi, g) \\ \text{subject to} & F(\varphi, g) = 0 \end{cases}$$

φ: state variables (u, v, p, T, ρ)

g: control variables / design parameters

$F(\varphi, g) = 0$ → NS eqn.

In many applications objective func. does not depend on control variable



How to include g in the opt. problem:

(1) Add a constraint on g: limit the size of admissible control

$$\|g\| < k$$

$$\begin{cases} \text{minimize} & \mathcal{E}(\varphi) \\ \text{s.t} & F(\varphi, g) = 0 \\ & \|g\| < k \end{cases}$$

(2) Penalize the objective function

instead of minimizing $\mathcal{E}(\varphi)$

minimize $J(\varphi, g) = \mathcal{E}(\varphi) - \alpha \|g\|^{\beta}$

Constants

- In (1) one is imposing an additional constraint

- In (2) one has changed the problem

  → usually the prefered strategy

One Shot Method:

Define Lagrange multipliers and get the optimality condition:

→ introduce { Lagrange multipliers / adjoint variables / co-state variables } (ξ)

$$\mathcal{L}(\varphi, q, \xi) = J(\varphi, q) - \xi^{*} F(\varphi, q)$$

$\xi^{*} F$: inner product

→ find φ, g, ξ such that

$$\partial \mathcal{L}/\partial(\varphi, g, \xi) = 0$$

⇒ Optimality System:

① $\delta \mathcal{L}/\delta \xi \geq 0$ → Constraint / state eqn.

② $\delta \mathcal{L}/\delta \varphi = 0$ → Adjoint / co-state eqn.

③ $\delta \mathcal{L}/\delta g = 0$ → Optimality condition

Setting ① $= 0$ ↑ arbitrary

$$\lim \left(\frac{\mathcal{L}(\varphi, g, \xi + \varepsilon\tilde{\xi}) - \mathcal{L}(\varphi, g, \xi)}{\varepsilon} \right) = 0$$

$$\lim \frac{1}{\varepsilon}\Big(\bar{J}(\phi,g) - (\xi+\varepsilon\tilde{\xi})^* F(\phi,g) - \bar{J}(\phi,g) - \xi^* F(\phi,g) \Big)$$

$$\Rightarrow \tilde{\xi}^* . F(\phi,g) = 0 \Rightarrow F(\phi,g) = 0$$

(2) Variation of $\mathcal{L}$ w.r.t ϕ

$$\lim \left(\frac{\mathcal{L}(\phi+\varepsilon\tilde{\phi}, g, \xi) - \mathcal{L}(\phi,g,\xi)}{\varepsilon} \right) = 0$$

$$\lim_{\varepsilon\to 0} \frac{1}{\varepsilon}\Big(\big(\bar{J}(\phi+\varepsilon\tilde{\phi}, g) - \xi^* F(\phi+\varepsilon\tilde{\phi}, g)\big) - \big(\bar{J}(\phi,g) - \xi^* F(\phi,g)\big) \Big)$$

$$\lim_{\varepsilon\to 0} \left[\frac{\bar{J}(\phi+\varepsilon\tilde{\phi}, g) - \bar{J}(\phi,g)}{\varepsilon} - \xi^* \frac{\left(F(\phi+\varepsilon\tilde{\phi}, g) - F(\phi,g)\right)}{\varepsilon} \right]$$

$= 0$

Introducing Taylor series:

$$\bar{J}(\varphi+\varepsilon\tilde{\varphi}, g) = \bar{J}(\varphi, g) + \varepsilon\left(\frac{\partial \bar{J}}{\partial \varphi}\bigg|_{(\varphi,g)}\right)\tilde{\varphi} + \cancel{\mathcal{O}(\varepsilon^2)}$$

$$\lim_{\varepsilon \to 0}\left(\left(\frac{\partial \bar{J}}{\partial \varphi}\bigg|_{(\varphi,g)}\right)\tilde{\varphi} - \xi^*\left(\frac{\partial F}{\partial \varphi}\bigg|_{(\varphi,g)}\tilde{\varphi}\right)\right) = 0$$

$$\left(\left(\frac{\partial \bar{J}}{\partial \varphi}\right)\tilde{\varphi} - \xi^*\left(\frac{\partial F}{\partial \varphi}\right)\tilde{\varphi}\right) = 0$$

$$\equiv \bar{\varphi}^*\left(\left(\frac{\partial \bar{J}}{\partial \varphi}\right)^* - \left(\frac{\partial F}{\partial \varphi}\right)^*\xi\right) = 0$$

$$\Rightarrow \boxed{\left(\frac{\partial \bar{J}}{\partial \varphi}\right)^* - \left(\frac{\partial F}{\partial \varphi}\right)^*\xi = 0}$$

Adj eqn. always linear!

(3)

$$\lim_{\varepsilon} \left(\frac{\mathcal{L}(\varphi, g + \varepsilon \tilde{g}, \xi) - \mathcal{L}(\varphi, g, \xi)}{\varepsilon} \right) = \cdots$$

$$\boxed{\left(\frac{\partial F}{\partial g}\bigg|_{\varphi,g} \right)^* \xi = \left(\frac{\partial \bar{J}}{\partial g}\bigg|_{\varphi,g} \right)^*}$$

Optimality Cond.

Optimality System:

State eqn: $F(\varphi, g) = 0$

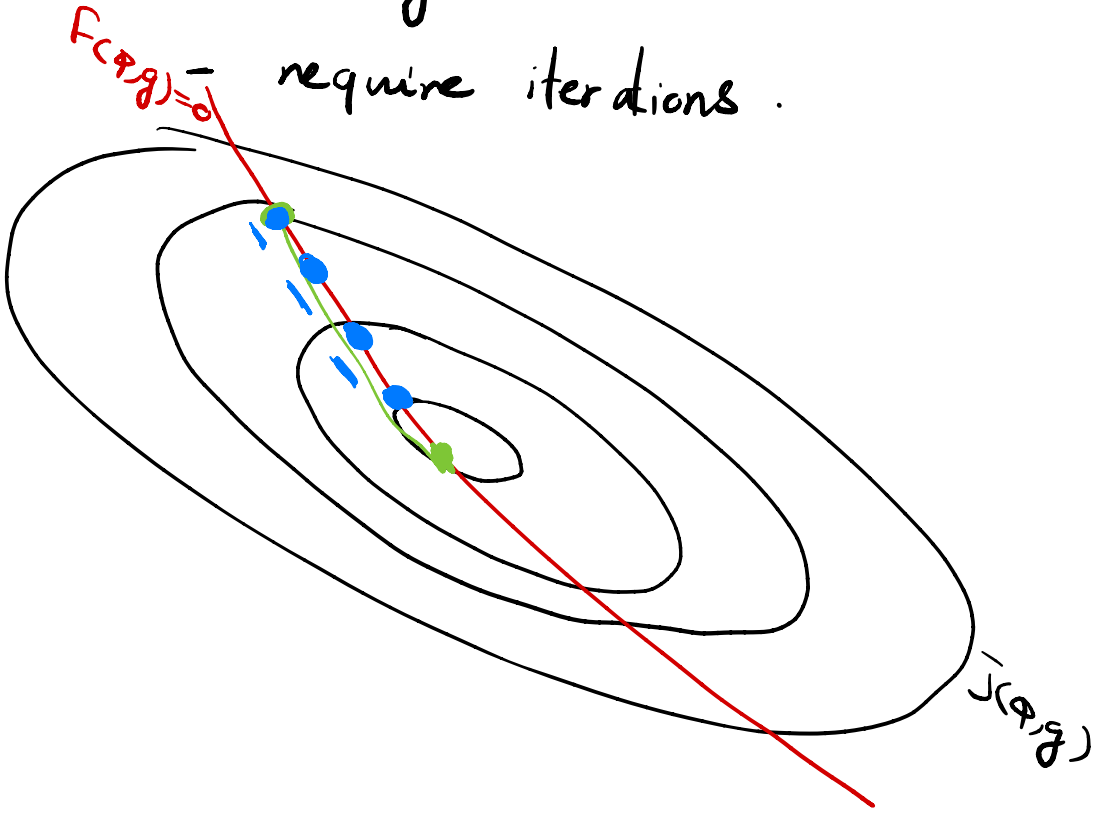
adjoint eqn: $\left(\frac{\partial f}{\partial \varphi}\bigg|_{\varphi,g} \right)^* \xi = \left(\frac{\partial \bar{J}}{\partial \varphi}\bigg|_{\varphi,g} \right)^*$

Optimality Cond: $\left(\frac{\partial F}{\partial g}\bigg|_{\varphi,g} \right)^* \xi = \left(\frac{\partial \bar{J}}{\partial g}\bigg|_{\varphi,g} \right)^*$

Solving the optimality system in one shot is not feasible

$\Rightarrow$ - Approximate the gradient locally

- require iterations.



Algorithm:

- Start with an initial guess

$g^{(0)}$ (design parameters)

for $m = 0, 1, 2 \longrightarrow$ until convergence

1. Solve $F(\varphi^m, g^m) = 0 \longrightarrow$ corresponding state variable

$$\varphi^m = \varphi(g^m)$$

2. Compute the gradient of the functional

$$\boxed{\frac{D\bar{J}}{Dg}(\varphi^m, g^m)}$$

3. use 1, 2 to compute $\delta g^{(m)}$

4. Set $g^{m+1} = g^m + \delta g^m$

for step 3 use your favorite optimisation method

- gradient descent
- CG
- QN

- **Computing the gradient using adjoints ; for (2)**

→ Let $\lambda^{(m)}$ be the soln of the adj eqn.

$$\left(\frac{\partial F}{\partial q}\Big|_{q^m}\right)^* \lambda^m = \left(\frac{\partial J}{\partial q}\Big|_{q^m}\right)^*$$

driven equation ← | dependent on q^m, φ^m

* the eqn. is linear to the adjoint variable

[*] if one has multiple design variables, still a single eqn for adj. is solved !

Recall

$$\left.\frac{Dj}{Dg}\right|_{g^{(m)}} = \left.\frac{\partial j}{\partial p}\right|_{g^m} \left.\frac{\partial p}{\partial g}\right|_{g^m} + \left.\frac{\partial \bar{J}}{\partial g}\right|_{g^m}$$

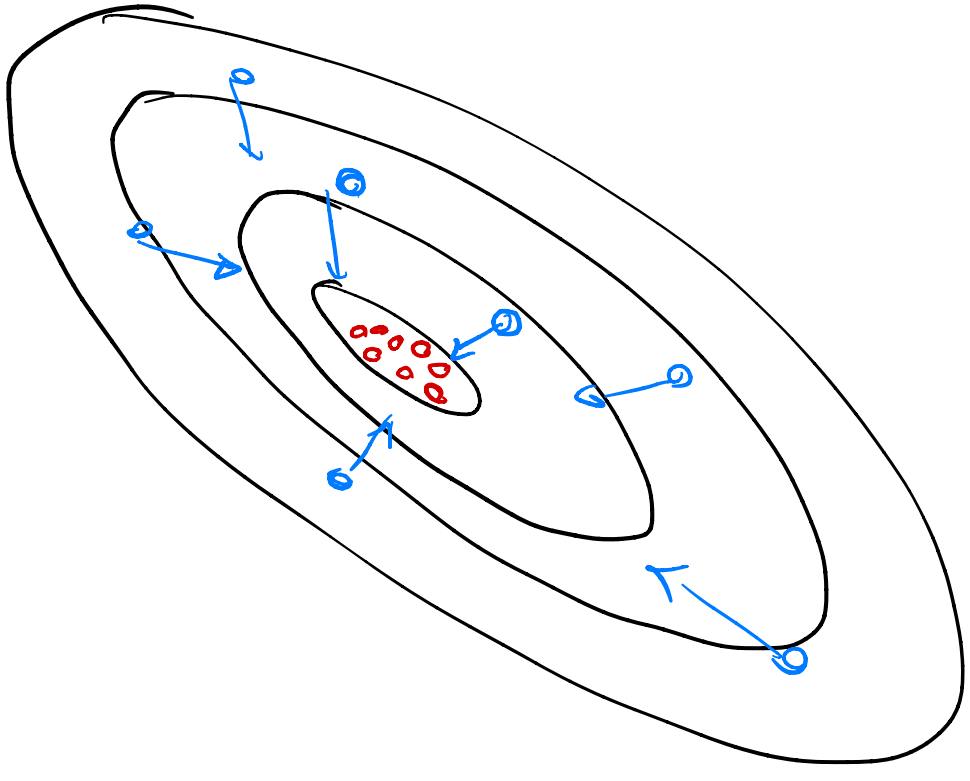
using adj eqn: $\quad -\left.\frac{\partial F}{\partial g}\right|_{g^{(m)}}$

$$\left.\frac{DJ}{Dg}\right|_{g^m} = (\lambda^m)^* \left(\frac{\partial F}{\partial p}\right) \left.\frac{\partial p}{\partial g}\right|_{g^m} + \left.\frac{\partial \bar{J}}{\partial g}\right|_{g^m}$$

$$\left.\frac{Dj}{Dg}\right|_{g^m} = \boxed{-\left(\lambda^m\right)^* \left.\frac{\partial F}{\partial g}\right|_{g^m} + \left.\frac{\partial J}{\partial g}\right|_{g^m}}$$

- An expression of the gradient of the functional in terms of the Soln. of the adj eqn -

Sampling Plans:



* Full Factorial:

Grid of evenly spaced points

→ easy to implement

→ does not rely on randomness

→ requires large number of points

* Random sampling:

- Straight forward
- random - long-uniform distribution

* Uniform projection Plans:

e.g. 2D optimisation problem

discretised into an mxm sampling grid (Full Factorial), instead of keeping all m^2 samples, we want only m

- sample spreads across the space
- " " individual components

- A uniform projection plan is a sampling plan over a discrete grid where the distribution over each dimension is uniform.

→ with n samples nxn grid can be constructed using an n-element permutation.

→ $n!_c$ possible uniform projection plans!

→ sampling with uniform projection plan is sometimes called "latin-hypercube-sampling" → latin squares

Population Methods:

* Initialisation:
  - Start with an initial population
  - Sampling plans.

* Genetic Algorithms!
  → biological evolution
  → fitter individuals more likely to pass on their Genes.
  → fitness is inversely related to the value of the objective function
  → design point associated with an individual → chromosome

→ Chromosomes of fitter individuals are passed to the next generation

- crossover
- mutation

Chromosomes:

real valued vectors in $\mathbb{R}^d$ corresponding to points in design-space

Selection:

The process of choosing chromosoms to be used as parents for next generation

→ Population with "m" chromosoms → list of m parents pairs → m children of next generation

→ Biasing the selection toward the fittest

1. truncation selection
2. tournament selection
3. roulette wheel selection

→ Minimize the objective function:

⇒ fitness is inversely related to

$$y^{(i)} = f(x^{(i)})$$

↓ individual

Crossover:

interpolates between real values of design points → additional Crossover

$$x = (1-\lambda)\, x_a + \lambda x_b \quad (\lambda = 1/2)$$

→ linear interpolation

Mutation :

if we only crossover

→ traits not present in initial population ⟹ never occure

→ Most fit saturate the population

→ mutation → new trails

→ Add zero-mean Gaussian noise

Particle Swarm Optimisation :

- uses Swarm intelligence to find the global minimum.
- Swarm intelligence is composed of agents comunicating with other agents

- exchange info about local & global environment

- Imitating human (insects) social behavior

- Similar as before, everything is evaluated based on the fitness function

- The computation of the particle velocity then has three components

(1) inertial component

(2) Cognitive component

(3) Social component

1 inertial Component

maintains a fraction of the current velocity vector, by continuing along the same direction as during previous step

2 Cognitive component:

- The difference between the current position and the particle's best position

  → local memory effect

3 Social Component:

Proportional to the difference of the current agent position and the current best position of the entire swarm

The final expression for the global velocity update:

$$\underline{x}^{(i)} \leftarrow \underline{x}^{(i)} + \underline{v}^{(i)}$$

$$v^{(i)} \leftarrow \omega v^{(i)} + c_1 r_1 \left(x^{(i)}_{best} - x^{(i)}\right) + c_2 r_2 \left(x_{best} - x^{(i)}\right)$$

w, c_1, c_2 : parameters

r_1, r_2 : random numbers
drawn from $\sim U(0,1)$