Multidisciplinary System Design Optimization (MSDO)

Gradient Calculation and Sensitivity Analysis

Lecture 9

Olivier de Weck

Karen Willcox

Today's Topics

- Gradient calculation methods
  - Analytic and Symbolic
  - Finite difference
  - Complex step
  - Adjoint method
  - Automatic differentiation

- Post-Processing Sensitivity Analysis
  - effect of changing design variables
  - effect of changing parameters
  - effect of changing constraints

Definition of the Gradient

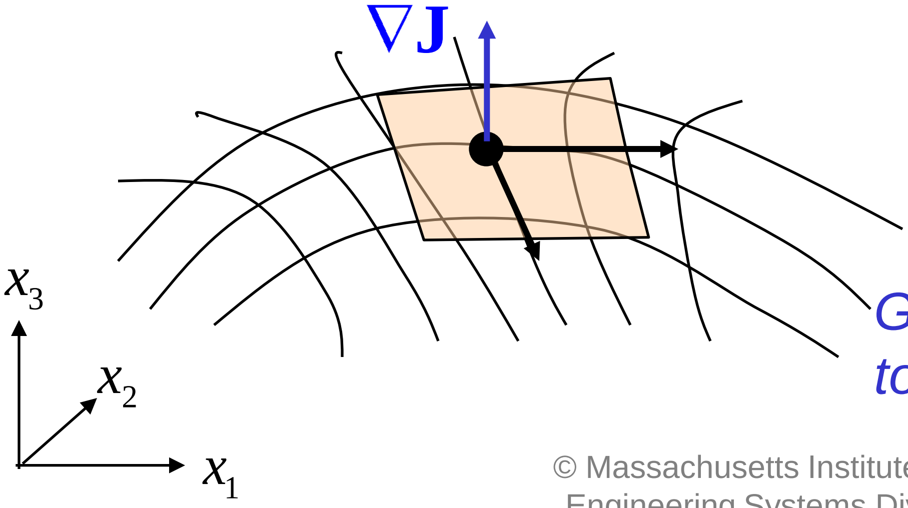
"How does the function *J* value change locally as we change elements of the design vector **x**?"

⇩

Compute partial derivatives of *J* with respect to x_i

$$\frac{\partial J}{\partial x_i}$$

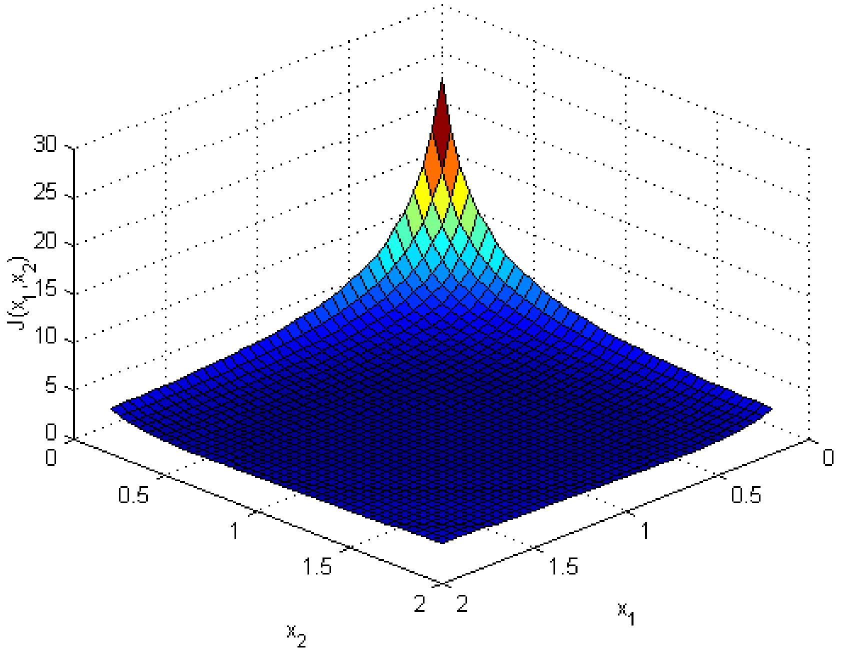
$$\nabla \mathbf{J} = \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \frac{\partial J}{\partial x_2} \\ \vdots \\ \frac{\partial J}{\partial x_n} \end{bmatrix}$$



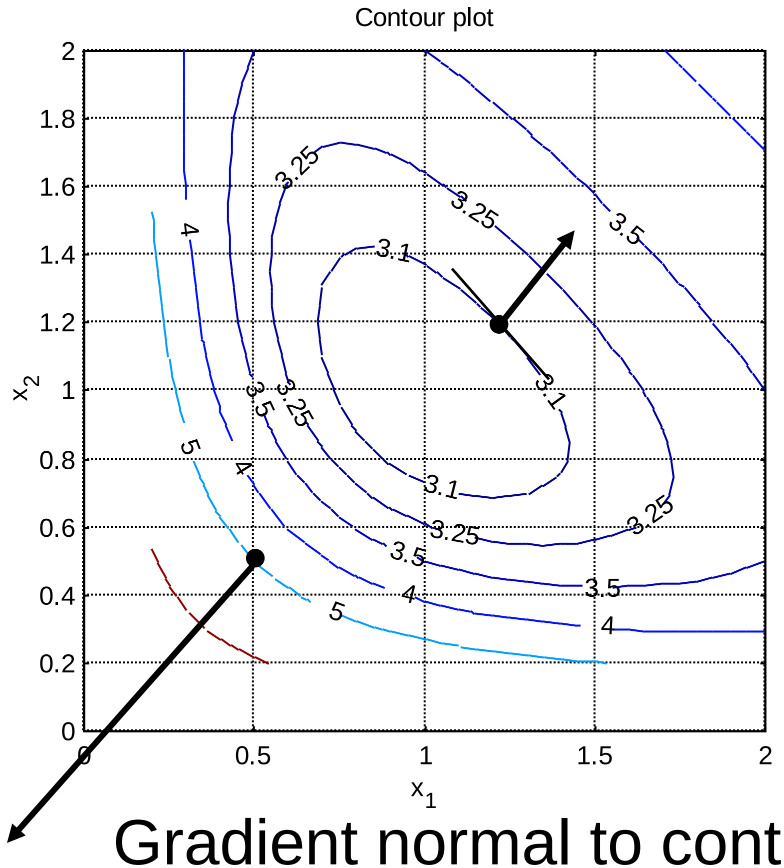
Gradient vector points normal to the tangent hyperplane of J(x)

Geometry of Gradient vector (2D)

Example function: $J\;\; x_1, x_2 \;\; = x_1 + x_2 + \dfrac{1}{x_1 \cdot x_2}$



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

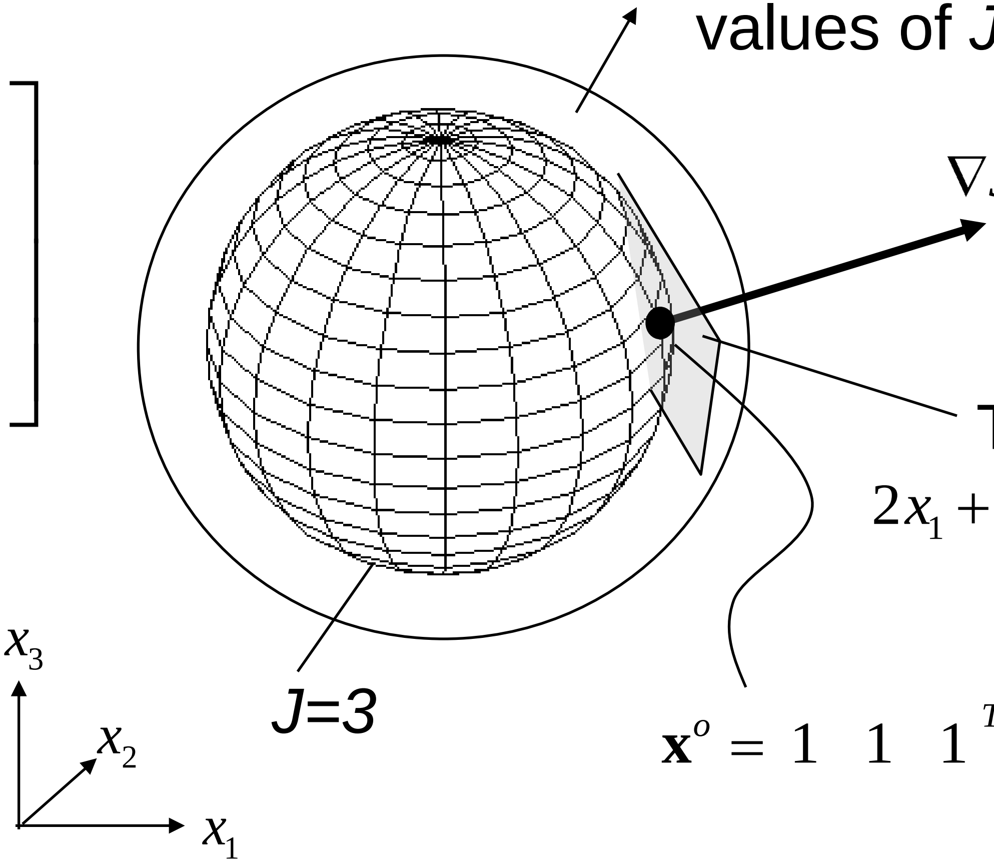


Gradient normal to contours

Geometry of Gradient vector (3D)

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

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



increasing values of *J*

$\nabla J\big|_{\mathbf{x}^o} = \ 2 \quad 2 \quad 2^{\ T}$

Tangent plane

$2x_1 + 2x_2 + 2x_3 - 6 = 0$

J=3

$\mathbf{x}^o = \ 1 \quad 1 \quad 1^{\ T}$

Gradient vector points to larger values of *J*

Other Gradient-Related Quantities

- Jacobian: Matrix of derivatives of multiple functions w.r.t. vector of variables

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

z x 1 *n* x *z*

- Hessian: Matrix of second-order derivatives

$$\mathbf{H} = \nabla^2 J = \begin{bmatrix} \frac{\partial^2 J}{\partial x_1^2} & \frac{\partial^2 J}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_1 \partial x_n} \\ \frac{\partial^2 J}{\partial x_2 \partial x_1} & \frac{\partial^2 J}{\partial x_2^2} & \cdots & \frac{\partial^2 J}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial x_n \partial x_1} & \frac{\partial^2 J}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_n^2} \end{bmatrix}$$

n x *n*

Why Calculate Gradients

- Required by gradient-based optimization algorithms
  - Normally need gradient of objective function and each constraint w.r.t. design variables at each iteration
  - Newton methods require Hessians as well
- Isoperformance/goal programming
- Robust design
- Post-processing sensitivity analysis
  - determine if result is optimal
  - sensitivity to parameters, constraint values

Analytical Sensitivities

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

Example: $J \;\; x_1, x_2 \;\; = x_1 + x_2 + \dfrac{1}{x_1 \cdot x_2}$

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

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

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

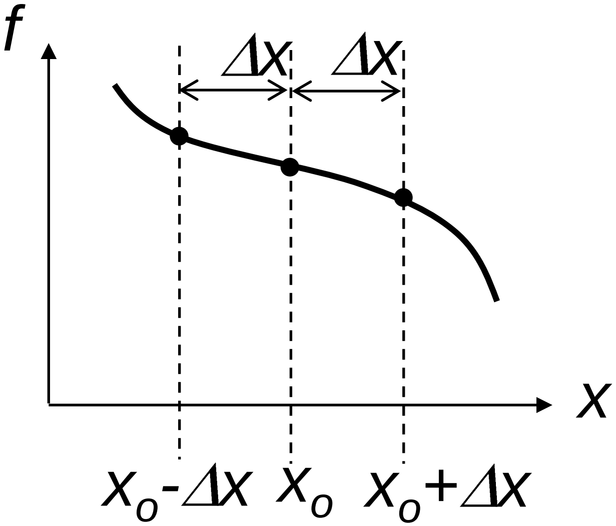
difference operator

Finite Differences (I)

Function of a single variable *f(x)*

- First-order finite difference approximation of gradient:

$$f'(x_o) = \underbrace{\frac{f(x_o+\Delta x) - f(x_o)}{\Delta x}}_{\textit{Forward difference approximation to the derivative}} + \underbrace{O(\Delta x)}_{\textit{Truncation Error}}$$



- Second-order finite difference approximation of gradient:

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

Finite Differences (II)

Approximations are derived from Taylor Series expansion:

$$f\;\; x_o + \Delta x \;\; = f\;\; x_o \;\; + \Delta x f^{'}\;\; x_o \;\; + \frac{\Delta x^2}{2} f^{''}\;\; x_o \;\; + O\;\; \Delta x^3$$

Neglect second order and higher order terms; solve for gradient vector:

$$f^{'}\;\; x_o \;\; = \underbrace{\frac{f\;\; x_o + \Delta x \;\; - f\;\; x_o}{\Delta x}}_{\text{Forward Difference}} + \underbrace{O\;\; \Delta x}$$

Truncation Error

$$O\;\; \Delta x \;\; = \frac{\Delta x}{2} f^{''}\;\; \zeta$$

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

Finite Differences (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

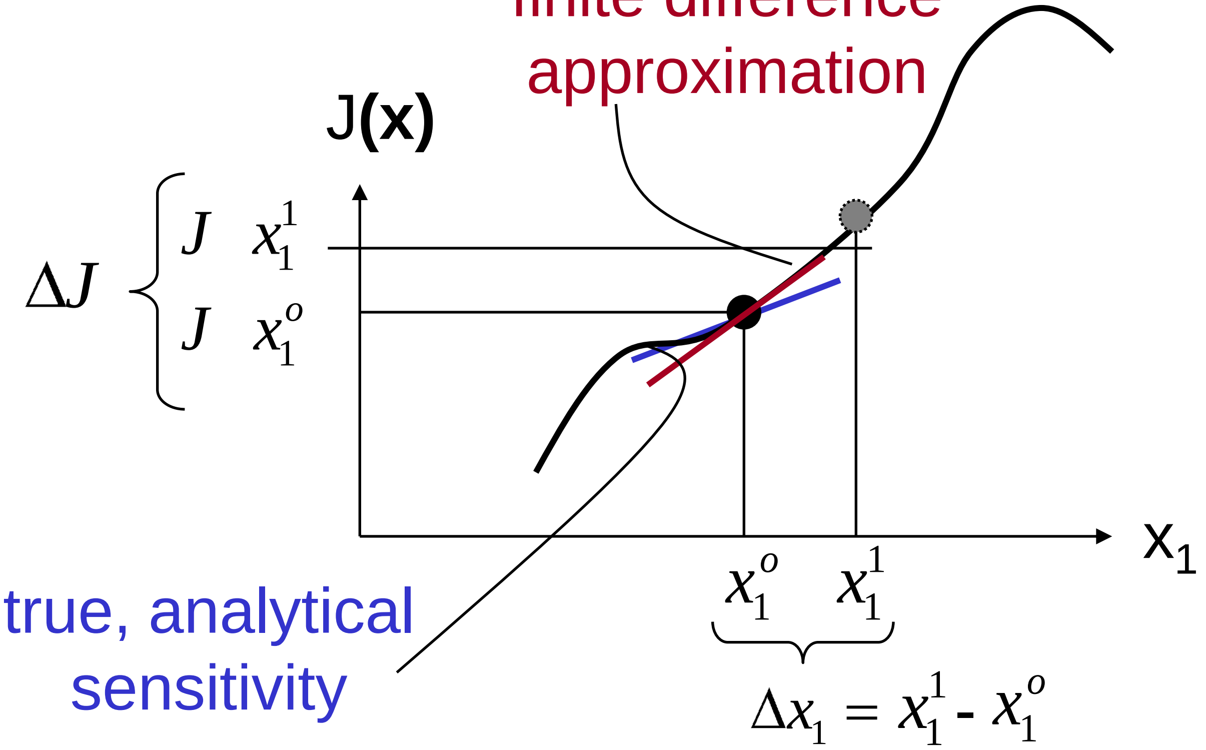
$$f'\;\;x_o\;\;=\underbrace{\frac{f\;\;x_o+\Delta x\;\;-f\;\;x_o-\Delta x}{2\Delta x}}_{\text{Central Difference}}+\underbrace{O\;\;\Delta x^2}_{\text{Truncation Error}}$$

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

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

Finite Differences (IV)

$$\frac{\partial J}{\partial x_1} \approx \frac{J \quad x_1^1 \quad - J \quad x_1^o}{x_1^1 - x_1^o} = \frac{J \quad x_1^o + \Delta x_1 \quad - J \quad x_1^o}{\Delta x_1} = \frac{\Delta J}{\Delta x_1}$$



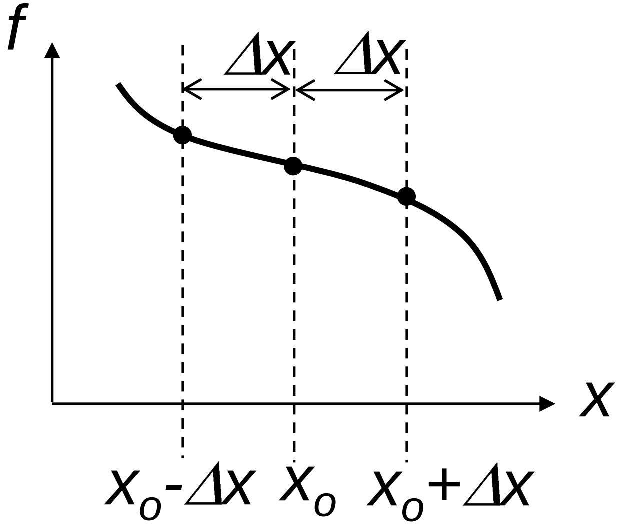
finite difference approximation

true, analytical sensitivity

Finite Differences (V)

- Second-order finite difference approximation of second derivative:

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



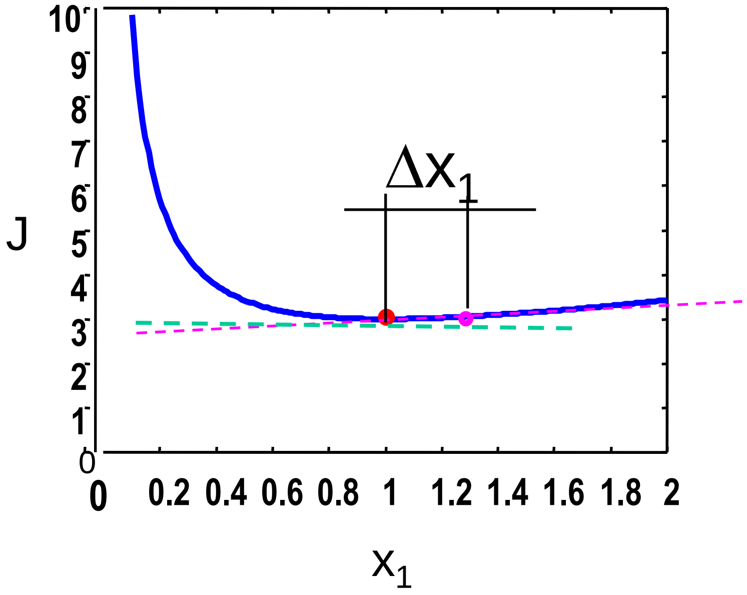
Errors of Finite Differencing

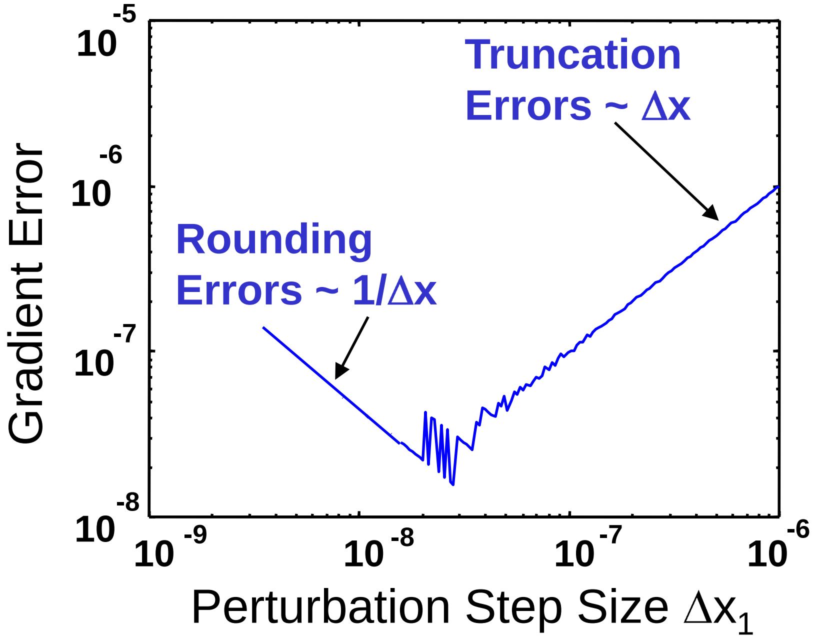
Caution: - Finite differencing always has errors
- Very dependent on perturbation size

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

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

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





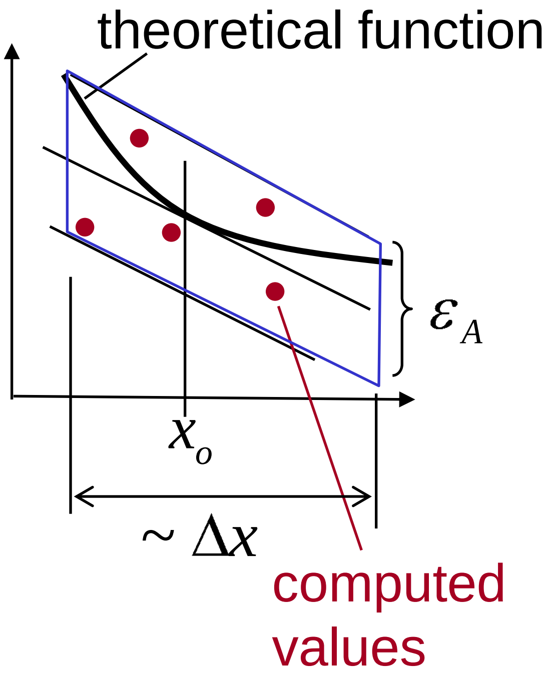
➡ Choice of Δx is critical

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



- 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

$F \quad J_i$

Cost of a single objective function evaluation of J_i

$n \cdot F \quad J_i$

Cost of gradient vector one-sided finite difference approximation for J_i for a design vector of length *n*

$z \cdot n \cdot F \quad J_i$

Cost of Jacobian finite difference approximation with *z* objective functions

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

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

Complex Step Derivative

- Similar to finite differences, but uses an imaginary step

$$f'(x_0) \approx \frac{\mathrm{Im}\left[f(x_0 + i\Delta x)\right]}{\Delta x}$$

- Second order accurate
- Can use very small step sizes e.g. $\Delta x \approx 10^{-20}$
  - Doesn't have rounding error, since it doesn't perform subtraction
- Limited application areas
  - Code must be able to handle complex step values

J.R.R.A. Martins, I.M. Kroo and J.J. Alonso, An automated method for sensitivity analysis using complex variables, AIAA Paper 2000-0689, Jan 2000

Automatic Differentiation

- Mathematical formulae are built from a finite set of basic functions, e.g. additions, sin *x*, exp *x*, etc.
- Using chain rule, differentiate analysis code: 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
- e.g., ADIFOR (FORTRAN), TAPENADE (C, FORTRAN), TOMLAB (MATLAB), many more…
- Resources at http://www.autodiff.org/

Adjoint Methods

Consider the following problem:

Minimize $J(\mathbf{x},\mathbf{u})$

s.t. $\mathbf{R}(\mathbf{x},\mathbf{u})=\mathbf{0}$

where **x** are the design variables and **u** are the state variables.

The constraints represent the state equation.

e.g. wing design: **x** are shape variables, **u** are flow variables, **R**(**x**,**u**)=0 represents the Navier Stokes equations.

We need to compute the gradients of *J* wrt **x**:

$$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{x}}=\frac{\partial J}{\partial \mathbf{x}}+\frac{\partial J}{\partial \mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}}$$

Typically the dimension of **u** is very high (thousands/millions).

Adjoint Methods

$$\frac{dJ}{d\mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}} + \frac{\partial J}{\partial \mathbf{u}} \frac{d\mathbf{u}}{d\mathbf{x}}$$

- To compute d**u**/d**x**, differentiate the state equation:

$$\frac{d\mathbf{R}}{d\mathbf{x}} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \mathbf{u}} \frac{d\mathbf{u}}{d\mathbf{x}} = \mathbf{0}$$

$$\frac{\partial \mathbf{R}}{\partial \mathbf{u}} \frac{d\mathbf{u}}{d\mathbf{x}} = -\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$$

$$\frac{d\mathbf{u}}{d\mathbf{x}} = -\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right)^{-1} \frac{\partial \mathbf{R}}{\partial \mathbf{x}}$$

Adjoint Methods

- We have $$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}} + \frac{\partial J}{\partial \mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}} - \underbrace{\frac{\partial J}{\partial \mathbf{u}}\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right)^{-1}}_{\boldsymbol{\lambda}^T}\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$$

- Now define $$\boldsymbol{\lambda} = \left[\frac{\partial J}{\partial \mathbf{u}}\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right)^{-1}\right]^T$$

- Then to determine the gradient:

  First solve $\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right)^T \boldsymbol{\lambda} = \left(\frac{\partial J}{\partial \mathbf{u}}\right)^T$ (adjoint equation)

  Then compute $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}} - \boldsymbol{\lambda}^T\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$

Adjoint Methods

- Solving adjoint equation

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right)^T \boldsymbol{\lambda} = \left(\frac{\partial J}{\partial \mathbf{u}}\right)^T$$

about same cost as solving forward problem (function evaluation)

- Adjoints widely used in aerodynamic shape optimization, optimal flow control, geophysics applications, etc.

- Some automatic differentiation tools have 'reverse mode' for computing adjoints

Post-Processing Sensitivity Analysis

- A sensitivity analysis is an important component of post-processing
- Key to understanding which design variables, constraints, and parameters are important drivers for the optimum solution
- How sensitive is the "optimal" solution J^* to changes or perturbations of the design variables $\boldsymbol{x}^*$?
- How sensitive is the "optimal" solution $\boldsymbol{x}^*$ to changes in the constraints $\boldsymbol{g}(\boldsymbol{x})$, $\boldsymbol{h}(\boldsymbol{x})$ 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: Spacecraft

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?
- ...

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.

Normalization

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

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

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

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

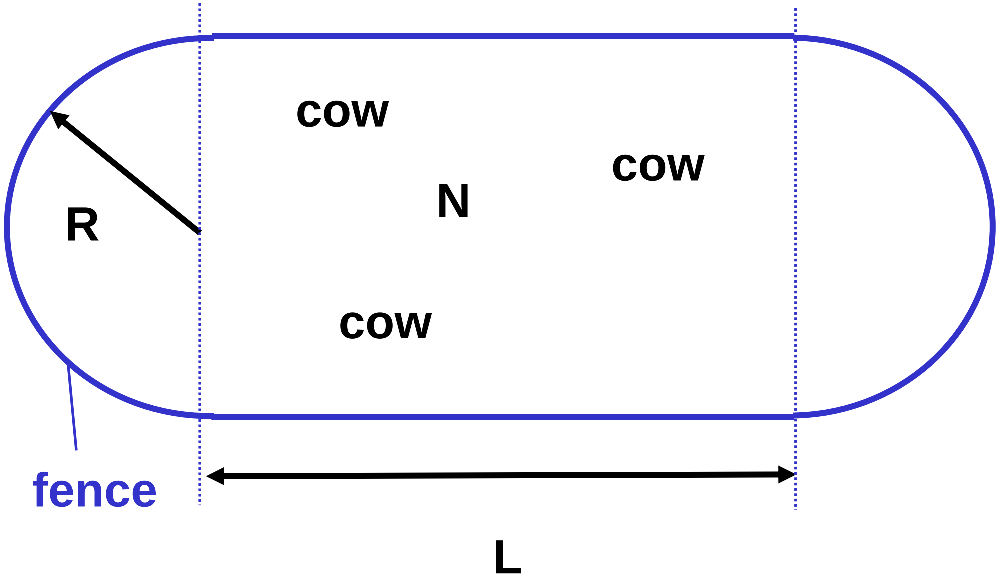
Normalized sensitivity captures relative sensitivity

~ % change in objective per % change in design variable

⇨ Important for comparing effect between design variables

Example: Dairy Farm Problem

“Dairy Farm” sample problem



$$\left.\begin{array}{l}\textbf{L – Length = 100 [m]}\\ \textbf{N - \# of cows = 10}\\ \textbf{R – Radius = 50 [m]}\end{array}\right\} x^o$$

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

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

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

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

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

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

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

$$P = I - C$$

Dairy Farm Sensitivity

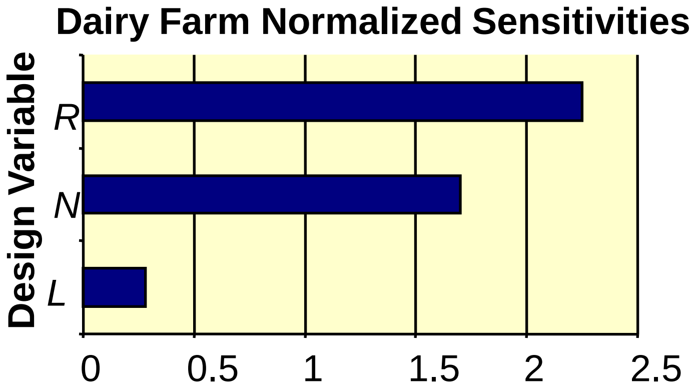
- Compute objective at **x°** $J(\mathbf{x}^o) = 13092$
- Then compute raw sensitivities

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

- Normalize

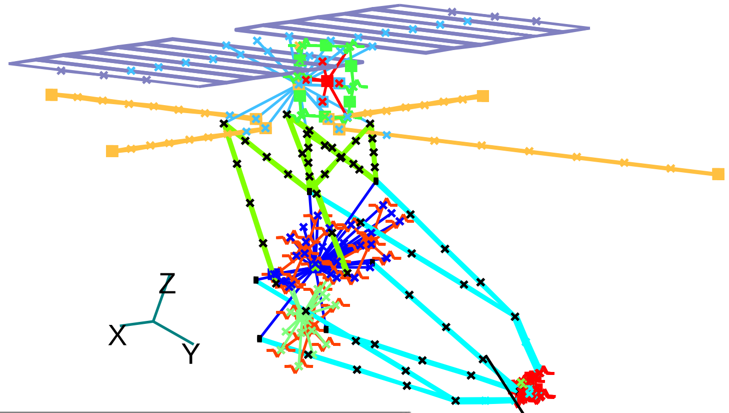
$$\nabla \bar{J} = \frac{\mathbf{x}^o}{J(\mathbf{x}^o)} \nabla J = \begin{bmatrix} \frac{100}{13092} \cdot 36.6 \\ \frac{10}{13092} 2225.4 \\ \frac{50}{13092} 588.4 \end{bmatrix} = \begin{bmatrix} 0.28 \\ 1.7 \\ 2.25 \end{bmatrix}$$

- Show graphically with tornado chart



Realistic Example: Spacecraft

NASA Nexus Spacecraft Concept



Finite Element Model

0 1 2
meters

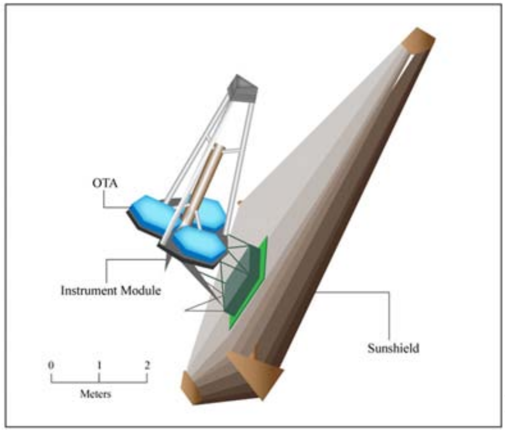
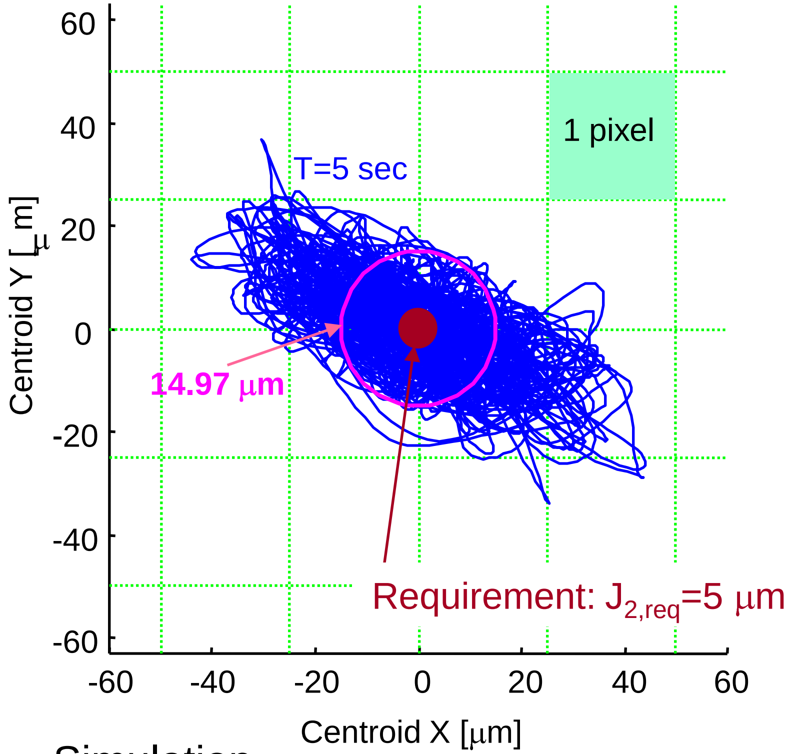


Image by MIT OpenCourseWare.

Spacecraft CAD model

J_2= Centroid Jitter on Focal Plane [RSS LOS]

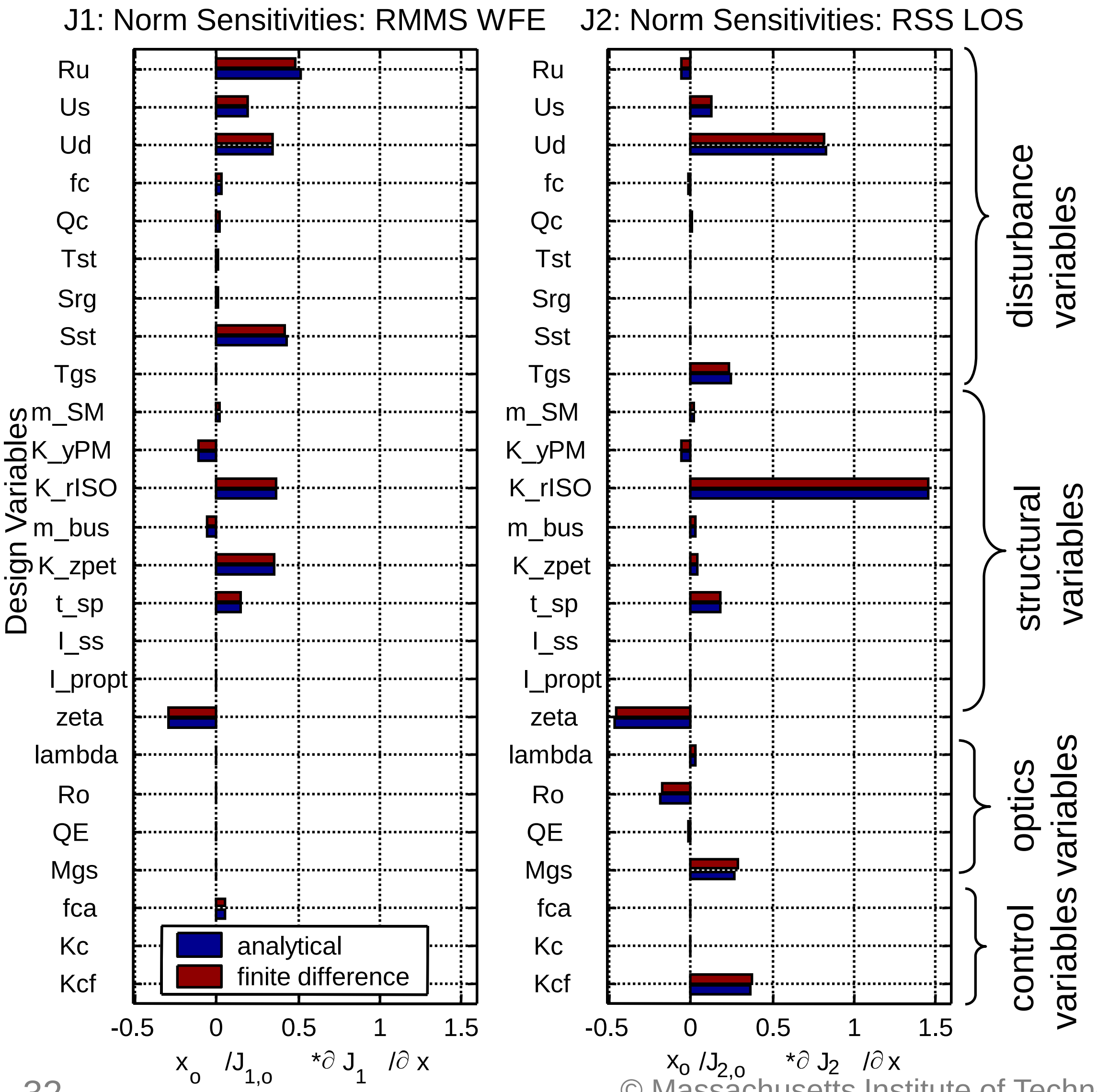


Simulation

"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:

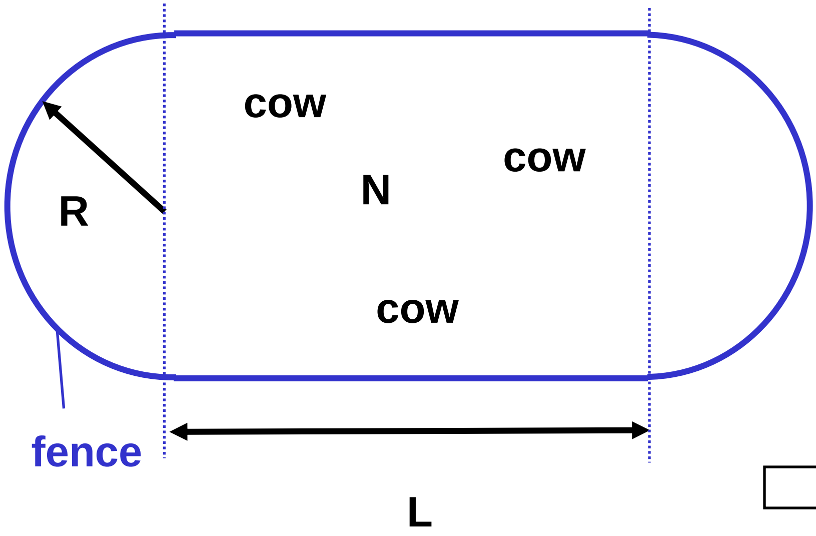
Ud - dynamic wheel imbalance [gcm^2]
K_rISO - isolator joint stiffness [Nm/rad]
zeta - proportional damping ratio [-]
Mgs - guide star magnitude [mag]
Kcf - FSM controller gain [-]

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:

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

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

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

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

Set of active constraints

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

For a small change in a parameter, *p*, we require that the KKT conditions remain valid:

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

Rewrite first equation:

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

Sensitivity Analysis

Recall chain rule. If: $Y = Y\big(p, \mathbf{x}(p)\big)$ then

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

Applying to first equation of KKT conditions:

$$\frac{d}{dp}\left(\frac{\partial J(\mathbf{x},p)}{\partial x_i} + \sum_{j\in M}\lambda_j(p)\frac{\partial \hat{g}_j(\mathbf{x},p)}{\partial x_i}\right)$$

$$= \frac{\partial^2 J}{\partial x_i \partial p} + \sum_{j\in M}\lambda_j\frac{\partial^2 J}{\partial x_i \partial p} + \sum_{k=1}^{n}\left(\frac{\partial^2 g}{\partial x_i \partial x_k} + \sum_{j\in M}\lambda_j\frac{\partial^2 \hat{g}_j}{\partial x_i \partial x_k}\right)\frac{\partial x_k}{\partial p} + \sum_{j\in M}\frac{\partial \lambda_j}{\partial p}\frac{\partial \hat{g}_j}{\partial x_i} = 0$$

$$\sum_{k=1}^{n} A_{ik}\frac{\partial x_k}{\partial p} + \sum_{j\in M} B_{ij}\frac{\partial \lambda_j}{\partial p} + c_i = 0$$

Sensitivity Analysis

Perform same procedure on equation: $g_j(x^*, p) = 0$

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

$$\sum_{k=1}^{n} B_{kj} \frac{\partial x_k}{\partial p} + d_j = 0$$

Sensitivity Analysis

In matrix form we can write:

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

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

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

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

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

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

Sensitivity Analysis

We solve the system to find δ**x** and δλ, then the sensitivity of the objective function with respect to *p* can be found:

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

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

(first-order approximation)

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

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 **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

Lagrange Multiplier Interpretation

- Consider the problem:

  minimize J(**x**) s.t. **h**(**x**)=0

  with optimal solution **x***

- What happens if we change constraint *k* by a small amount?

$$h_k(\mathbf{x}^*) = \varepsilon \qquad h_j(\mathbf{x}^*) = 0, \quad \forall j \neq k$$

- Differentiating w.r.t ε

$$\nabla h_k \frac{d\mathbf{x}^*}{d\varepsilon} = 1 \qquad \nabla h_j \frac{d\mathbf{x}^*}{d\varepsilon} = 0, \quad \forall j \neq k$$

Lagrange Multiplier Interpretation

- How does the objective function change?

$$\frac{dJ}{d\varepsilon} = \nabla J \frac{d\mathbf{x}^*}{d\varepsilon}$$

- Using KKT conditions:

$$\frac{dJ}{d\varepsilon} = \left(-\sum_j \lambda_j \nabla h_j \right) \frac{d\mathbf{x}^*}{d\varepsilon} = -\sum_j \lambda_j \nabla h_j \frac{d\mathbf{x}^*}{d\varepsilon} = -\lambda_k$$

- Lagrange multiplier is negative of sensitivity of cost function to constraint value. Also called *shadow price.*

Lecture Summary

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

Reading

Papalambros – Section 8.2 Computing Derivatives