IE5268 Theory and Algorithms for Nonlinear Optimization
Week 1: Course Overview & Introduction

Instructor: Guanyi Wang[†]

Department of Industrial Systems Engineering and Management (ISEM)[†],
National University of Singapore (NUS), Singapore.

Teaching Staff & Teaching Assistants

Teaching Instructor:

- Instructor: Dr Guanyi Wang
  - Email: guanyi.w@nus.edu.sg
  - Office: E1-07-16
  - Office-Hour: Monday after the lecture

Teaching Assistant:

- Mr Renyuan Li
  - Email: renyuan.li@u.nus.edu
  - Office-Hour: TBD (Will be updated on Canvas)

Canvas & Assessments

NUS Canvas website:

- Course code: IE 5268
- Announcements
- Files (Will be uploaded to Canvas)
- Recording (Will be uploaded to Canvas)

Assessments:

- Assignments: 40% (2 assignments, 20% each)
- Midterm Take-home Quiz: 30% (Week 7)
- Final Take-home Quiz: 30% (Week 13)

Textbooks

Required:

- **Nocedal, Jorge, and Stephen J. Wright, eds.** *Numerical optimization.* New York, NY: Springer New York, 1999.
- **Y. Nesterov,** *Introductory Lectures on Convex Optimization: A Basic Course,* Kluwer Academic Publishers, Norwell, MA, (2004)

Supplementary/Optional:

- **A. Ruszcynski,** *Nonlinear Optimization,* Princeton University Press, Princeton, NJ (2006)
- **D. P. Bertsekas,** *Nonlinear Programming,* Second Edition, Athena Scientific, Belmont, MA, (1999)

Module Outline

- Week 1: Overview & Introduction
- Week 2 - 6: Unconstrained Optimization
  - Week 2: Convex Analysis I & Optimality Conditions
  - Week 3: Gradient Method
  - Week 4: Proximal Gradient Method
  - Week 5: Stochastic Gradient Method
  - Week 6: Newton's Method
- Week 7: Review for first-half & Take home midterm (2 days)
- Week 8 - 12: Convex Constrained Optimization
  - Week 8: Convex Analysis II
  - Week 9: Constrained Convex Optimization
  - Week 10: Duality
  - Week 11: Equality Constrained Minimization
  - Week 12: Decomposition method
- Week 13: Review for second-half & Take home final (2 days)

Course Learning Outcomes

What I expect you to achieve after taking this course ...

- **(Basic Results)** Have a comprehensive knowledge to the basic results for different optimization methods
- **(Optimality Conditions)** Understand the necessary and sufficient optimality conditions for unconstrained and constrained optimization problems
- **(Duality)** Understand the duality theory for optimization problems
- **(Large-scale Opt)** Familiar with methods for large-scale optimization problems
- **(Make Your Choice)** Able to pick appropriate optimization methods based on given conditions/settings

Pre-requirements: Topics that I expect you have already known

Will be frequently used in this course.
If you are not familiar with the following knowledge,
this course will take you additional effort ...

Pre-requirements

- **MA3210 Mathematical Analysis II**
  - derivatives, gradients, Jacobians, Hessians
  - Taylor's expansion
  - sequences (subsequences, boundedness, accumulation points, etc.)
  - continuity and limits
- **MA4230 Matrix Computation (Linear Algebra)**
  - vectors and vector norms, matrices and matrix norms
  - matrix properties, e.g., symmetric, positive (semi) definite, (non)singular, etc.
  - determinants, eigenvalues, and eigenvectors
  - matrix factorizations
  - condition number of a matrix
- **IE4210 Operations Research II** (Recommended)
  - convex and concave function
  - format of linear/nonlinear programming

Introduction to Nonlinear Unconstrained Optimization

Brief Introduction – 1

- **Basic Problem:**

$$\min_{x \in \mathbb{R}^n} \quad f(x)$$

  - decision variable $x \in \mathbb{R}^n$
  - objective function $f : \mathbb{R}^n \to \mathbb{R}$
  - maximizing can be transferred into minimization, i.e., $\max_x \widehat{f}(x) \;=\; \min_x -\widehat{f}(x)$
- **Definition (global minimizer):** The vector x^* is a global minimizer if $f(x^*) \leq f(x)$ such that $x \in \mathbb{R}^n$.
- **Definition (local minimizer):** For some given $\epsilon > 0$, the vector x^* is a local minimizer if $f(x^*) \leq f(x)$ for all $x \in \|x - x^*\| \leq \epsilon$.
- **Definition (strict local minimizer):** For some given $\epsilon > 0$, the vector x^* is a strict local minimizer if $f(x^*) < f(x)$ for all $x \neq x^*$ such that $x \in \|x - x^*\| \leq \epsilon$.

Brief Introduction – 2

- **Recall the basic problem:**

$$\min_{x \in \mathbb{R}^n} \quad f(x)$$

- **Notation** (without specific comments, not always hold in this course or references)
  - **Gradient:** $g(x) := \nabla f(x) \in \mathbb{R}^n$
  - **Hessian:** $H(x) := \nabla^2 f(x) \in \mathbb{R}^{n \times n}$
- **Assumptions:** we will always assume that f is a continuous function
- **Going to see the following three types of f:**
  - f is once or twice continuously differentiable (smooth optimization)
  - f is non-differentiable but with structure (structured non-smooth optimization)
  - derivatives of f are too expensive or unavailable (derivative free optimization)

Examples for above three types of objective f

Data Fitting Example – Asteroid Ceres

In January 1801: Asteroid Ceres is discovered, but in Autumn 1801 it "disappeared". Gauss considers an elliptic orbit instead of a circular orbit

circular orbit: $x^2 + y^2 = r \quad \text{for some } r > 0$

elliptic orbit: $\alpha x^2 + \beta y^2 + \gamma xy = 1 \quad \text{for some } \alpha, \beta, \gamma$

How did Gauss do it?

- used a collection of N previous location measurements $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$.
- find the "best" α, β, γ that satisfy

$$(\alpha^*, \beta^*, \gamma^*) := \arg\min_{\alpha,\beta,\gamma} \frac{1}{N} \sum_{i=1}^{N} \left(\alpha x_i^2 + \beta y_i^2 + \gamma x_i y_i - 1\right)^2,$$

  which is nonlinear and twice continuously differentiable
- looked for Ceres along the ellipse defined by $\alpha^* x^2 + \beta^* y^2 + \gamma^* xy = 1$

Speech recognition: Multi-Class Regression – 1

Consider the following problem ...

- Number of classes $N_c \approx 100$ (basic units of sound)
- Number of features $N_f \approx 10^4$ (coefficients in the mathematical representation of a digital sample of sound)
- Number of parameters $N_p \approx 10^6$ (#classes $\times$ #features)
- Number of data points $N_d \approx 10^10$ and growing (size of data)
- Compute w^* as solution to

$$\max_{w \in \mathbb{R}^{N_p}} f(w) + \lambda \|w\|_1 \quad \lambda > 0 \text{ is a sparsity parameter}$$

where

$$f(w) := \sum_{i=1}^{N_d} \log \left(\frac{\exp(w_{y_i}^\top x_i)}{\sum_{j=1}^{N_c} \exp(w_j^\top x_i)} \right)$$

Speech recognition: Multi-Class Regression – 2

- Predicted probability of new input $\widehat{x}$ being in class $k \in [N_c]$ is

$$\mathbb{P}\left(y = k \mid x = \widehat{x}\right) = \frac{\exp((w^*)_{y_i}^\top \widehat{x})}{\sum_{j=1}^{N_c} \exp((w^*)_j^\top \widehat{x})}$$

- Major challenges
  - $f(w) + \lambda \|w\|_1$ is nonlinear
  - $\|w\|_1$ is non-smooth when $w_i = 0$ for some i (structured non-smooth)
  - Computing $\nabla f(w)$ is very expensive! Must sum up 10 billion gradients.

Tuning of algorithmic parameters

This is widely considered in machine learning ...

- Almost all numerical codes use various parameters
- The programmer must choose default values for these parameters
- The values chosen for the parameters are often crucial for excellent performance
- Sensible values for the parameters may be obtained by (approximately) solving

$$\min_{x \in \mathbb{R}^n} \quad f(x) \;:=\; \text{CPU-time}(x;\text{solver}) \;\text{ or }\; \text{ITER}(x;\text{solver})$$

  - n represents the number of parameters
  - p represents the vector of problem parameters
  - CPU-time represents the time required to solve a collection of test problems for a given value of the parameter vector p
  - ITER represents the number of iterates required to solve a collection of test problems for a given value of the parameter vector p
- Computing the gradient of f is difficult (it is likely noisy and possibly non-differentiable)

Quick Summary of Nonlinear Unconstrained Optimization

Summary – 1

Unconstrained optimization problems may

- have a linear or nonlinear objective function
- be convex or nonconvex
- have an objective function that is twice continuously differentiable, once continuously differentiable, structurally non-smooth, non-smooth
- contain continuous and/or discrete variables
- vary in size
  - small size $\approx 1 \sim 10^2$ variables
  - median size $\approx 10^2 \sim 10^3$ variables
  - large size $\approx 10^3 \sim 10^6$ variables
  - very large size $\geq 10^6$ variables

Summary – 2

We may be interested in

- a local solution or global solution
- the minimum value of the objective function and/or the minimizer
- finding multiple distinct minimizers
- the lowest value of the objective given time constraints or limits on the number of allowed evaluations of the objective function

Background and Basics for Optimization: Computer Arithmetic

Computer arithmetic – Floating-point (real) Numbers 1

Modern computers store real numbers as

$$x = \pm \left(d_0 + \frac{d_1}{\beta_1} + \cdots + \frac{d_{p-1}}{\beta_{p-1}} \right) \beta^E.$$

- base: β (e.g., 2)
- significand precision: p (e.g., 24 (SP), 53 (DP))
- exponent: $E \in [L, U]$ (e.g., [-126, 127] (SP), [-1022, 1023] (DP))
- $d_i \in \{0, \ldots, \beta - 1\}$ for $i = 0, \ldots, p-1$
- the floating-point system is completely characterized by the four integers β, p, L, U
- mantissa $d_0 d_1 \cdots d_{p-1}$
- fraction $d_1 \cdots d_{p-1}$
- floating-point system is normalized if d_0 is always nonzero unless the number represented is zero
- we will only consider normalized floating-point systems

Computer arithmetic – Floating-point (real) Numbers 2

Consider the following example with

$$\beta = 10, \quad p = 4, \quad L = -99, \quad U = 99$$

- Some numbers

$$1 = 1.000 \times 10^{00}$$
$$34.67 = 3.467 \times 10^{01}$$
$$0.0346 = 3.460 \times 10^{-02}$$

- smallest positive number

$$1.000 \times 10^{-99}$$

- largest positive number

$$9.999 \times 10^{99}$$

Computer arithmetic – Floating-point (real) Numbers 3

Facts about floating-point systems

- It is finite, i.e, not all real numbers can be stored
- Machine numbers are those real numbers that may be exactly represented
- Floating-point numbers are equally spaced only between successive powers of β
- Total number of normalized floating-point numbers is

$$2(\beta - 1)\beta^{p-1}(U - L + 1) + 1$$

- Smallest positive number: $\mathrm{UFL} = \beta^{L}$ underflow level
  - numbers smaller than UFL stored as zero
  - often not serious, because zero is a good approximation
- Largest number: $\mathrm{OFL} = \beta^{U}(1 - \beta^{-p})$ overflow level
  - numbers larger than OFL may not be stored
  - serious problem, compilers typically terminate

Computer arithmetic – Floating-point (real) Numbers 4

Rounding

When a real number x is not exactly representable, it is approximated by a "nearby" floating-point number $\mathrm{fl}(x)$. This process is called rounding and the error that is introduced is called rounding error.

- Common rounding strategies:
  - chopping: $\mathrm{fl}(x)$ is obtained by truncating the expansion of x after d_{p-1}. Also called round-to-zero. Widely-used in neural network training.
  - round-to-nearest: $\mathrm{fl}(x)$ is the closest floating-point number to x. In case of a tie, use the floating-point number whose last stored digit is even. Also called round-to-even
- Assume round-to-nearest in our course since it is the most accurate and the default rounding rule on machines

Question: How bad can the rounding error be?
Answer: Involves the concept of machine precision.

Computer arithmetic – Floating-point (real) Numbers 5

Machine precision assuming round-to-nearest

The machine precision is defined as $\epsilon := \frac{1}{2}\beta^{1-p}$, which bounds the relative error in storing a floating-point number

$$\frac{|\mathrm{fl}(x) - x|}{|x|} \leq \epsilon.$$

The machine precision ϵ is equal to

- the liminf such that $\mathrm{fl}(1+\epsilon) > 1$
- the the largest number such that $\mathrm{fl}(1+\epsilon) = 1$
- half the distance between 1 and the nearest floating-point number

For example, using round-to-nearest, $p = 4$ and $\beta = 10$, we have

$$1.000 + 0.0005 = 1.0005 \overset{\text{comp}}{=} 1$$

$$1.000 + 0.00051 = 1.00051 \overset{\text{comp}}{=} 1.001$$

with $\epsilon = \frac{1}{2} \times 10^{1-4} = 0.0005$.

Computer arithmetic – Floating-point (real) Numbers 6

Exceptional values in the floating-point system

Institute of Electrical and Electronics Engineers (IEEE) standard allows for the following exceptional values:

- Inf: represents "infinity" and results from dividing a finite number by zero
- NaN: stands for "not a number" and results from undefined or not well-defined operations, e.g., $0/0,\ 0 \times \infty,\ \infty/\infty$.

Computer arithmetic – Floating-point (real) Numbers 7

Floating-point addition & multiplication: Let $x = 4.452 \times 10^{02}$ and $y = 6.436 \times 10^{-01}$

- **Addition of two floating-point numbers (similar for subtraction)**
  - shift so that exponents are the same, add, then re-normalize
  - for example

$$\begin{aligned} x + y &= (4.452 \times 10^{02}) + (6.436 \times 10^{-01}) \\ &= (4.452 \times 10^{02}) + (0.006436 \times 10^{02}) \\ &= 4.458436 \times 10^{02} \overset{\text{comp}}{=} 4.458 \times 10^{02} \end{aligned}$$

  - generally, trailing digits of smaller (in magnitude) number are lost
- **Multiplication of two floating-point numbers (similar for division)**
  - exponents are summed and mantissas multiplied
  - product of two p digit mantissas is generally $2p$ digits (must round)
  - for example

$$\begin{aligned} x \times y &= (4.452 \times 10^{02}) \times (6.436 \times 10^{-01}) \\ &= 28.653072 \times 10^{01} = 2.8653072 \times 10^{02} \overset{\text{comp}}{=} 2.865 \times 10^{02} \end{aligned}$$

Background and Basics for Optimization: Linear systems, norms, and condition numbers

Linear System 1

The problem of interest: Given a data matrix $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$, solve the linear system $Ax = b$.

- We use a_{ij} to denote the component of A in row i and column j
- Can consider questions of existence and uniqueness of solutions
- What about conditioning (sensitivity of the solution)
- If A is nonsingular, A^{-1} exists and the unique solution is $x = A^{-1}b$

Consider the following example:

$$\left.\begin{aligned} x_1 + 3x_2 &= 5 \\ 2x_1 + 7x_2 &= 3 \end{aligned}\right\} \quad \Rightarrow \quad Ax = b$$

where

$$n = 2, \quad A = \begin{pmatrix} 1 & 3 \\ 2 & 7 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad b = \begin{pmatrix} 5 \\ 3 \end{pmatrix}$$

Is the solution unique?

Linear System 2

Nonsingular case

A square matrix $A \in \mathbb{R}^{n \times n}$ is said to be nonsingular if any of the following equivalent conditions are satisfied:

- The inverse matrix A^{-1} exists
- $\det(A) \neq 0$
- $\operatorname{rank}(A) = n$
- $Az = 0 \;\Rightarrow\; z = 0$
- $z \neq 0 \;\Rightarrow\; Az \neq 0$

If A is nonsingular, then $Ax = b$ has a unique solution.

Linear System 3

Singular case

A square matrix $A \in \mathbb{R}^{n \times n}$ is said to be singular, then

- if $b \in \text{span}(A)$, then infinitely many solutions exist
- if $b \notin \text{span}(A)$, then no solutions exist

Consider the following example:

$$\underbrace{\begin{pmatrix} 1 & 3 \\ 2 & 6 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}}_{b}$$

- $\det(A) = 0 \quad \Rightarrow \quad A$ is singular
- If $b = \begin{pmatrix} 4 \\ 8 \end{pmatrix} \quad \Rightarrow \quad x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad x = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$, or ... infinitely many solutions.
- If $b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, then no solutions.

Linear System 4

"Known" material for square A

- general A: solve $Ax = b$ using $A = LU$ factorization by Gaussian elimination method
- positive-definite A: solve $Ax = b$ using $A = LL^\top$ by Cholesky factorization

New material for square A

- Conditioning: when is the solution x to the system $Ax = b$ sensitive to the input data A and b?
- To understand conditioning, we will introduce the condition number of a matrix A

$$\mathrm{cond}(A) := \|A\|\|A^{-1}\|$$

- This requires us to understand matrix norms $\|A\|$
- Which requires us to understand vector norms

Norms 1

Examples of vector norm

In linear algebra, we know many vector norms

- $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$ ℓ_2-norm
- $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$ ℓ_1-norm
- $\|x\|_\infty = \max_{i=1}^n |x_i|$ ℓ_∞-norm

For example, let

$$x = \begin{pmatrix} -12 \\ -3 \\ 4 \end{pmatrix}, \quad \|x\|_2 = 13, \quad \|x\|_1 = 19, \quad \|x\|_\infty = 12$$

Norms 2

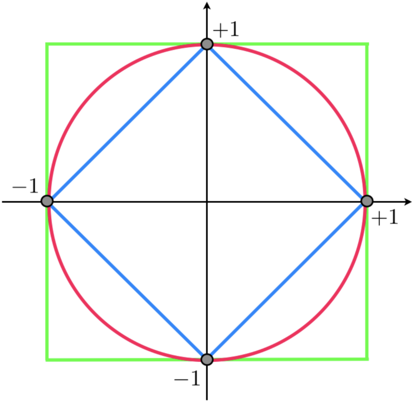
Sometimes a specific norm may be better than another ...

Suppose we have accumulated data as the result of a carefully designed experiment, and then obtained a model of the data.

If we store the error of each data point (of right-hand-side figure) in the vector $x = (10^{-2}, 10^{-1}, \dots, \mathbf{1}, \dots, 10^{-1})$. We get $\|x\|_\infty = \mathbf{1}$ by outlier. Maybe better to consider $\|x\|_2/n$?

Norms 3

Geometry of different types of ℓ_p-norms



Some results:

- $\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1$
- $\|x\|_1 \leq \sqrt{n}\|x\|_2$
- $\|x\|_2 \leq \sqrt{n}\|x\|_\infty$
- $\|x\|_1 \leq n\|x\|_\infty$

Norms 4

Vector norm

A vector norm is any real-valued function $\|\cdot\|$ of a vector that satisfies the following properties:

- if $x \neq 0$, then $\|x\| > 0$
- $\|\alpha x\| = |\alpha|\|x\|$ for all $\alpha \in \mathbb{R}$
- $\|x + y\| \leq \|x\| + \|y\|$ (triangle inequality)

- Using the above properties, it may be shown that
  - $\|x\| = 0$ if and only if $x = 0$
  - $\|x\| - \|y\| \leq |\|x\| - \|y\|| \leq \|x - y\|$ (reverse triangle inequality)
- We have already seen some examples $\ell_1, \ell_2, \ell_\infty$ norms satisfy the definition of vector norm.

Norms 5

Induced matrix norm

Given a vector norm $\|x\|$, we define the induced matrix norm of A as

$$\|A\| := \max_{\|x\|=1} \|Ax\|$$

- Measures the maximum amount of "elongation" resulting from multiplication by A
- It can be shown that
  - $\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}|$ maximum absolute column sum
  - $\|A\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}|$ maximum absolute row sum

Consider the following example:

$$A = \begin{pmatrix} -7 & 4 & 3 & 1 \\ 8 & -5 & 6 & 0 \\ -1 & -3 & 7 & 4 \\ 5 & 0 & 0 & -5 \end{pmatrix} \quad \text{with} \quad \|A\|_1 = 21, \quad \|A\|_\infty = 19$$

Norms 6

Matrix norm

A matrix norm is any real-valued function $\|\cdot\|$ of a matrix that satisfies the following properties:

- if $A \neq 0$, then $\|A\| > 0$
- $\|\alpha A\| = |\alpha|\|A\|$ for all $\alpha \in \mathbb{R}$
- $\|A + B\| \leq \|A\| + \|B\|$ (triangle inequality)

- Using the above properties, it may be shown that
  - $\|A\| = 0$ if and only if $A = 0$
  - $\|A\| - \|B\| \leq |\|A\| - \|B\|| \leq \|A - B\|$ (reverse triangle inequality)
- We have already seen some examples $\|A\|_1, \|A\|_\infty$ norms satisfy the definition of vector norm.
- Induced matrix norms (not all norms) are consistent, i.e., satisfy
  - $\|AB\| \leq \|A\|\|B\|$
  - $\|Ax\| \leq \|A\|\|x\|$ for all x

Condition Number 1

Condition number

We define the condition number of a square matrix A as

$$\operatorname{cond}(A) := \begin{cases} \|A\|\|A^{-1}\| & \text{if } A \text{ is non-singular} \\ \infty & \text{if } A \text{ is singular} \end{cases}$$

- large condition number $\Rightarrow$ A is nearly singular
- geometric interpretation: the condition number is the ratio of the largest stretching over the smallest shrinking caused by multiplication by A
- the residual $r := b - A\widehat{x}$ is not a reliable measure of accuracy
- for well-conditioned problems, the relative residual is reliable:

$$\frac{\|b - A\widehat{x}\|}{\|\widehat{x}\|\|A\|}$$

- fact: backward stable algorithms produce small relative residuals

Condition Number 2

Properties of the condition number: If the condition number is defined by any induced matrix norm, then

- $\text{cond}(I) = 1$
- $\text{cond}(A) \geq 1$
- $\text{cond}(\alpha A) = \text{cond}(A)$ for all $\alpha \neq 0$
- If D is a diagonal matrix, then

$$\text{cond}(D) = \frac{\max_{i=1}^{n} |d_{ii}|}{\min_{i=1}^{n} |d_{ii}|}$$

Consider the following example:

$$D = \begin{pmatrix} 2 & 0 & 0 \\ 0 & -40 & 0 \\ 0 & 0 & 0.01 \end{pmatrix} \quad \Rightarrow \quad \text{cond}(D) = \frac{|-40|}{|0.01|} = 4000$$

Condition Number 3

Computing the condition number

- Computing $\|A\|$ is computationally cheap
- Computing A^{-1} is very computationally expensive
- It is more expensive to compute A^{-1} than it is to solve $Ax = b$
- Some software cheaply estimates cond(A) while solving $Ax = b$
  - LINPACK $\rightarrow$ sgeco
  - LAPACK $\rightarrow$ sgecon
  - NAG $\rightarrow$ f07agf
  - Matlab $\rightarrow$ condest
  - Python $\rightarrow$ numpy.linalg.norm

Accuracy analysis 1

- Suppose we are given A, b and a perturbed right-hand-side $\widehat{b} = b + \Delta b$
- Let x and $\widehat{x}$ satisfy

$$\begin{aligned} Ax &= b \quad \Rightarrow \quad \|b\| = \|Ax\| \leq \|A\|\|x\| \ \ \text{(consistency)} \\ A\widehat{x} &= \widehat{b} \end{aligned}$$

- Define $\Delta x := \widehat{x} - x$.

$$A\Delta x = A(\widehat{x} - x) = A\widehat{x} - Ax = \widehat{b} - b = \Delta b \quad \Rightarrow \quad \Delta x = A^{-1}\Delta b$$

- Using the previous equality, the consistency property, and consistency property,

$$\frac{\|\Delta x\|}{\|x\|} = \frac{\|A^{-1}\Delta b\|}{\|x\|} \leq \frac{\|A^{-1}\|\|\Delta b\|}{\|x\|} \leq \frac{\|A\|\|A^{-1}\|\|\Delta b\|}{\|b\|} \leq \text{cond}(A)\frac{\|\Delta b\|}{\|b\|}$$

Accuracy analysis 2

We proved the following perturbation result in the previous slide.

Theorem (Error bound for linear systems)

If A is nonsingular, $Ax = b$ and $A\widehat{x} = \widehat{b}$, then

$$\frac{\|\Delta x\|}{\|x\|} \leq \mathrm{cond}(A) \frac{\|\Delta b\|}{\|b\|}$$

A similar analysis shows the following.

Theorem (Error bound for linear systems)

If A is nonsingular, $Ax = b$ and $A\widehat{x} = \widehat{b}$, then

$$\frac{\|\Delta x\|}{\|x\|} \leq \mathrm{cond}(A) \frac{\|\widehat{A} - A\|}{\|A\|}$$

- Similar result holds when A and b are perturbed simultaneously
- What does this mean in terms of computer representation?

Accuracy analysis 3

What does this mean in terms of computer representation?

- We give the computer A and b and want to find x such that $Ax = b$. We assume that A is exactly representable, but that b is not.
- Define $\widehat{b} := \mathrm{fl}(b)$ so that $\widehat{b}$ satisfies

$$\frac{\|\widehat{b} - b\|}{\|b\|} = \frac{\|\mathrm{fl}(b) - b\|}{\|b\|} \leq \epsilon$$

- We solve $A\widehat{x} = \widehat{b}$
- From result on previous slide we know that

$$\frac{\|\Delta x\|}{\|x\|} \leq \mathrm{cond}(A)\frac{\|\Delta b\|}{\|b\|} \leq \mathrm{cond}(A)\epsilon$$