Approximation and Errors
Well-Posed Problems

- Problem is well-posed if solution
  - exits
  - is unique
  - depends continuously on problem data

Otherwise, problem is ill-posed
Remedy to ill-posed problem

- Replace difficult problem by easier one having same or close enough solution
  - nonlinear $\rightarrow$ linear
  - infinite $\rightarrow$ finite
  - differential $\rightarrow$ algebraic
  - complicate $\rightarrow$ simple
- Solution obtained may only approximate that of the original problem
Sources of Approximation

- Before computation
  - Modeling
  - Empirical measurements
  - Previous computations

- During computation
  - Truncation or discretization
  - Rounding

- Accuracy of final result reflects all these
- Uncertainty in input may be amplified by problem
- Perturbations during computation may be amplified by algorithm

From Michael T. Heath’s slide
Example: Approximations

- Computing surface area of Earth using formula \( V = 4\pi r^2 \) involves several approximations

- Earth is modeled as sphere, idealizing its true shape
- Value for radius is based on empirical measurements and previous computations
- Value for \( \pi \) requires truncating infinite process
- Values for input data and results of arithmetic operations are rounded in computer

From Michael T. Heath’s slide
Error Definition

- Absolute error: $|\text{approximate value} - \text{true value}|$
- Relative error: $(\text{absolute error})/(\text{true value})$
- True value is usually unknown, so we estimate or bound error rather than compute it exactly.
- Relative error often taken relative to approximate value rather than (unknown) true value.

From Michael T. Heath’s slide
Truncation Error

- Difference between true result and result produced by given algorithm using exact arithmetic

- Due to approximations such as truncating infinite series or terminating iterative sequence before convergence

\[
e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \sum_{n=4}^{\infty} \frac{x^n}{n!}
\]

\[
e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!}
\]

Truncation error
\[ e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \sum_{n=4}^{\infty} \frac{x^n}{n!} \]

\[ e^x \approx 1 \]
\[ e^{0.5} \approx 1 \]

\[ e^x \approx 1 + \frac{x}{1!} \]
\[ e^{0.5} \approx 1.5 \]

\[ \varepsilon = \frac{\text{current approximation} - \text{previous approximation}}{\text{current approximation}} \]

\[ \varepsilon = \frac{1.5 - 1}{1.5} 100\% = 33.3\% \]
\[ e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \sum_{n=4}^{\infty} \frac{x^n}{n!} \]

\[ e^x \approx 1 \]
\[ e^x \approx 1 + \frac{x}{1!} \]
\[ e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} \]

\[ e^{0.5} \approx 1 \]
\[ e^{0.5} \approx 1.5 \]
\[ e^{0.5} \approx 1.625 \]

\[ \varepsilon = \frac{1.625 - 1}{1.625} \times 100\% = 7.69\% \]
\[ e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \sum_{n=4}^{\infty} \frac{x^n}{n!} \]

\[ e^x \approx 1 \]
\[ e^{0.5} \approx 1 \]
\[ e^x \approx 1 + \frac{x}{1!} \]
\[ e^{0.5} \approx 1.5 \]
\[ e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} \]
\[ e^{0.5} \approx 1.625 \]
\[ e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} \]
\[ e^{0.5} \approx 1.645833333 \]

\[ \varepsilon = 33.3\% \]
\[ \varepsilon = 7.69\% \]
\[ \varepsilon = 1.27\% \]
Round-off Error

- Difference between result produced by given algorithm using exact arithmetic and result produced by same algorithm using limited precision arithmetic
- Due to inexact representation of real numbers and arithmetic operations upon them
- Computational error = truncation error + round-off error
Propagated Error

- An error in the succeeding steps of a process due to an occurrence of an earlier error
Forward and Backward Error

- Suppose we want to compute $y = f(x)$, where $f$ is a real value function, but obtain approximate value $y_{calc}$.

- **Forward error**
  
  $E_{fwd} = y_{calc} - y_{exact}$

- **Backward error**
  
  $E_{backw} = x_{calc} - x$

\[ y_{exact} = f(x) \]

\[ x_{calc} = f^{-1}(y_{calc}) \]
Forward and Backward Error

- Suppose we want to compute $y = f(x)$, where $f$ is a real value function, but obtain approximate value $\hat{y}$.

- Forward error: $\Delta y = \hat{y} - y$

- Backward error: $\Delta x = \hat{x} - x$

![Diagram showing forward and backward error calculation]
Backward Error Analysis

- Idea: approximate solution is exact solution to modified problem
- How much must original problem change to give result actually obtained?
- How much data error in input would explain all error in computed result?
- Approximate solution is good if it is exact solution to near problem

From Michael T. Heath’s slide
Example: Backward Error Analysis

- Approximating cosine function by truncating Taylor series after two terms gives
  \[ \hat{y} = \hat{f}(x) = 1 - \frac{x^2}{2} \]

- Forward error is given by
  \[ \hat{y} - y = \hat{f}(x) - f(x) = 1 - \frac{x^2}{2} - \cos(x) \]

- Backward error is given by \( \hat{x} - x \), where
  \[ \hat{x} = f^{-1}(\hat{y}) = \arccos(\hat{y}) \]
Example (cont.)

For $x=1$,

\[
y = f(1) = \cos(1) \approx 0.5403
\]
\[
\hat{y} = f(1) = 1 - 1^2/2 = 0.5
\]
\[
\hat{x} = \arccos(\hat{y}) = \arccos(0.5) \approx 1.0472
\]

- **Forward error:** $\Delta y = \hat{y} - y \approx 0.5 - 0.5403 = -0.0403$
- **Backward error:** $\Delta x = \hat{x} - x \approx 1.0472 - 1 = 0.0472$
Sensitivity and Conditioning

- Problem is insensitive or well-conditioned, if relative change in input causes similar change in solution.
- Problem is sensitive or ill-conditioned, if relative change in solution can be much larger than that in input.
- Condition number

\[
\text{cond} = \frac{|\text{relative change in solution}|}{|\text{relative change in input data}|} = \frac{[f(\hat{x}) - f(x)]/f(x)}{|(\hat{x} - x)/x|}
\]
Sensitivity and Conditioning (cont.)

- Problem is sensitive or ill-conditioned, if $\text{cond} >> 1$

- A well-posed problem can be ill-conditioned

- Computational algorithm should not make sensitivity worse
Condition Number

- Condition number is amplification factor relating relative forward error to relative backward error

\[
\text{cond} = \left| \frac{[f(\hat{x}) - f(x)]/f(x)}{(\hat{x} - x)/x} \right| = \frac{|\Delta y/y|}{|\Delta x/x|}
\]

- Condition number is usually not known exactly and may vary with input, so rough estimate or upper bound is used
Example: Evaluating Function

- Evaluating function $f$ for approximate input $\hat{x} = x + h$

\[
\text{cond} = \frac{|\text{relative change in solution}|}{|\text{relative change in input data}|} = \frac{\left| \frac{f(\hat{x}) - f(x)}{f(x)} \right|}{\left| \frac{(\hat{x} - x)}{x} \right|}
\]

\[
\hat{x} = x + h
\]

\[
f(x + h) \approx f(x) + hf'(x)
\]
Example: Sensitivity

- Tangent function is sensitive for arguments near $\pi/2$

\[
\begin{align*}
\tan(1.57078) & \approx 6.58058 \times 10^4 \\
\tan(1.57079) & \approx 1.58058 \times 10^5
\end{align*}
\]

- For $x = 1.57079$, \text{cond} \approx 2.48275 \times 10^5
Stability

- Algorithm is stable if result produced is relatively insensitive to perturbations during computation

- Stability of algorithm is analogous to conditioning of problems

- From viewpoint of backward error analysis, algorithm is stable if result produced is exact solution to nearby problem

- For stable algorithm, effect of computational error is worse than effect of small data error in input
Accuracy

- Accuracy: closeness of computed solution to true solution of problem

- Stability alone does not guarantee accurate results

- Accuracy depends on conditioning of problem as well as the stability of algorithm

- Inaccuracy can result from applying
  - stable algorithm to ill-conditioned problem
  - unstable algorithm to well-conditioned problem

- Applying stable algorithm to well-conditioned problem yields accurate solution

From Michael T. Heath’s slide
Floating-Point Arithmetic

- Floating-point number is represented by
  - sign
  - fraction (mantissa)
  - exponent

\[
x = \pm \left(1 + \frac{d_1}{2} + \frac{d_2}{2^2} + \cdots + \frac{d_{p-1}}{2^{p-1}}\right)2^E
\]

where \(0 \leq d_i \leq 1, i = 1, \ldots, p - 1\), and \(L \leq E \leq U\)
Machine Numbers

- Not all real numbers exactly representable; those that are are called machine numbers.
- Machine numbers are unequally spaced.
- If a real number is not exactly representable, then it is approximated by a nearby machine number → causing round-off error.
Machine Precision

- Accuracy of floating-point system characterized by unit round-off (or machine precision or machine epsilon)
- Smallest number $\varepsilon$ such that
  \[
  \text{float}(1 + \varepsilon) > 1
  \]
- If $\varepsilon < \varepsilon$
  \[
  (1 + \varepsilon) + \varepsilon =
  \]
  \[
  1 + (\varepsilon + \varepsilon) =
  \]
Machine Precision

- Accuracy of floating-point system characterized by unit round-off (or machine precision or machine epsilon)
- Smallest number $\varepsilon$ such that
  \[ \text{float}(1 + \varepsilon) > 1 \]
- If $\varepsilon < \text{eps}$
  \[ (1 + \varepsilon) + \varepsilon = 1 \]
  \[ 1 + (\varepsilon + \varepsilon) = \]
Machine Precision

- Accuracy of floating-point system characterized by unit round-off (or machine precision or machine epsilon)
- Smallest number ε such that
  \[ \text{float}(1 + \text{eps}) > 1 \]
- If ε < eps
  \[ (1 + \varepsilon) + \varepsilon = 1 \]
  \[ 1 + (\varepsilon + \varepsilon) > 1 \]
Summary

- Well-posed problem
- Computational error
  - truncation error
  - round-off error
- Sensitivity (conditioning) of problem
  - Condition number
- Stability of algorithm