Lecture - 5
NUMERICAL ESSENTIALS

Reference: Book Chapter 7

NUMERICAL CONDITIONING

- **Numerical conditioning?**
  - A **numerically well-conditioned** problem or matrix is one that lends itself to easy numerical computation.
  - A **numerically ill-conditioned** problem or matrix is one that lends itself to difficult numerical computation.
- Numerical conditioning can also refer to the property of a particular algorithm.
- A well-conditioned algorithm is likely to converge relatively easily, while an ill-conditioned algorithm may converge after many more iterations, or may not converge at all.

NUMERICAL CONDITIONING

(i) Algorithms:

- Reputed optimization codes are usually adequately well-conditioned.

(ii) Matrices:

- The numerical conditioning of a matrix is quantified by the condition number.
- For symmetric matrices, the condition number is the square of the ratio of the highest to the lowest eigenvalues.
- A condition number with order-of-magnitude one is very good.

(iii) Optimization Problems: Use scaling.

POSSIBLE REASONS FOR FAILURES

1. Coding bugs:

Employ the debugging strategy.

2. Ill conditioned:

Properly scale the problem.

3. Mistakenly formulated:

Proper formulation.

4. Unrealistic design:

Examine constraints, objectives, and models.

5. Algorithm:

(i) Not robust enough for problems with poor numerical conditioning;

(ii) Not appropriate for problems of large dimension;

(iii) Limited to solving only a specific type of problem.

(iv) Noisy objective functions or constraints.

Example:

An example about ***numerical conditioning issues***

$$A = \begin{bmatrix} 1/\alpha & 0 & 0 \\ 0 & (\alpha+1)/2 & (\alpha-1)/2 \\ 0 & (\alpha-1)/2 & (\alpha+1)/2 \end{bmatrix}$$

- The three eigenvalues of the matrix ***A*** can be evaluated as: $1/\alpha$, 1, and α.
- The condition number of ***A*** (the square of the ratio of the highest to the lowest eigenvalue) is $1/\alpha^4$ when $\alpha \leq 1$, and is α^4 when $\alpha \geq 1$.
- For very low and very high values of α, we may experience numerical difficulties

EXPOSING NUMERICAL CONDITIONING ISSUES

- ***n*** is any positive integer. **I** is the identity matrix.

$$A_1 = (A \cdot A^{-1})^n - I = [0]$$
$$A_2 = A^n \cdot (A^{-1})^n - I = [0]$$

- Using elementary linear algebra, both $\mathbf{A_1}$ and $\mathbf{A_2}$ are identically equal to 3 × 3 zero matrices.

$$\| A_1 \| = \| (A \cdot A^{-1})^n - I \| = 0$$
$$\| A_2 \| = \| A^n \cdot (A^{-1})^n - I \| = 0$$

- Use the maximum norm defined as $\mathbf{|| M ||= max\{| } m_{ij} \mathbf{ |\}}$, with m_{ij} denoting the ***ij*** -**th** entry of the matrix M.

- The zero answers are exact only from a theoretical standpoint. Compute $\mathbf{||}A_1\mathbf{||}$ and $\mathbf{||}A_2\mathbf{||}$ using a computer, the numerical results depart markedly from the theoretical answers.

EXPOSING NUMERICAL CONDITIONING ISSUES

α	C_n	$\| A_1 \| = \| (A \cdot A^{-1})^n - I \|$			$\| A_2 \| = \| A^n \cdot (A^{-1})^n - I \|$		
		n			n		
		10	20	50	10	20	50
0	∞	–	–	–	–	–	–
10^{-6}	10^{24}	$3e^{-10}$	$6e^{-10}$	$1e^{-9}$	**1**	**1**	**1**
10^{-3}	10^{12}	0	0	0	**1**	**1**	**1**
0.4	39	$1e^{-15}$	$2e^{-15}$	$6e^{-15}$	$5e^{-13}$	$4e^{-9}$	**1**
0.8	2.4	0	0	0	$1e^{-15}$	$2e^{-15}$	$4e^{-12}$
0.95	1.2	$3e^{-17}$	$7e^{-17}$	$2e^{-16}$	$7e^{-16}$	$9e^{-16}$	$3e^{-15}$
1	1	0	0	0	0	0	0
2	16	0	0	0	0	0	0
10	10^4	$4e^{-15}$	$9e^{-15}$	$2e^{-14}$	$1e^{-15}$	**1**	**1**
10^2	10^8	0	0	0	**1**	**1**	**1**
10^5	10^{20}	$4e^{-11}$	$7e^{-11}$	$2e^{-10}$	$\mathbf{5e^{33}}$	$\mathbf{10e^{83}}$	$\mathbf{4e^{233}}$

- Even for high values of C_n, $||A_1||$ is evaluated accurately.
- For values of the condition number that are even less than ***100 (α = 0.4)***, $||A_2||$ is unacceptably inaccurate.

EXPOSING NUMERICAL CONDITIONING ISSUES

α	C_n	$\| A_1 \| = \| (A \cdot A^{-1})^n - I \|$			$\| A_2 \| = \| A^n \cdot (A^{-1})^n - I \|$		
		n			n		
		10	20	50	10	20	50
0	∞	–	–	–	–	–	–
10^{-6}	10^{24}	$3e^{-10}$	$6e^{-10}$	$1e^{-9}$	**1**	**1**	**1**
10^{-3}	10^{12}	0	0	0	**1**	**1**	**1**
0.4	39	$1e^{-15}$	$2e^{-15}$	$6e^{-15}$	$5e^{-13}$	$4e^{-9}$	**1**
0.8	2.4	0	0	0	$1e^{-15}$	$2e^{-15}$	$4e^{-12}$
0.95	1.2	$3e^{-17}$	$7e^{-17}$	$2e^{-16}$	$7e^{-16}$	$9e^{-16}$	$3e^{-15}$
1	1	0	0	0	0	0	0
2	16	0	0	0	0	0	0
10	10^4	$4e^{-15}$	$9e^{-15}$	$2e^{-14}$	$1e^{-15}$	**1**	**1**
10^2	10^8	0	0	0	**1**	**1**	**1**
10^5	10^{20}	$4e^{-11}$	$7e^{-11}$	$2e^{-10}$	$\mathbf{5e^{33}}$	$\mathbf{10e^{83}}$	$\mathbf{4e^{233}}$

- The computation of $||A_1||$ is more numerically robust than that of $||A_2||$.
- In general, the stability of the algorithm and the condition numbers of the matrices involved can greatly impact the accuracy of the solutions obtained.

Example:

$$\min_{x_1,x_2} \quad f(x_1,x_2) = (x_1 - 3.67 \times 10^{-6})^2 + (x_2 - 3.67 \times 10^{-7})^2$$

$$\text{subject to}$$

$$x_1 - 2x_2 = 0$$

$$0 \le x_i \le 10^{-4} \quad (i = 1, 2)$$

- Using scaling, the solution is $\boldsymbol{x = 10^{-6} \times \{3.083, 1.541\}}$.
- Without scaling, MATLAB converged to the inaccurate solution x = $10^{-6}\times\{4.948, 2.474\}$.

➢ *fmincon Optimization Model*

$$\min_{\mathtt{x}} \ \mathtt{f(x)}$$

subject to

$$\mathtt{c(x)} \leq 0$$

$$\mathtt{ceq(x)} = 0$$

$$\mathtt{A\,x} \leq \mathtt{b}$$

$$\mathtt{Aeq\ x} = \mathtt{beq}$$

$$\mathtt{LB} \leq \mathtt{x} \leq \mathtt{UB}$$

$$\min_{x} \ f(x)$$

subject to

$$g(x) \leq 0$$

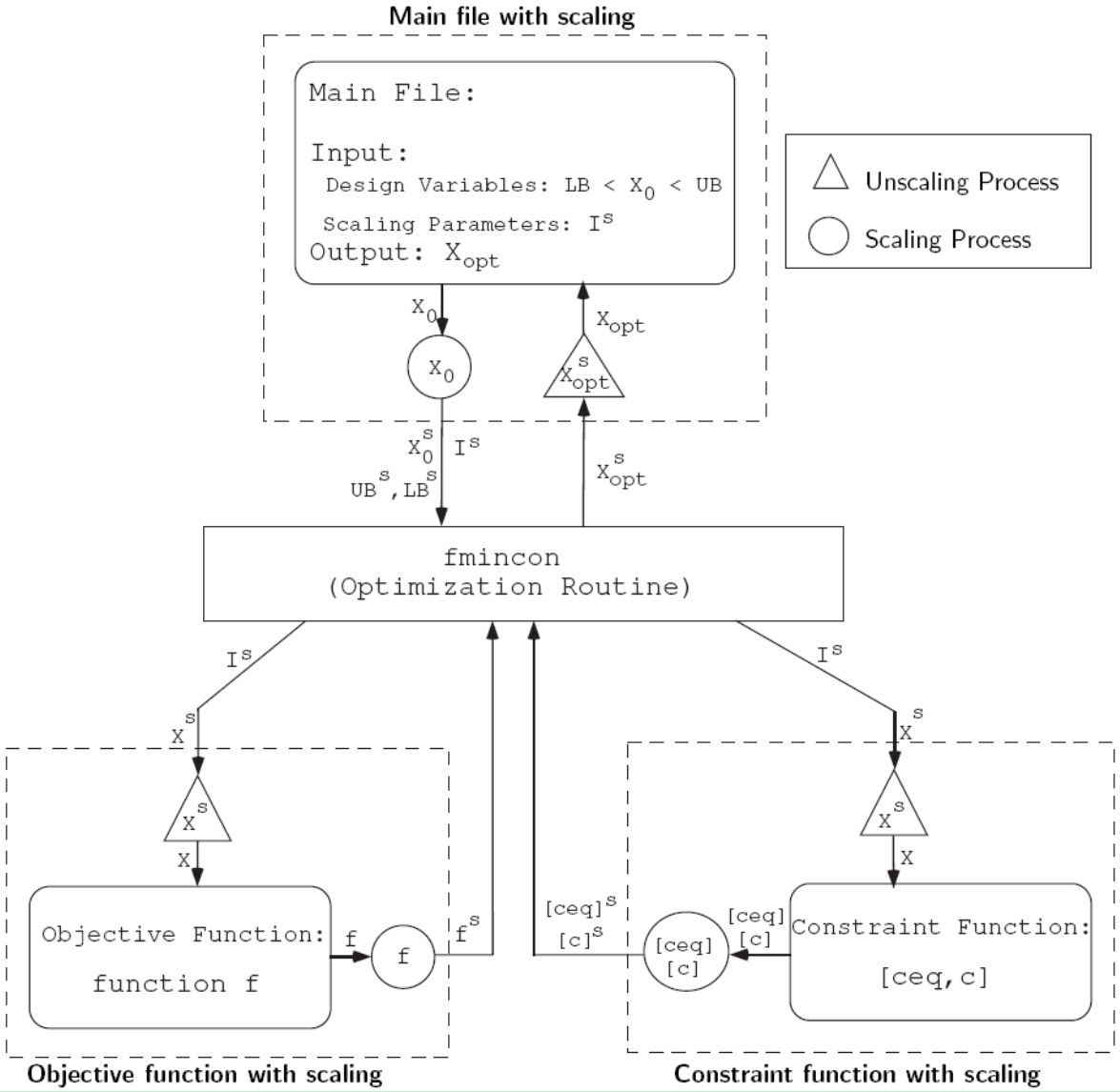
$$h(x) = 0$$

$$x_l \leq x \leq x_u$$

General Formulation

- The ***fmincon*** Optimization Model provides the flexibility to differentiate between linear and nonlinear constraints.

EXPOSING NUMERICAL CONDITIONING ISSUES



ACCURACY OF THE REPORTED RESULTS

- An important aspect of our work is to understand

  (i) ***the accuracy of the results we obtain***, and

  (ii) ***how to report these results***.

- These issues entail two related components.

  - **The first involves** understanding and controlling the accuracy of the results produced by the optimization code – using scaling.

  - **The second involves** understanding the inherent accuracy of the physical models used in the optimization.

ACCURACY OF THE REPORTED RESULTS

- **Decide how many digits of the number are significant.**
  - **Example;** We may use 12,345,000, or 1.2345×10^7, for 5 significant digits, even though the first 8 digits in the computer memory might be 12,345,678.
  - As we decided to report 5 significant digits, we did so simply because we determined that the optimization converged to within 5 significant digits.
  - This convergence does not tell us that these 5 significant digits are physically meaningful.

ACCURACY OF THE REPORTED RESULTS

- For the optimal value of the deflection of a beam as 1.23456×10^{-6}, perhaps only 3 or 4 of these digits might be physically meaningful.
- Depending on the model we used, the result might only have 4 significant digits.
- The physical model produces n_{pm} significant digits of accuracy.
- The optimization code converges to within n_{op} significant digits of accuracy, then it is advisable to use n_r such that

$$n_r \leq \min\{n_{pm}, n_{op}\}$$

DESIGN VARIABLE SCALING

➢ Order of Magnitude

- Design variables scaling is often desirable anytime the order of magnitude of the design variables is much higher or much lower than 1.

- Multiply each design variable by a constant that will (i) bring it close to 1, and/or (ii) address some tolerance/accuracy issues.

$$\mathbf{x}_i^s = \alpha_i^s \mathbf{x}_i \qquad (i = 1, ..., n_x)$$

- The unscaling process takes $x_i = x_i^s / \alpha_i^s$

- The scaling for the upper and lower bounds:

$$\{\mathrm{LB}_i^s, \mathrm{UB}_i^s\} = \alpha_i^s \{\mathrm{LB}_i, \mathrm{UB}_i\} \qquad (i = 1, ..., n_x)$$

DESIGN VARIABLE SCALING

➢ Order of Magnitude

- In the case where the initial design variable is several orders of magnitude different from the final/optimal value, it may be ***necessary to perform more than one optimization run***, ***each with an updated scaling*** – i.e., each time using the final design variable value of one run, as the initial value of the next run.

Example:

- $\mathbf{x_{3-initial}} = 12345.67$. $\mathbf{x_{3-optimal}} = 0.012345$.
- Determine the scaling factor using $x_{3-initial}$
- Determine a new scaling factor using the results obtained for $\boldsymbol{x_3}$ from the first optimization run.
- Perform a second optimization run.

DESIGN VARIABLE SCALING

➢ Tolerance Definition

- ***Decimal Accuracy (DA)***
- ***Number of Significant Digits (NSD)***

Generic Number	DA	NSD
0.0123	10^{-4}	3
100.0123456	10^{-7}	10
123000	10^{3}	3

DESIGN VARIABLE SCALING

➢ Optimization Code Decimal Accuracy Setting

- The number of significant digits of a number for different scenarios.

Scenario 1

- Optimize the thickness of a sheet of paper, with order of magnitude of 10^{-3} (say, 0.001,234,567).
- Decimal accuracy 10^{-6}, yielding 0.001,234.

Scenario 2

- Optimize the annual revenue of a company with order of magnitude 10^{9} (say, $1,234,567,891.234).
- Decimal accuracy 10^{3}, yielding $1,234,567,000.

DESIGN VARIABLE SCALING

➢ Optimization Code Decimal Accuracy Setting

- **MATLAB** decimal accuracy (***TolX***) 10^{-6}.
- Results of Scenario 1: 0.001,234.
- **Results of Scenario** 2: $1,234,567,891.234,000.
- Scenario 2: ***16 significant digits*** of accuracy is problematic. The code will either not converge to this many significant digits, or it may take a long time to do so.

DESIGN VARIABLE SCALING

➢ Combining Order of Magnitude and Desired Tolerance

Scenario 1:

- Use scaling and unscaling to obtain 5 significant digits of accuracy.
- An alternative approach for obtaining 5 significant digits is to let ***TolX=10^{-7}***.
- Changing ***TolX*** in MATLAB affects all design variables, and not just x_i.

Scenario 2:

- To obtain 7 significant digits of accuracy, scale and let $\alpha^s_i = 10^{-9}$, leading to x^s_i=1.234,567,891,234, yielding the final value x^s_i=1.234,567.
- After un-scaling: x_i =\$1, 234, 567, 000.

DESIGN VARIABLE SCALING

➢ Setting Scaling Parameters

- The magnitude of the design variable x_i has a direct impact:
  - ***on the success of the optimization (i.e., convergence),***
  - ***on the computation time,***
  - ***on the accuracy of the final result (e.g., tolerance).***

DESIGN VARIABLE SCALING

➢ Setting Scaling Parameters

❑ *How to bring optimal design variables to the order of magnitude 1*?

- Divide the design variable by a number that is approximately equal to its optimal value.
- For ***TolX=10^{-6}***, and ***7*** significant digits of accuracy, the optimal value is approximately on the order of **1**. To yield ***7 + n_{dvi}*** significant digits of accuracy:

$$\alpha_i^s = 10^{n_{dvi}} / x_{i-typical} \qquad (i = 1, \ldots, n_x)$$

- When ***n_{dvi}*** is negative, we will have decreased the accuracy by ***n_{dvi}*** digits.
- The absolute value of ***n_{dvi}*** should generally be no greater than ***3***, to avoid requesting excessive accuracy.

DESIGN VARIABLE SCALING

➢ Setting Scaling Parameters

❑ *Two important notes:*

- A design variable is nearly zero: ***unnecessary to scale***.
- Design variables are nearly of the same order of magnitude with similar tolerances: ***it may be acceptable to use a single value of the constant α for all design variables***.

OBJECTIVE FUNCTION SCALING

- Most optimization codes are designed to perform well when *the objective function value is on the order of 1*.
- It is safer to let objective functions take on the desired order of magnitude.
- For values that lie significantly outside of this range, the optimization algorithm can become ill-conditioned.
- The scaling of the objective function only needs to be performed after its evaluation in objective function.
- **Scaling form**:

$$\mathrm{f}^{\mathrm{s}}(x) = \beta^{s}\mathrm{f}(x)$$

- **Un-scale in the Main file by** $f = f^{s}/\beta^{s}$.

OBJECTIVE FUNCTION SCALING

- In **MATLAB**: ***TolFun*** represents the decimal accuracy of the objective function. The default value is 10^{-6}.
- To yield $7 + n_{obj}$ significant digits of accuracy.

$$\beta^s = 10^{n_{obj}} / f_{typical}$$

- Negative n_{obj} decrease n_{obj} digits.
- The absolute value of n_{obj} should generally be no more than 3, to avoid requesting excessive accuracy.
- If the optimal value of the objective function is near zero, scaling may be unnecessary.

OBJECTIVE FUNCTION SCALING

- Some codes set the accuracy of the objective function by *prescribing the number of significant digits, rather than the decimal accuracy*.
- ***Example;***
  - if the optimal value of the objective function is f_{opt}***=1,234.123,456,789***, the former case will yield f_{opt}***=1,234.12***, while the latter case will f_{opt}***=1,234.123,456*** (without scaling).
  - On the other hand, if we have $\mathbf{f_{opt}=0.000,001,234,567}$, the former case will yield $\mathbf{f_{opt}=0.000,001,234,56,}$ while the latter case will $\mathbf{f_{opt}=0.000,001}$ (without scaling).
- **With scaling, both approaches can be made to yield the same answer**.

BEHAVIORAL CONSTRAINTS SCALING

- The following constraints are satisfied to within a given decimal accuracy.

$$\mathtt{ceq(x)} = 0$$

- In **MATLAB**: The tolerance parameter is ***TolCon***. The default value is 10^{-6}.
- The optimization algorithm stops when
  - ***ceq(x) = 10^{-7} is considered numerically acceptable.***

$$\left|\mathtt{ceq(x)}_j\right| \leq \mathtt{TolCon} \qquad (j = 1, ..., n_{ec})$$

BEHAVIORAL CONSTRAINTS SCALING

Scenario 1

- Constraint: $2x_1^2+x_2 = 0$.
- The design variables: $\mathbf{x_1 = x_2 = 2.5\times10^{-7}}$.
- $\mathbf{2x_1^2+x_2 = 2.50000125\times10^{-7} \le 10^{-6}}$. It numerically satisfies the constraint.
- Let $\mathbf{ceq(x)_1 = (2x_1^2 + x_2) \times 10^6}$.
- It yields $2.50000125\times10^{-1} > 10^{-6}$. Not satisfied.
- Multiplying the constraint by a constant enforces satisfaction of the constraint within the accuracy.
- It also applies to inequality constraints.

BEHAVIORAL CONSTRAINTS SCALING

➢Scenario 2

- Constraint: $ceq_1 = x_1 - 2\,x_2$.
- x_1 = 2,222,222.33. x_2=1,111,111.11.
- For practical purposes, these values of x satisfy the constraint. However, it yields $0.11 \geq 10^{-6}$, which is clearly not satisfied.
- Let $ceq(x)_1 = (x_1 - 2\,x_2) \times 10^{-6}$.
- It yields $0.11 \times 10^{-6} \leq 1.0 \times 10^{-6}$, which is satisfied.
- Multiplying the constraint by a constant enforces satisfaction of the constraint within the accuracy.

CONSTRAINT SCALING APPROACH

- **Scale equality constraints**:

$$\mathrm{ceq}(\mathrm{x})_i^s = \gamma_i^s \mathrm{ceq}(\mathrm{x})_i \qquad (i = 1, \ldots, n_{ec})$$

γ_i^s is a constant used to increase or decrease the degree of constraint satisfaction.

CONSTRAINT SCALING APPROACH

- The value for γ_i^s depends on
  (i) The accuracy to be satisfied
  (ii) The value of the constraints setting (***TolCon***).

- We can address this question in various ways.
  (a)The most direct way is to simply let $\gamma_i^s = 10^n$ **or** 10^{-n}.
  (b) The pseudo-normalization scheme.

$$\gamma_i^s = 10^{n_{eqi}} / \mathrm{ceq}_{i-nominal}$$

$\mathbf{c_{eqi-nominal}}$ is a nominal value of $\boldsymbol{c_{eqi}}$.

- Three representative cases for $c_{eq\text{-}nominal}$:

$$\mathrm{ceq(x)}_i \equiv r(x) - 1.23 \times 10^{-8} = 0; \quad \mathrm{ceq}_{i-nominal} = 1.23 \times 10^{-8}$$
$$\mathrm{ceq(x)}_i \equiv r(x) - 1.23 \times 10^{8} = 0; \quad \mathrm{ceq}_{i-nominal} = 1.23 \times 10^{8}$$
$$\mathrm{ceq(x)}_i \equiv 2x_1 + x_2 = 0; \quad \mathrm{ceq}_{i-nominal} = x_{2-typical}$$

SETTING MATLAB OPTIMIZATION OPTIONS, AND SCALING PARAMETERS

Change certain optimization settings:

- Several at once:

```
options = optimset('TolFun', 1e-8, 'TolX', 1e-7, 'TolCon', 1e-6)
```

- One at a time:

```
options1 = optimset
options2 = optimset(options1, 'TolFun', 1e-8)
options3 = optimset(options2, 'TolX', 1e-7)
options = optimset(options3, 'TolCon', 1e-6)
```

- Any set of "options" can be used in the ***fmincon*** call.

```
[xopt,fopt] = fmincon('fun',x0,A,B,Aeq,Aeq,LB,UB,'nonlcon', options)
```

- Scaling parameters can be defined in the main calling function, and passed to ***fmincon***, the constraint function, and the objective function.

```
[xopts,fopts] = fmincon('objfun',x0s,A,B,Aeq,Aeq,LB,UB,'nonlcon',...
                options, alphas, betas, gammas);
[Cs,Ceqs]  =  nonlcon(xs, alphas, betas, gammas);
fs  =  objfun(xs, alphas, betas, gammas);
```

SIMPLE SCALING EXAMPLES

Example:

$$\min_{x_1,x_2} \quad f(x_1,x_2) = (x_1 - 3.67\times 10^{-6})^2 + (x_2 - 3.67\times 10^{-7})^2$$
$$\text{subject to} \qquad x_1 - 2x_2 = 0$$
$$0 \le x_i \le 10^{-4} \quad (i = 1,2)$$

- The answer **<u>with scaling</u>** is **$x = 10^{-6} \times \{3.083, 1.541\}$** and **$f = 1.724\times10^{-12}$**.
- The answer **<u>without scaling</u>** is **$x = 10^{-6} \times \{4.948, 2.474\}$** and **$f = 6.0714 \times 10^{-12}$**.
- The scaling of the design variables: ***$n_{dvi} = 0$***, and ***$x_{i-typical} = 10^{-7}$***.

$$\mathrm{x}_i^{\mathrm{s}} = \alpha_i^s \mathrm{x}_i \qquad (i = 1,\ldots,n_x)$$
$$\alpha_i^s = 10^{n_{dvi}} / x_{i-typical} \qquad (i = 1,\ldots,n_x)$$

SIMPLE SCALING EXAMPLES

- The scaling of the objective functions: $n_{obj} = 0$, and $f_{typical} = 10^{-12}$.

$$\mathtt{f}^{\mathtt{s}}(x) = \beta^{s}\mathtt{f}(x)$$
$$\beta^{s} = 10^{n_{obj}} / f_{typical}$$

- The scaling of the constraints, $n_{eqi} = 0$, and $c_{eq\text{-}nominal} = 10^{-6}$.

$$\mathtt{ceq(x)}_i^s = \gamma_i^s \mathtt{ceq(x)}_i$$
$$\gamma_i^s = 10^{n_{eqi}} / \mathtt{ceq}_{i-nominal}$$
$$\mathtt{ceq(x)}_i \equiv 2x_1 + x_2 = 0; \mathtt{ceq}_{i-nominal} = x_{2-typical}$$

SIMPLE SCALING EXAMPLES

- Add **a constant term** to the objective function.

$$\min_{x_1,x_2} \quad f(x_1,x_2) = 1.111 \times 10^{-8} + (x_1 - 3.67 \times 10^{-6})^2 + (x_2 - 3.67 \times 10^{-7})^2$$

$$\text{subject to} \quad x_1 - 2x_2 = 0$$

$$0 \le x_i \le 10^{-4} \quad (i = 1, 2)$$

SIMPLE SCALING EXAMPLES

- Without scaling:

$$x=10^{-6}\times\{4.948, 2.474\}, f=1.112\times10^{-8}.$$

- Scaling as previous discussion:

  (a) Design variables: $n_{dvi} = 0$, and $x_{i-typical} = 10^{-7}$.

  (b) Objective function: $n_{obj} = 0$, and $f_{typical} = 10^{-8}$.

  (c) Equality constraint: $n_{eqi} = 0$, and $ceq_{nominal} = 10^{-6}$.

  (d) Incorrect answer: $x = 10^{-6} \times \{1.200, 0.600\}$, and $f = 1.1116 \times 10^{-8}$

- **Correct answer:**

  Let $n_{obj} = 2$. $x=10^{-6}\times\{3.0828, 1.541\}$ and $f = 1.111 \times 10^{-8}$.

FINITE DIFFERENCE

- A finite difference derivative: an approximation of a derivative instead of an analytical derivative.

$$\frac{\partial f}{\partial x}(x_0) \approx \frac{\triangle f}{\triangle x}(x_0) = \left\{ \frac{\triangle f_0}{\triangle x_1}, \cdots, \frac{\triangle f_0}{\triangle x_n} \right\}^T$$

- **Forward Difference**

$$\frac{\triangle f_0}{\triangle x_i} = \frac{f(x_0 + \triangle x_i) - f(x_0)}{\triangle x_i}$$

- **Backward Difference**

$$\frac{\triangle f_0}{\triangle x_i} = \frac{f(x_0) - f(x_0 - \triangle x_i)}{\triangle x_i}$$

- **Central Difference**

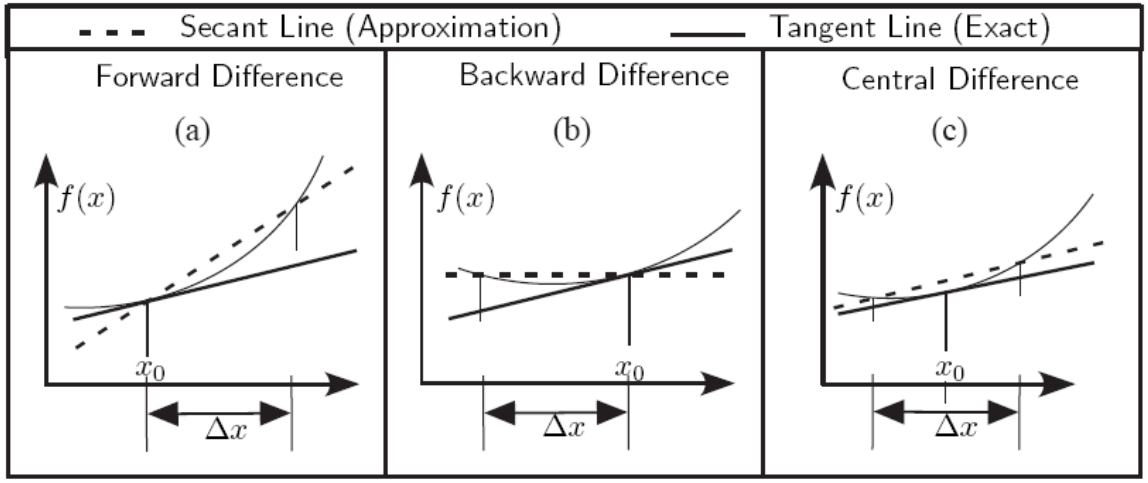
$$\frac{\triangle f_0}{\triangle x_i} = \frac{f(x_0 + \frac{\triangle x_i}{2}) - f(x_0 - \frac{\triangle x_i}{2})}{\triangle x_i}$$

- The error of the finite difference approximation

$$\frac{\partial f}{\partial x_i}(x_0) = \frac{\triangle f_0}{\triangle x_i} \quad + \quad \propto (\triangle x_i)^2 \quad + \quad HOT$$

FINITE DIFFERENCE

- Graphical interpretations of finite differences



- As Δx tends to 0, the secant line converges to the tangent line; and the finite difference converges to the gradient.
- However, excessively small values of Δx pose some numerical difficulties.

FINITE DIFFERENCE

Number of Function Calls:

- When x is a scalar, it requires 2 function calls.
- When x is a n_x-dimension vector.
  - Forward and backward difference: $n_x + 1$ function evaluations.
  - Central difference: $2n_x$ function evaluations.

ACCURACY OF FINITE DIFFERENCE APPROXIMATION

$f(x)$ has n_{sda} significant digits of accuracy.

- As a rule of thumb, the number of digits of accuracy of derivatives drops by half ($n_{sda}/2$).
- The second derivatives drop by another half ($n_{sda}/4$).
- In particular practical cases, the situation could be much worse or much better.

HOW TO CONTROL FINITE DIFFERENCE ACCURACY

1. By adequate scaling of the objective functions, the design variables, the constraints, and the pertinent parameters.
2. By regulating the magnitude of Δx:

   Too large : Too distinct results.

   Too small: The corresponding Δf becomes inaccurate.

Example: $f(x_0)$ = 1.234,567,81234 with 8 significant digits.

Small Δx: $f(x_0 + \Delta x)$ = 1.234,567,82567.

Forward difference, Δf_0 = 0.000,000,01301.

Large Δx: $f(x_0 + \Delta x)$ = 1.234,688,92557.

Forward difference, Δf_0 = 0.000,121,11323.

SELECTING FINITE DIFFERENCE ACCURACY

- For the function fmincon, two options are available: 'forward' (the default), and 'central' (about the center).
- options = optimset('FinDiffType','central')

EXAMPLE OF FINITE DIFFERENCE ACCURACY

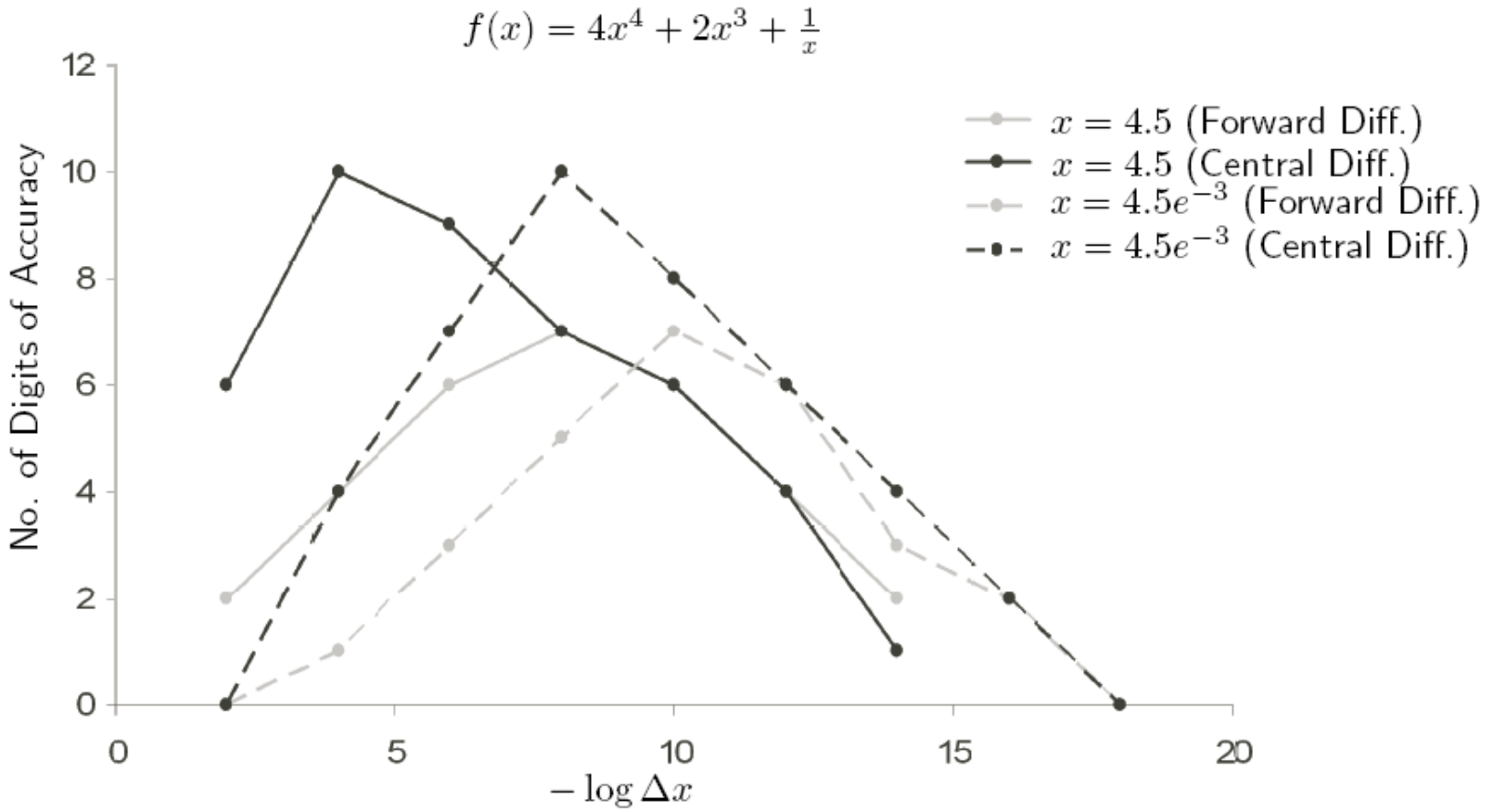
x	Δx	$\frac{\partial f}{\partial x}$	$\frac{\Delta f}{\Delta x}$ (Forward Diff.)		$\frac{\Delta f}{\Delta x}$ (Central Diff.)	
			Value	DoA	Value	DoA
4.5	10^{-2}	1579.4506172	1584.588	2	1579.452	6
	10^{-4}	"	1579.502	4	1579.4506175	10
	10^{-6}	"	1579.451	6	1579.450616	9
	10^{-8}	"	1579.45056	7	1579.4508	7
	10^{-10}	"	1579.451	6	1579.451	6
	10^{-12}	"	1579.792	4	1579.338	4
	10^{-14}	"	1546.141	2	1682.565	1
$4.5e^{-3}$	10^{-2}	−49382.715926	−15325.670	0	210526.316	0
	10^{-4}	"	−48309.179	1	−49388.813	4
	10^{-6}	"	−49371.744	3	−49382.717	7
	10^{-8}	"	−49382.606	5	−49382.715923	10
	10^{-10}	"	−49382.7147	7	−49382.71587	8
	10^{-12}	"	−49382.720	6	−49382.749	6
	10^{-14}	"	−49379.878	3	−49385.562	4
	10^{-16}	"	−49169.557	2	−49737.992	2
	10^{-18}	"	−28421.709	0	−85265.128	0

DoA: Number of Digits of Accuracy

1. For very small or very large values of Δx, the DoA is lower.
2. For mid-range values of Δx, the accuracy is generally highest.
3. Central difference is more accurate than forward difference.
4. Generally, the finite difference yields approximately 6 to 8 digits of accuracy. Less than 1/2 of 16 – the accuracy of f(x).

EXAMPLE OF FINITE DIFFERENCE ACCURACY

- Trends of the accuracy is shown here.
- The horizontal axis depicts $-\log x$ (smaller Δx on the right, and larger on the left).



ANALYTICAL DIFFERENTIATION

Comparison:

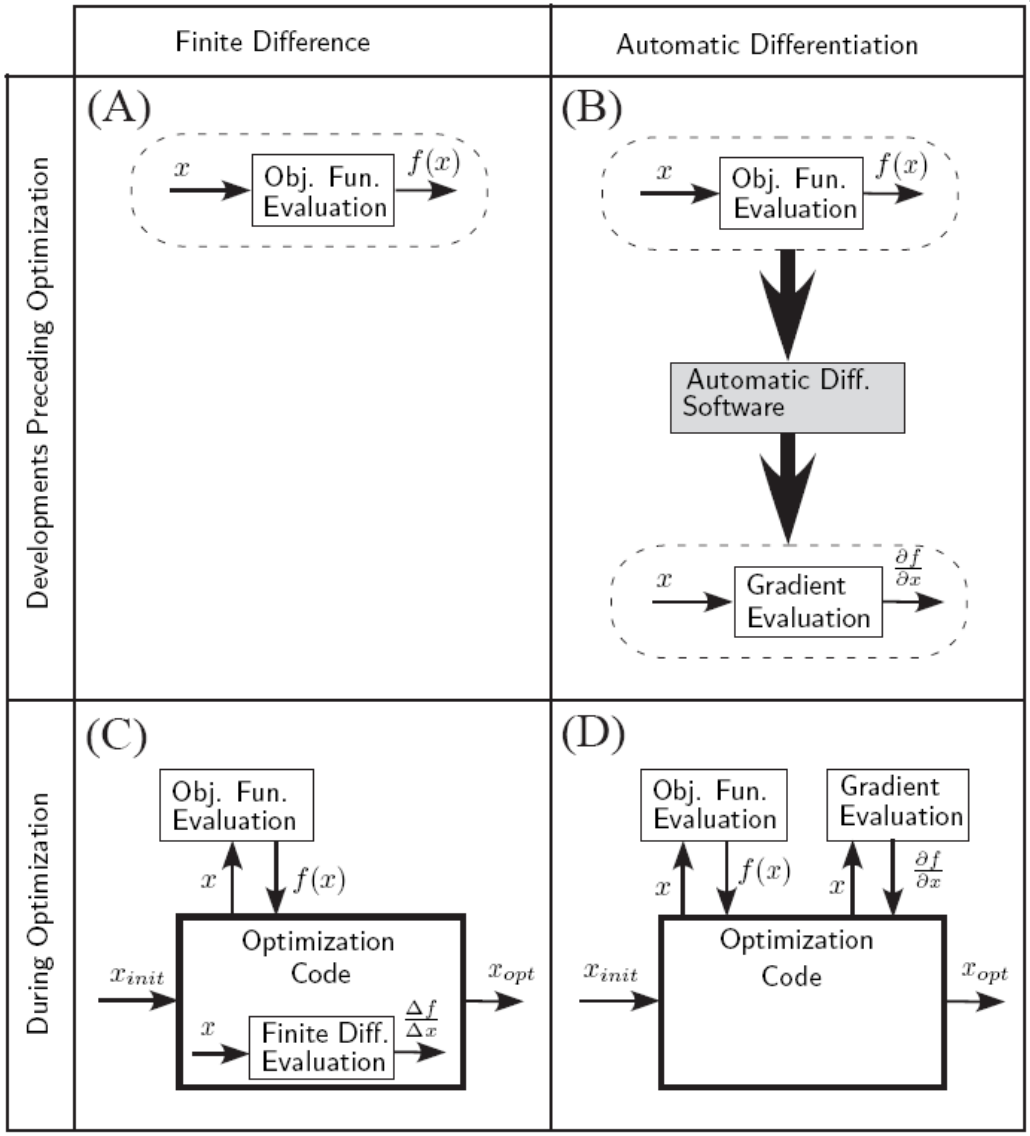
Analytical Differentiation

1. Computationally efficient.
2. Faster convergence and numerically stable.
3. More difficult and time consuming.
4. Implemented by users.

Finite Difference

1. Computationally intensive.
2. Converges slowly. Possible numerical instabilities.
3. Much easier to implement.
4. No user coding.

AUTOMATIC DIFFERENTIATION



OPTIMIZATION TERMINATION CRITERIA

- Optimization codes terminate the convergence process for numerous reasons.
- Some generally indicate a successful outcome.
- Some are likely to indicate non-convergence, or convergence to a non-optimal solution.
- MATLAB optimization settings include:
  - Max number of function evaluations (MaxFunEvals). Default: 100*(Number of Variables).
  - Maximum number of iterations (MaxItr).The default is 400.