Multidisciplinary System Design Optimization (MSDO)

Sensitivity Analysis

Lecture 8

1 March 2004

Olivier de Weck

Karen Willcox

Today's Topics

- Sensitivity Analysis
  - effect of changing design variables
  - effect of changing parameters
  - effect of changing constraints
- Gradient calculation methods
  - Analytical and Symbolic
  - Finite difference
  - Adjoint methods
  - Automatic differentiation

Standard Problem Definition

$$
\begin{aligned}
\min \quad & J(\mathbf{x}) \\
\text{s.t.} \quad & g_j(\mathbf{x}) \le 0 \quad j = 1,..,m_1 \\
& h_k(\mathbf{x}) = 0 \quad k = 1,..,m_2 \\
& x_i^{\ell} \le x_i \le x_i^{u} \quad i = 1,..,n
\end{aligned}
$$

For now, we consider a single objective function, $J(\mathbf{x})$.

There are n design variables, and a total of m constraints ($m=m_1+m_2$).

The bounds are known as **side constraints**.

Sensitivity Analysis

- Sensitivity analysis is a key capability aside from the optimization algorithms we discussed.
- Sensitivity analysis is key to understanding which design variables, constraints, and parameters are important drivers for the optimum solution **x***.
- The process is NOT finished once a solution x* has been found. A sensitivity analysis is part of post-processing.
- Sensitivity/Gradient information is also needed by:
  - gradient search algorithms
  - isoperformance/goal programming
  - robust design

Sensitivity Analysis

- How sensitive is the "optimal" solution *J** to changes or perturbations of the design variables **x***?

- How sensitive is the "optimal" solution **x*** to changes in the constraints **g(x), h(x)** and fixed parameters **p** ?

Sensitivity Analysis: Aircraft

Questions for aircraft design:

How does my solution change if I

- change the cruise altitude?
- change the cruise speed?
- change the range?
- change material properties?
- relax the constraint on payload?
- ...

Sensitivity Analysis

Questions for spacecraft design:

How does my solution change if I

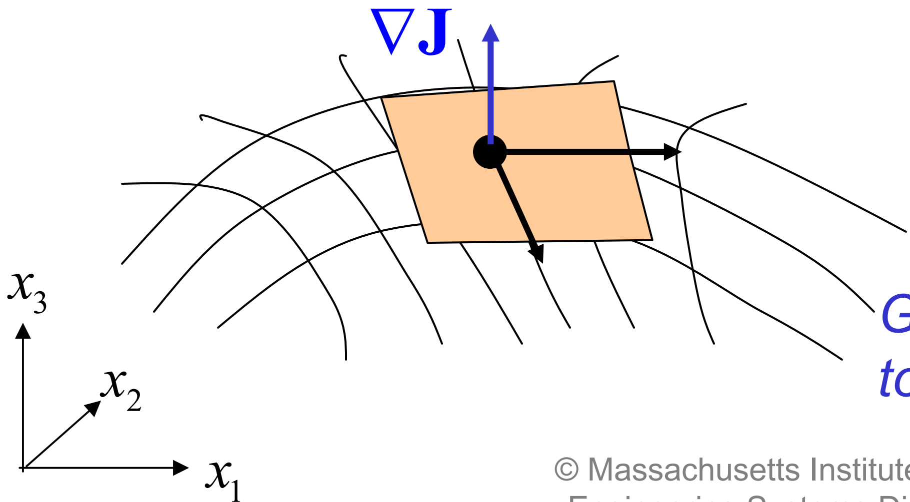
- change the orbital altitude?
- change the transmission frequency?
- change the specific impulse of the propellant?
- change launch vehicle?
- Change desired mission lifetime?
- ...

Gradient Vector – single objective

"How does the objective function J value change as we change elements of the design vector **x**?"

Compute partial derivatives of J with respect to x_i $\quad \dfrac{\partial J}{\partial x_i}$

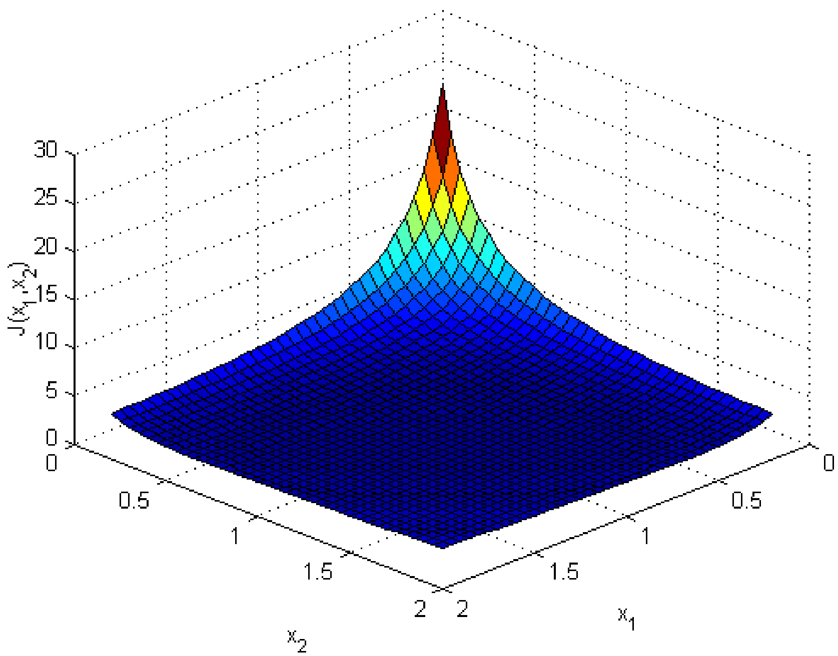
$$\nabla \mathbf{J} = \begin{bmatrix} \dfrac{\partial J}{\partial x_1} \\ \dfrac{\partial J}{\partial x_2} \\ \vdots \\ \dfrac{\partial J}{\partial x_n} \end{bmatrix}$$

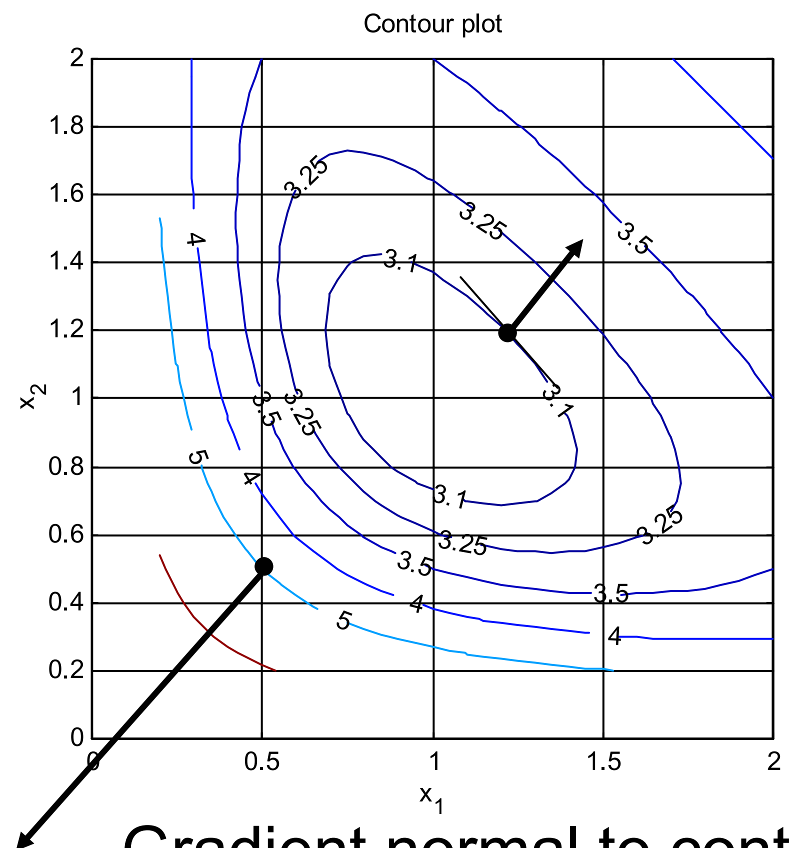


Gradient vector points normal to the tangent hyperplane of J(x)

Geometry of Gradient vector (2D)

Example function: $$J(x_1, x_2) = x_1 + x_2 + \frac{1}{x_1 \cdot x_2}$$





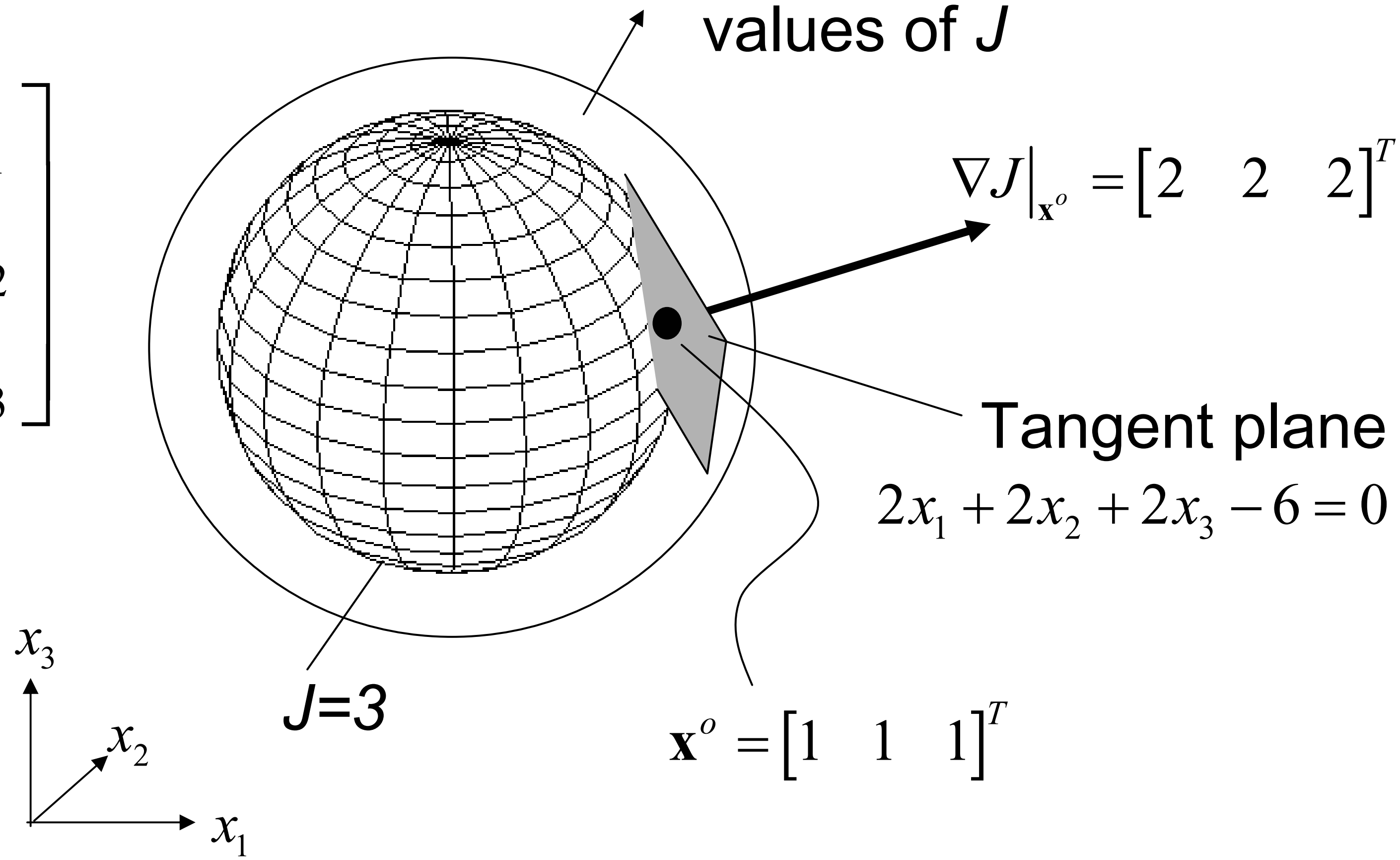
$$\nabla J = \begin{bmatrix} \dfrac{\partial J}{\partial x_1} \\ \dfrac{\partial J}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 - \dfrac{1}{x_1^2 x_2} \\ 1 - \dfrac{1}{x_1 x_2^2} \end{bmatrix}$$

Gradient normal to contours

Geometry of Gradient vector (3D)

Example $J = x_1^2 + x_2^2 + x_3^2$

$$\nabla J = \begin{bmatrix} 2x_1 \\ 2x_2 \\ 2x_3 \end{bmatrix}$$



Gradient vector points to larger values of *J*

Taylor Series Expansion

$$\mathbf{x}^k \mapsto J\left(\mathbf{x}^k\right),\ where\ \ \mathbb{R}^n \mapsto \mathbb{R}$$

Taylor Series Expansion of Objective Function

$$J(\mathbf{x}) = J(\mathbf{x}^0) + \underbrace{\left[\nabla J(\mathbf{x}^0)\right]^T (\mathbf{x}-\mathbf{x}^0)}_{\text{first order term}} + \underbrace{\frac{1}{2}(\mathbf{x}-\mathbf{x}^0)^T \mathbf{H}(\mathbf{x}^0)(\mathbf{x}-\mathbf{x}^0)}_{\text{second order term}} + \text{H.O.T.}$$

Tangential hyperplane at $\mathbf{x^o}$

Effect of curvature (2nd derivative) at $\mathbf{x^o}$

Jacobian Matrix – multiple objectives

If there is more than one objective function, i.e.
if we have a gradient vector for each J_i, arrange them columnwise and get Jacobian matrix:

$$\mathbf{J} = \underbrace{\begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_z \end{bmatrix}}_{z \times 1} \quad \Longrightarrow \quad \nabla \mathbf{J} = \underbrace{\begin{bmatrix} \frac{\partial J_1}{\partial x_1} & \frac{\partial J_2}{\partial x_1} & \cdots & \frac{\partial J_z}{\partial x_1} \\ \frac{\partial J_1}{\partial x_2} & \frac{\partial J_2}{\partial x_2} & \cdots & \frac{\partial J_z}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial J_1}{\partial x_n} & \frac{\partial J_2}{\partial x_n} & \cdots & \frac{\partial J_z}{\partial x_n} \end{bmatrix}}_{n \times z}$$

Normalization

In order to compare sensitivities from different design variables in terms of their *relative* sensitivity it is necessary to normalize:

$$\left.\frac{\partial J}{\partial x_i}\right|_{\mathbf{x}^{\mathbf{o}}}$$

"raw" - unnormalized sensitivity = partial derivative evaluated at point $x_{i,o}$

$$\frac{\Delta J / J}{\Delta x_i / x_i} \simeq \frac{x_{i,o}}{J(\mathbf{x}^{\mathbf{o}})} \cdot \left.\frac{\partial J}{\partial x_i}\right|_{\mathbf{x}^{\mathbf{o}}}$$

Normalized sensitivity captures relative sensitivity

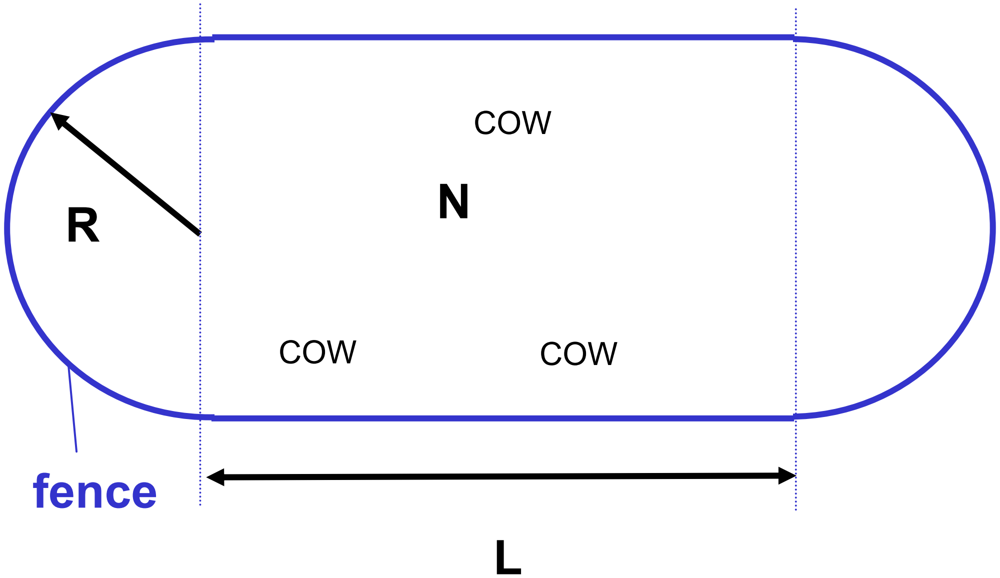
~ % change in objective per % change in design variable

⇨ Important for comparing effect between design variables

Example: Dairy Farm Problem

16.888
ESD.77

"Dairy Farm" sample problem



L – Length = 100 [m]
N - # of cows = 10
R – Radius = 50 [m]

$\Big\} \; x^o$

With respect to which design variable is the objective most sensitive?

Parameters:
f=100$/m
n=2000$/cow
m=2$/liter

$$A = 2LR + \pi R^2$$

$$F = 2L + 2\pi R$$

$$M = 100 \cdot \sqrt{A / N}$$

$$C = f \cdot F + n \cdot N$$

$$I = N \cdot M \cdot m$$

$$P = I - C$$

Assume that we are not at the optimal point **x*** !

Dairy Farm Sensitivity

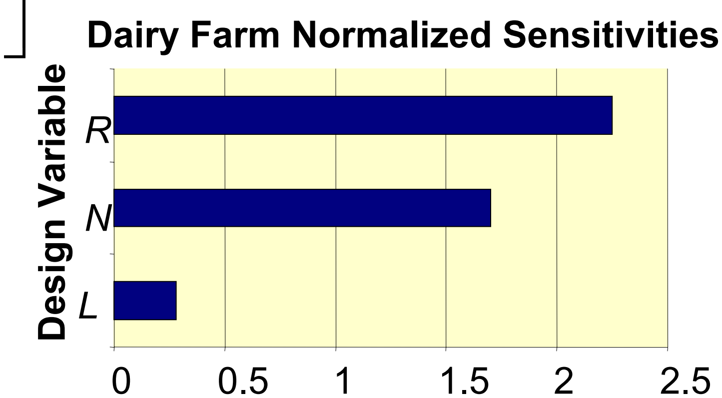
- Compute objective at **xº** $J(\mathbf{x}^o) = 13092$
- Then compute raw sensitivities

$$\nabla J = \begin{bmatrix} \frac{\partial P}{\partial L} \\ \frac{\partial P}{\partial N} \\ \frac{\partial P}{\partial R} \end{bmatrix} = \begin{bmatrix} 36.6 \\ 2225.4 \\ 588.4 \end{bmatrix}$$

- Normalize

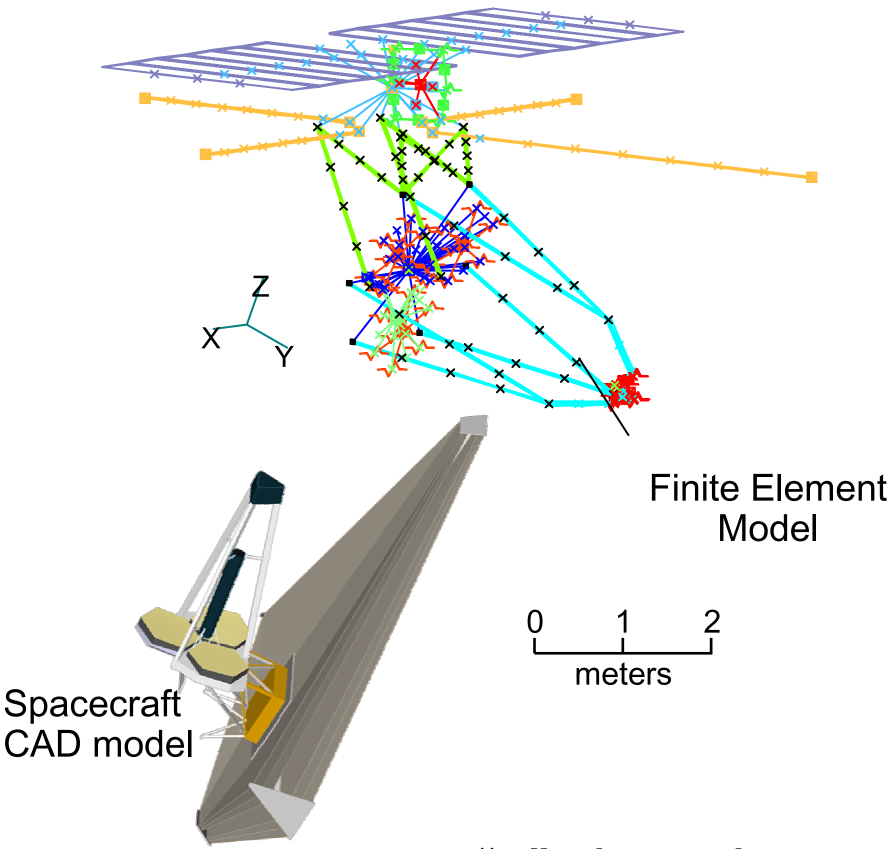
$$\nabla \bar{J} = \frac{\mathbf{x}^o}{J(\mathbf{x}^o)} \nabla J = \begin{bmatrix} \frac{100}{13092} \cdot 36.6 \\ \frac{10}{13092} 2225.4 \\ \frac{50}{13092} 588.4 \end{bmatrix} = \begin{bmatrix} 0.28 \\ 1.7 \\ 2.25 \end{bmatrix}$$

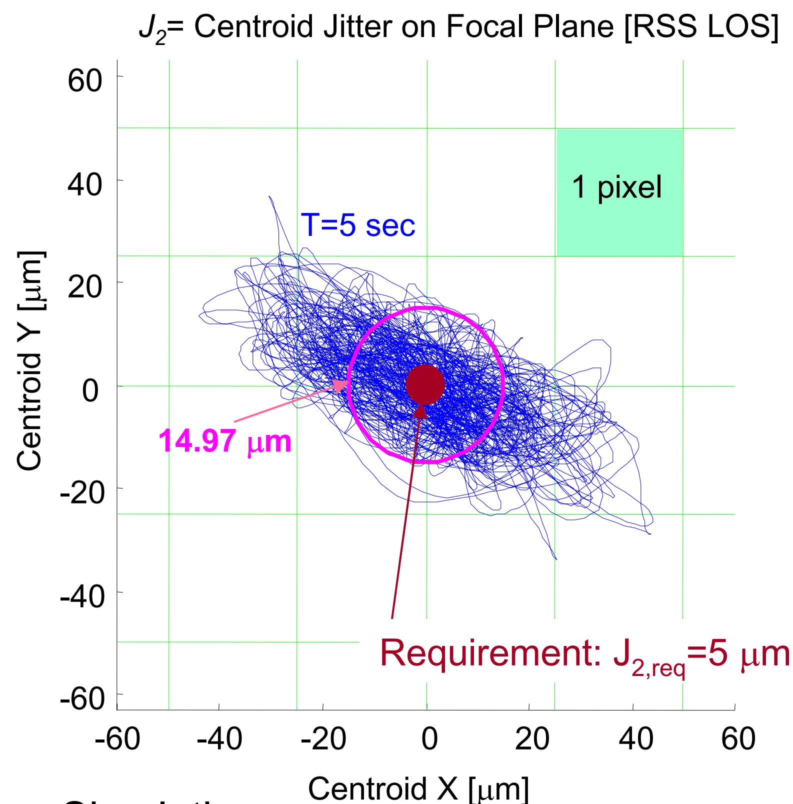
- Show graphically (optional)



Realistic Example: Spacecraft

NASA Nexus Spacecraft Concept



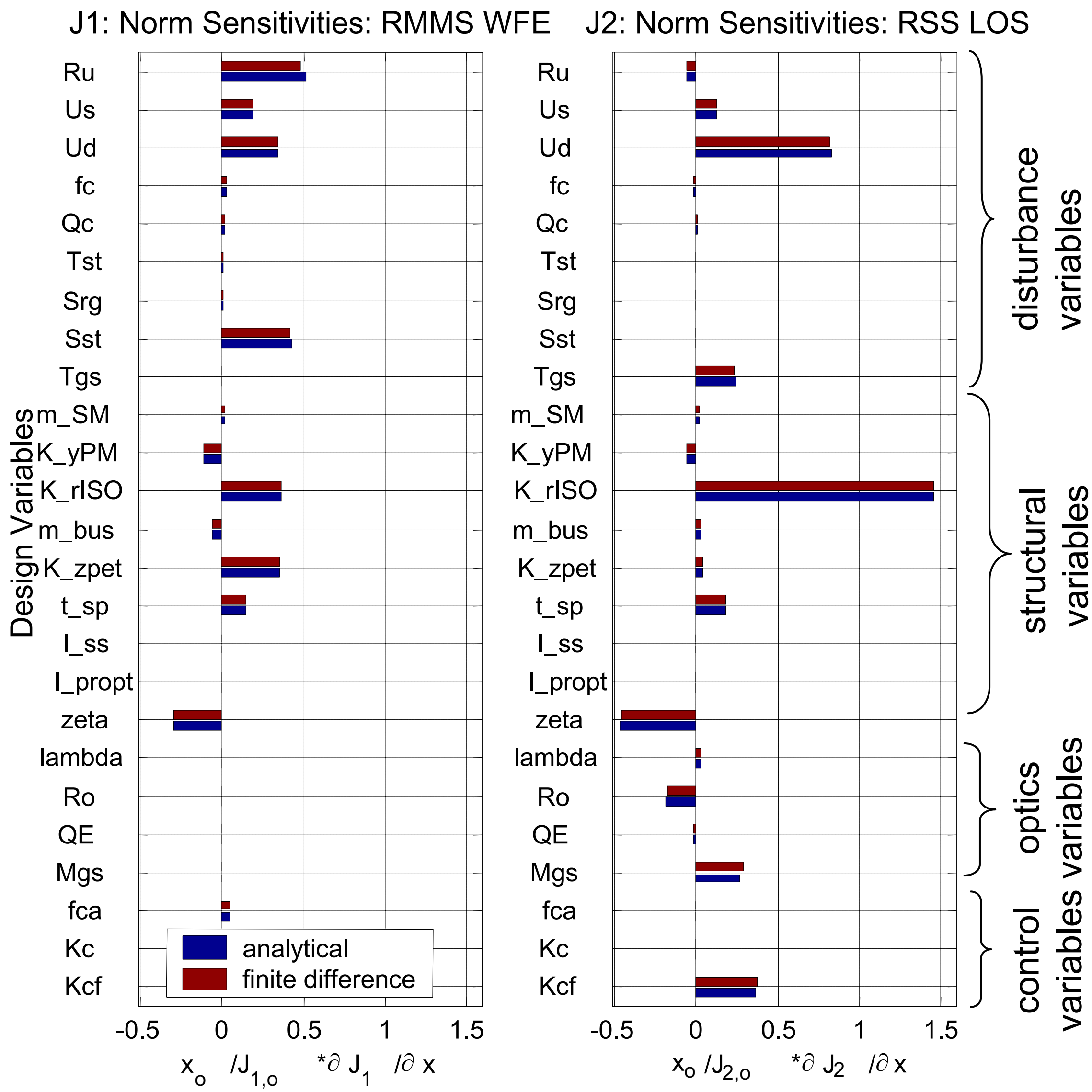


Simulation

"x"-domain → "J"-domain

⇨ What are the design variables that are "drivers" of system performance ?

Graphical Representation



Graphical Representation of Jacobian evaluated at design x^o, normalized for comparison.

$$\bar{\nabla} J = \frac{\mathbf{x}^0}{J_o} \begin{bmatrix} \frac{\partial J_1}{\partial R_u} & \frac{\partial J_2}{\partial R_u} \\ \cdots & \cdots \\ \frac{\partial J_1}{\partial K_{cf}} & \frac{\partial J_2}{\partial K_{cf}} \end{bmatrix}$$

J1: RMMS WFE most sensitive to:

Ru - upper wheel speed limit [RPM]
Sst - star tracker noise 1σ [asec]
K_rISO - isolator joint stiffness [Nm/rad]
K_zpet - deploy petal stiffness [N/m]

J2: RSS LOS most sensitive to:

Ud - dynamic wheel imbalance [gcm^2]
K_rISO - isolator joint stiffness [Nm/rad]
zeta - proportional damping ratio [-]
Mgs - guide star magnitude [mag]
Kcf - FSM controller gain [-]

Analytical Sensitivities

If the objective function is known in closed form, we can often compute the gradient vector(s) in closed form (analytically, symbolically):

Example: $$J(x_1, x_2) = x_1 + x_2 + \frac{1}{x_1 \cdot x_2}$$

Analytical Gradient: $$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \frac{\partial J}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 - \frac{1}{x_1^2 x_2} \\ 1 - \frac{1}{x_1 x_2^2} \end{bmatrix}$$

Example

$x_1 = x_2 = 1$

J(1,1)=3

$$\nabla J(1,1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Minimum

For complex systems analytical gradients are rarely available

Symbolic Differentiation

- Use symbolic mathematics programs
- E.g. Matlab,Maple, Mathematica

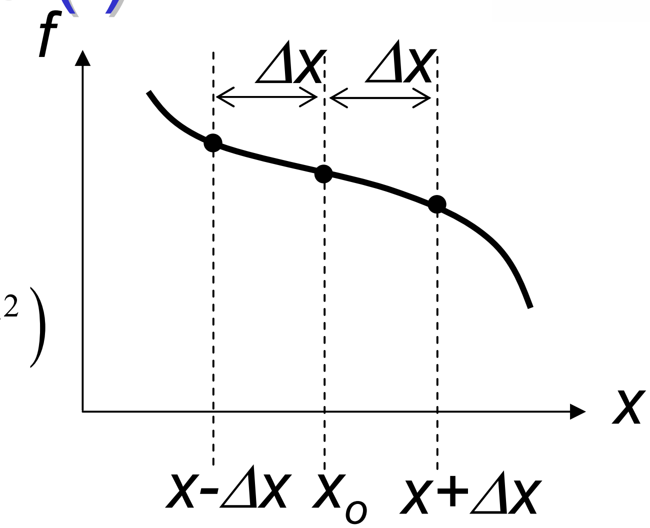
construct a symbolic object

```
EDU» syms x1 x2
EDU» J=x1+x2+1/(x1*x2);
EDU» dJdx1=diff(J,x1)
 dJdx1 =1-1/x1^2/x2
EDU» dJdx2=diff(J,x2)
 dJdx2 = 1-1/x1/x2^2
```

difference operator

Finite Differences (I)

Function of a single variable *f(x)*



$$\Delta x > 0, \Delta x \in \mathbb{R}$$

Taylor Series expansion

$$f(x_o + \Delta x) = f(x_o) + \Delta x f'(x_o) + \frac{\Delta x^2}{2} f''(x_o) + O(\Delta x^2)$$

Neglect second order and H.O.T.

Solve for gradient vector

$$f'(x_o) = \underbrace{\frac{f(x_o + \Delta x) - f(x_o)}{\Delta x}}_{\text{Forward Difference}} + \underbrace{O(\Delta x)}_{\text{Truncation Error}}$$

Forward Difference

Approximation to the derivative

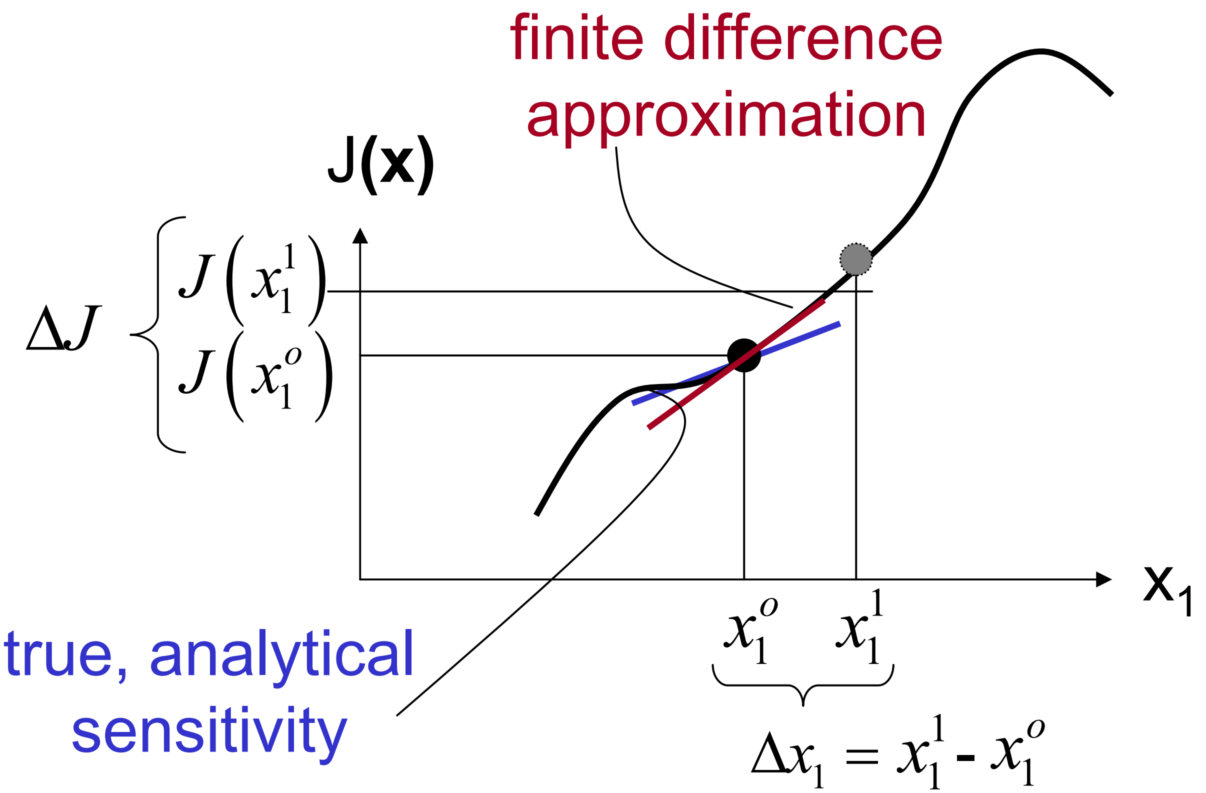
Truncation Error

$$O(\Delta x) = \frac{\Delta x}{2} f''(\zeta)$$

$$x_o \le \zeta \le x_o + \Delta x$$

Finite Differences (II)

$$\frac{\partial J}{\partial x_1} \approx \frac{J\left(x_1^1\right) - J\left(x_1^o\right)}{x_1^1 - x_1^o} = \frac{J\left(x_1^o + \Delta x_1\right) - J\left(x_1^o\right)}{\Delta x_1} = \frac{\Delta J}{\Delta x_1}$$



Finite Differencing (III)

Take Taylor expansion backwards at $x_o - \Delta x$

$$f(x_o + \Delta x) = f(x_o) + \Delta x f'(x_o) + \frac{\Delta x^2}{2} f''(x_o) + O(\Delta x^2) \qquad (1)$$

$$f(x_o - \Delta x) = f(x_o) - \Delta x f'(x_o) + \frac{\Delta x^2}{2} f''(x_o) + O(\Delta x^2) \qquad (2)$$

(1)-(2) and solve again for derivative

$$f'(x_o) = \underbrace{\frac{f(x_o + \Delta x) - f(x_o - \Delta x)}{2\Delta x}}_{\text{Central Difference Approximation to the derivative}} + \underbrace{O(\Delta x^2)}_{\text{Truncation Error}}$$

Truncation Error

$$O(\Delta x^2) = \frac{\Delta x^2}{6} f'''(\zeta)$$

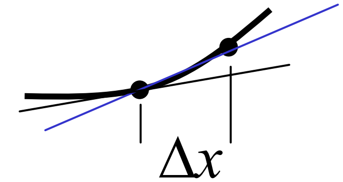
$$x_o \le \zeta \le x_o + \Delta x$$

Central Difference

Approximation to the derivative

Finite Difference Overview

Forward Difference



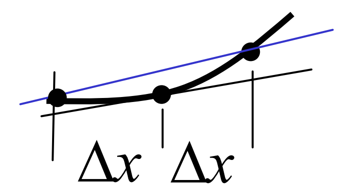
1st derivative

$$f'(x_o) \approx \frac{f\left(x_o + \Delta x\right) - f\left(x_o\right)}{\Delta x}$$

2nd derivative

$$f''(x_o) \approx \frac{f\left(x_o + 2\Delta x\right) - 2f\left(x_o + \Delta x\right) + f\left(x_o\right)}{\Delta x^2}$$

Central Difference



1st derivative

$$f'(x_o) \approx \frac{f\left(x_o + \Delta x\right) - f\left(x_o - \Delta x\right)}{2\Delta x}$$

2nd derivative

$$f''(x_o) \approx \frac{f\left(x_o + \Delta x\right) - 2f\left(x_o\right) + f\left(x_o - \Delta x\right)}{\Delta x^2}$$

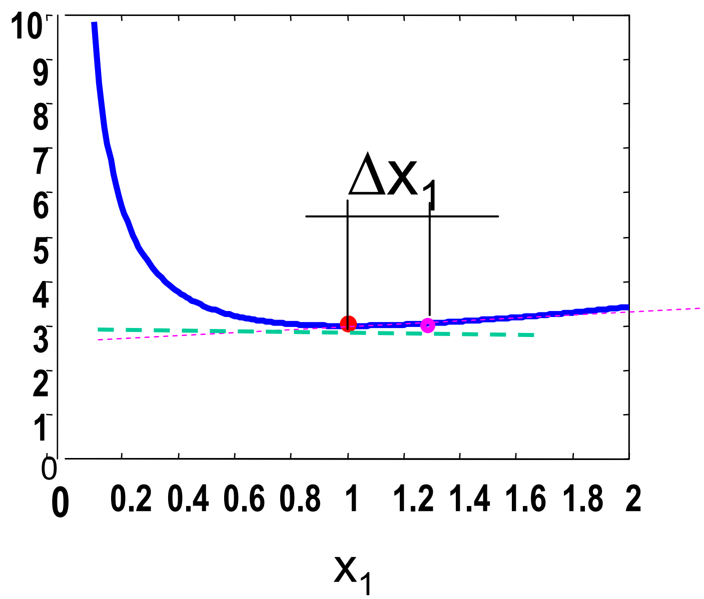
Errors of Finite Differencing

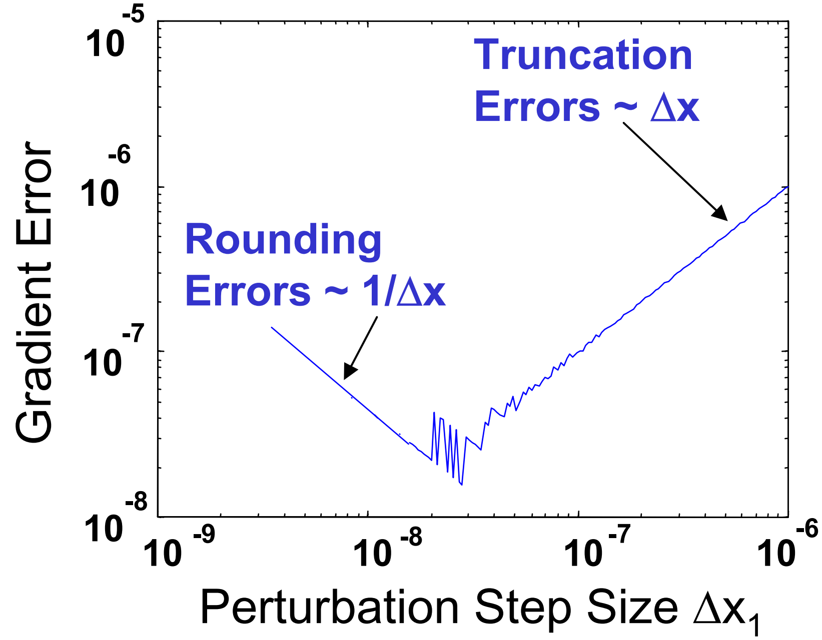
Caution: - Finite Differencing always has errors
- very dependent on perturbation size

$$J(x_1, x_2) = x_1 + x_2 + \frac{1}{x_1 \cdot x_2}$$

$x_1 = x_2 = 1$
$J(1,1)=3$

$$\nabla J(1,1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$





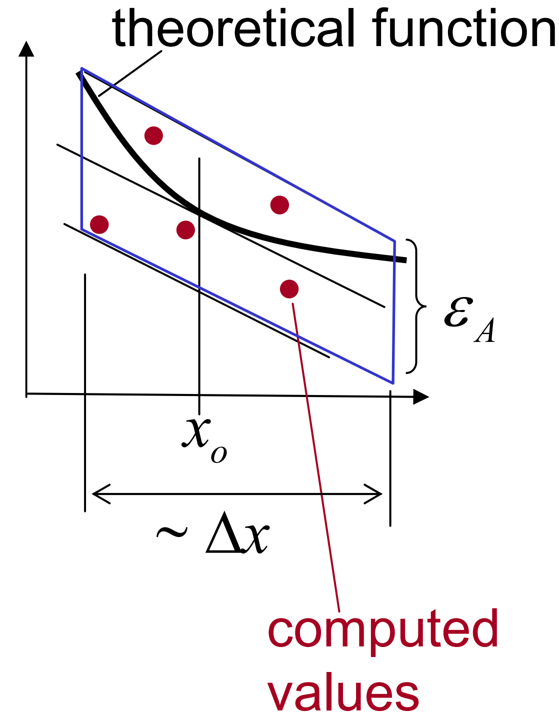
➡ Choice of Δx is critical

Perturbation Size Δx Choice

- Error Analysis (Gill et al. 1981)

$$\Delta x \cong \left(\varepsilon_A / |f|\right)^{1/2}$$ - Forward difference

$$\Delta x \cong \left(\varepsilon_A / |f|\right)^{1/3}$$ - Central difference



- Significant digits (Barton 1992)
- Machine Precision

Step size at k-th iteration $\Delta x_k \cong x_k \cdot 10^{-q}$ *q*-# of digits of machine Precision for real numbers

- Trial and Error – typical value ~ 0.1-1%

Computational Expense of FD

$F(J_i)$ — Cost of a single objective function evaluation of J_i

$n \cdot F(J_i)$ — Cost of gradient vector finite difference approximation for J_i for a design vector of length n

$z \cdot n \cdot F(J_i)$ — Cost of Jacobian finite difference approximation with z objective functions

Example: 6 objectives
30 design variables
1 sec per simcode evaluation

3 min of CPU time for a single Jacobian estimate - expensive !

Automatic Differentiation

- Mathematical formulas are built from a finite set of basic functions, e.g. sin *x*, cos *x*, exp *x*
- Take analysis code in C or Fortran
- Using chain rule, add statements that generate derivatives of the basic functions
- Tracks numerical values of derivatives, does not track symbolically as discussed before
- Outputs modified program = original + derivative capability

Chain Rule example

quantity of interest $\longrightarrow$ $u = q(s)$

$s = p(t)$

Want to take derivatives w.r.t "t"

First compute

$$\frac{ds}{dt} = \frac{d}{dt}p(t)$$

Store this value numerically

substitute

Then apply chain rule

desired sensitivity $\longleftarrow$ $\dfrac{du}{dt} = \dfrac{d}{dt}q(s(t)) = \dfrac{d}{ds}q(s)\cdot\dfrac{ds}{dt}$

Chain Rule Coding Example

16.888 ESD.77

```
hr       =  gm*eps/rho-0.5*g1*(ux*ux + vy*vy);
h_u[0] = -gm*eps/(rho*rho);
h_u[1] = -g1*ux;
h_u[2] = -g1*vy;
h_u[3] =  gm/rho;
hi       = (di*hr+hl)*d1;
hi_u[0] = (di*hr+hl)*d1_u[0]
          + d1*(di_u[0]*hr+di*h_u[0]);
```

- *compute hr*

differentiate:

- *wrt rho*
- *wrt ux*
- *wrt vy*
- *wrt eps*

Adjoint Methods

- A way to get gradient information in a computationally efficient way
- Based on theory from controls
- Applied extensively in aerodynamic design and optimization
- For example, in aerodynamic shape design, need objective gradient with respect to shape parameters **and** with respect to flow parameters
  - Would be expensive if finite differences are used!
- Adjoint methods have allowed optimization to be used for complicated, high-fidelity fluids problems.

Adjoint Methods

Consider $$J = J(\mathbf{w},\mathbf{F})$$

where J is the cost function, **w** contains the N flow variables, and **F** contains the n shape design variables.

At an optimum, the variation of the cost function is zero:

$$\delta J = \underbrace{\left[\frac{\partial J}{\partial \mathbf{w}}\right]^T}_{N\times 1} \delta \mathbf{w} + \underbrace{\left[\frac{\partial J}{\partial \mathbf{F}}\right]^T}_{n\times 1} \delta \mathbf{F} = 0$$

$$N >> n$$

Adjoint Methods

Fluid governing equations: $R(\mathbf{w},\mathbf{F})=0$

$$\delta R=\left[\frac{\partial R}{\partial \mathbf{w}}\right]\delta\mathbf{w}+\left[\frac{\partial R}{\partial \mathbf{F}}\right]\delta\mathbf{F}=0$$

We can append these constraints to the cost function using a Lagrange multiplier approach:

$$\delta J=\left[\frac{\partial J}{\partial \mathbf{w}}\right]^T\delta\mathbf{w}+\left[\frac{\partial J}{\partial \mathbf{F}}\right]^T\delta\mathbf{F}-\varphi^{\mathrm{T}}\left(\left[\frac{\partial R}{\partial \mathbf{w}}\right]\delta\mathbf{w}+\left[\frac{\partial R}{\partial \mathbf{F}}\right]\delta\mathbf{F}\right)$$

$$=\left(\left[\frac{\partial J}{\partial \mathbf{w}}\right]^T-\varphi^{\mathrm{T}}\left[\frac{\partial R}{\partial \mathbf{w}}\right]\right)\delta\mathbf{w}+\left(\left[\frac{\partial J}{\partial \mathbf{F}}\right]^T-\varphi^{\mathrm{T}}\left[\frac{\partial R}{\partial \mathbf{F}}\right]\right)\delta\mathbf{F}$$

Adjoint Methods

$$\delta J = \left(\left[\frac{\partial J}{\partial \mathbf{w}} \right]^T - \varphi^{\mathrm{T}} \left[\frac{\partial R}{\partial \mathbf{w}} \right] \right) \delta \mathbf{w} + \left(\left[\frac{\partial J}{\partial \mathbf{F}} \right]^T - \varphi^{\mathrm{T}} \left[\frac{\partial R}{\partial \mathbf{F}} \right] \right) \delta \mathbf{F}$$

Choose φ to satisfy the adjoint equation:

$$\left[\frac{\partial R}{\partial \mathbf{w}} \right]^{\mathrm{T}} \varphi = \left[\frac{\partial J}{\partial \mathbf{w}} \right]$$

equivalent to one flow solve

Then

$$\delta J = \underbrace{\left(\left[\frac{\partial J}{\partial \mathbf{F}} \right]^T - \varphi^{\mathrm{T}} \left[\frac{\partial R}{\partial \mathbf{F}} \right] \right)}_{\text{total gradient of } J} \delta \mathbf{F}$$

does not depend on the number of flow variables

Sensitivity Analysis

"How does the optimal solution change as we change the problem parameters?"

⇓

effect on design variables
effect on objective function
effect on constraints

Want to answer this question without having to solve the optimization problem again.

Two approaches:

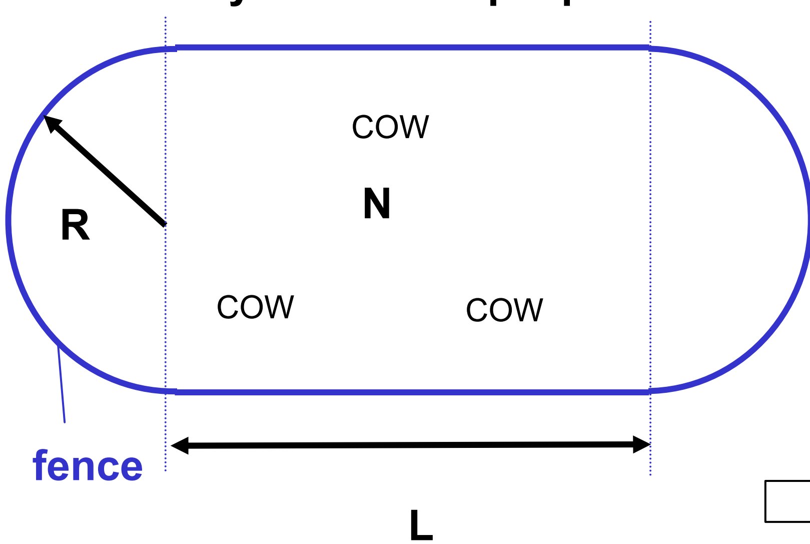
- use Kuhn-Tucker conditions
- use feasible directions

Parameters

Parameters **p** are the fixed assumptions.
How sensitive is the optimal solution x* with respect to fixed parameters ?

Example:

"Dairy Farm" sample problem



Maximize Profit

Optimal solution:

$x^* =[R=106.1m, L=0m, N=17 \text{ cows}]^T$

Fixed parameters:

Parameters:

f=100$/m - Cost of fence

n=2000$/cow - Cost of a single cow

m=2$/liter - Market price of milk

How does x* change as parameters change?

Sensitivity Analysis

Recall the Kuhn-Tucker conditions. Let us assume that we have M active constraints, which are contained in the vector $\hat{\mathbf{g}}(\mathbf{x})$

$$\nabla J(\mathbf{x}^*) + \sum_{j \in M} \lambda_j \nabla \hat{g}_j(\mathbf{x}^*) = 0$$

$$\hat{g}_j(\mathbf{x}^*) = 0, \quad j \in M$$

$$\lambda_j > 0, \quad j \in M$$

For a small change in a parameter, p, we require that the Kuhn-Tucker conditions remain valid:

$$\frac{d(\text{KT conditions})}{dp} = 0$$

Sensitivity Analysis

First, let us write out the components of the first equation:

$$\nabla J(\mathbf{x}^*) + \sum_{j \in M} \lambda_j \nabla \hat{g}_j(\mathbf{x}^*) = 0$$

$$\frac{\partial J}{\partial x_i}(\mathbf{x}^*) + \sum_{j \in M} \lambda_j \frac{\partial \hat{g}_j}{\partial x_i}(\mathbf{x}^*) = 0, \quad i = 1, \ldots, n$$

Now differentiate with respect to the parameter p using the chain rule:

$$\frac{dY}{dp} = \frac{\partial Y}{\partial p} + \sum_{k=1}^{n} \frac{\partial Y}{\partial x_i} \frac{\partial x_i}{\partial p}$$

Sensitivity Analysis

$$\frac{\partial J}{\partial x_i}(\mathbf{x}^*) + \sum_{j \in M} \lambda_j \frac{\partial \hat{g}_j}{\partial x_i}(\mathbf{x}^*) = 0$$

$$\hat{g}_j(\mathbf{x}^*) = 0$$

differentiate wrt p:

$$\sum_{k=1}^{n} \left[\frac{\partial^2 J}{\partial x_i \partial x_k} + \sum_{j \in M} \lambda_j \frac{\partial^2 \hat{g}_j}{\partial x_i \partial x_k} \right] \frac{\partial x_k}{\partial p} +$$

$$\frac{\partial^2 J}{\partial x_i \partial p} + \sum_{j \in M} \frac{\partial^2 \hat{g}_j}{\partial x_i \partial p} + \sum_{j \in M} \frac{\partial \hat{g}_j}{\partial x_i} \frac{\partial \lambda_j}{\partial p} = 0$$

$$\frac{\partial \hat{g}_j}{\partial p} + \sum_{k=1}^{n} \frac{\partial \hat{g}_j}{\partial x_k} \frac{\partial x_k}{\partial p} = 0$$

unknowns are $\frac{\partial x_i}{\partial p}$ and $\frac{\partial \lambda_j}{\partial p}$

Prof. de Weck and Prof. Willcox
Engineering Systems Division and Dept. of Aeronautics and Astronautics

Sensitivity Analysis

In matrix form we can write:

$$\begin{array}{cc} & \begin{array}{cc} \overset{n}{\longleftrightarrow} & \overset{M}{\longleftrightarrow} \end{array} \\ \begin{array}{c} n \updownarrow \\ M \updownarrow \end{array} & \begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \end{array} \begin{Bmatrix} \delta \mathbf{x} \\ \delta \boldsymbol{\lambda} \end{Bmatrix} + \begin{Bmatrix} c \\ d \end{Bmatrix} = 0$$

$$A_{ik} = \frac{\partial^2 J}{\partial x_i \partial x_k} + \sum_{j \in M} \lambda_j \frac{\partial^2 \hat{g}_j}{\partial x_i \partial x_k}$$

$$B_{ij} = \frac{\partial \hat{g}_j}{\partial x_i}$$

$$c_i = \frac{\partial^2 J}{\partial x_i \partial p} + \sum_{j \in M} \frac{\partial^2 \hat{g}_j}{\partial x_i \partial p}$$

$$d_j = \frac{\partial \hat{g}_j}{\partial p}$$

$$\delta \mathbf{x} = \begin{Bmatrix} \dfrac{\partial x_1}{\partial p} \\ \dfrac{\partial x_2}{\partial p} \\ \vdots \\ \dfrac{\partial x_n}{\partial p} \end{Bmatrix} \qquad \delta \boldsymbol{\lambda} = \begin{Bmatrix} \dfrac{\partial \lambda_1}{\partial p} \\ \dfrac{\partial \lambda_2}{\partial p} \\ \vdots \\ \dfrac{\partial \lambda_M}{\partial p} \end{Bmatrix}$$

Sensitivity Analysis

We solve the system to find $\delta\mathbf{x}$ and $\delta\lambda$, then the sensitivity of the objective function with respect to *p* can be found:

$$\frac{dJ}{dp} = \frac{\partial J}{\partial p} + \nabla J^T \delta\mathbf{x}$$

$$\Delta J \approx \frac{dJ}{dp}\Delta p$$

(first-order approximation)

$$\Delta\mathbf{x} \approx \delta\mathbf{x}\,\Delta p$$

To assess the effect of changing a different parameter, we only need to calculate a new RHS in the matrix system.

Sensitivity Analysis - Constraints

- We also need to assess when an active constraint will become inactive and vice versa
- An active constraint will become inactive when its Lagrange multiplier goes to zero:

$$\Delta\lambda_j = \frac{\partial \lambda_j}{\partial p}\Delta p = \delta\lambda_j \, \Delta p$$

Find the Δp that makes λ_j zero:

$$\lambda_j + \delta\lambda_j \, \Delta p = 0$$

$$\boxed{\Delta p = \frac{-\lambda_j}{\delta\lambda_j} \qquad j \in M}$$

This is the amount by which we can change p before the j^{th} constraint becomes inactive (to a first order approximation)

Sensitivity Analysis - Constraints

An inactive constraint will become active when $g_j(\mathbf{x})$ goes to zero:

$$g_j(\mathbf{x}) = g_j(\mathbf{x}^*) + \Delta p \left[\nabla g_j(\mathbf{x}^*)^T \delta \mathbf{x} \right] = 0$$

Find the Δp that makes g_j zero:

$$\boxed{\Delta p = \frac{-g_j(\mathbf{x}^*)}{\nabla g_j(\mathbf{x}^*)^T \delta \mathbf{x}}}$$

for all j not active at $\mathbf{x}^*$

- This is the amount by which we can change p before the j^{th} constraint becomes active (to a first order approximation)
- If we want to change p by a larger amount, then the problem must be solved again including the new constraint
- Only valid close to the optimum

Lecture Summary

- Sensitivity analysis
  - Yields important information about the design space, both as the optimization is proceeding and once the "optimal" solution has been reached.
- Gradient calculation approaches
  - Analytical and Symbolic
  - Finite difference
  - Automatic Differentiation
  - Adjoint methods

Reading

Papalambros – Section 8.2 Computing Derivatives