Agenda

- Inexact computation
- Bad pivoting
- Rooting of polynomials
Numerical stability

- Computer arithmetic is inexact (finite memory)
- Issues arise from inexact computations
- Interested in robust and stable algorithms
Number representation

Floating point representation

\[ x = \pm .d_1 \ldots d_s \ldots 10^e \]

e.g.

\[ 10/3 = \pm 0.33\ldots 3\ldots 10^1 \]

Representation in base \( b \)

\[ x = .d_1 \ldots d_s \ldots b^e \]

Other common representation: binary representation where \( b = 2 \)

10/3 in base 2?
IEEE floating point numbers with base 2

Used in almost every computer

\[ x = \pm ( \cdot d_1 \ldots d_s )_2 \cdot 2^e \]

- \( \cdot d_1 \ldots d_s \) is the mantissa \( (d_i \in \{0, 1\}, d_1 = 1 \text{ if } x \neq 0) \)
- \( s \) is the mantissa length
- \( e \) is the exponent \( e_{\text{min}} \leq e \leq e_{\text{max}} \)

Interpretation

\[ x = (d_1 2^{-1} + d_2 2^{-2} + \ldots d_s 2^{-s}) \cdot 2^e \]

- Finite set of unequispaced numbers
- Smallest positive number

\[ x_{\text{min}} = 2^{e_{\text{min}}-1} \]

- Largest positive number

\[ x_{\text{max}} = (1 - 2^{-s})2^{e_{\text{max}}} \]
IEEE floating point standard

Single precision

\[ s = 24, \quad e_{\min} = -125, \quad e_{\max} = 128 \]

Requires 32 bits: 1 sign bit + 23 bits for mantissa + 8 bits for exponent

Double precision

\[ s = 53, \quad e_{\min} = -1021, \quad e_{\max} = 1024 \]

Requires 64 bits: 1 sign bit + 52 bits for mantissa + 11 bits for exponent

Used in almost all modern computers
Machine precision

Definition: the *machine precision* of a binary floating point number system with mantissa length $s$ is

$$\epsilon_M = 2^{-s}$$

Example: IEEE std. double precision

$$\epsilon_M = 2^{-53} \approx 1.1 \cdot 10^{-16}$$

Interpretation: $1 + 2\epsilon_M$ is the smallest floating point number greater than 1.
Rounding error

- $fl(x)$ is the floating point representation of $x$

- Numbers are rounded to the nearest floating point number; e.g.

$$fl(x) = \begin{cases} 
1 & 1 \leq x < 1 + \epsilon_M \\
1 + 2\epsilon_M & 1 + \epsilon_M \leq x \leq 1 + 2\epsilon_M 
\end{cases}$$

Gives another interpretation of $\epsilon_M$

- Rounding error and machine precision

$$\left| \frac{fl(x) - x}{|x|} \right| \leq \epsilon_M$$

  - machine precision bounds the relative error
  - number of correct decimal digits is about 16 in IEEE double precision
  - fundamental limit on accuracy of numerical computation
Floating point arithmetic

Computations reduce to elementary operations: $+, -, \times, \div$

Model for computation: roundoff error: $x$ and $y$ are floating point numbers and op is one of the four basic operations

$$x \sim op \ y = fl(x \ op \ y)$$

Fundamental axiom of floating point arithmetic

$$x \sim op \ y = (x \ op \ y)(1 + \epsilon)$$

here $|\epsilon| \leq 2^{-s}$ where results are rounded (binary number system)

Relative error

$$\frac{|x \ sim op \ y - x \ op \ y|}{|x \ op \ y|} \leq \epsilon_M$$

Consequences

- Simple operations can be inexact
- Important to keep this in mind when designing algorithms
- Goal: robustness
Example: bad pivoting

Suppose we wish to solve the system

\[
\begin{bmatrix}
\epsilon & 1 \\
1 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
= 
\begin{bmatrix}
1 \\
0
\end{bmatrix}
\]

The solution is given by

\[
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
= 
\frac{1}{1 - \epsilon}
\begin{bmatrix}
1 \\
-1
\end{bmatrix}
\]

Think of \( \epsilon \) as being very small: e.g. \( \epsilon = 10^{-20} \) so that solution obeys

\[
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
\approx
\begin{bmatrix}
-(1 + \epsilon) \\
1 + \epsilon
\end{bmatrix}
\]
Now solve this by Gaussian elimination which leads to

\[ \epsilon x_1 + x_2 = 1 \]
\[ (1/\epsilon - 1)x_2 = 1/\epsilon \]

The problem is that with \( 1/\epsilon = 10^{20} \), \( fl(1/\epsilon - 1) = fl(1/\epsilon) \) and so

\[ x_2 = 1 \]

Then we need to solve

\[ \epsilon x_1 + 1 = 1 \]

which leads to \( x_1 = 0 \). This solution is completely wrong!
Solving the same system by QR

\[
\text{epsilon} = 1e-20 \\
A = [\text{epsilon, 1; 1 1}] \\
A = 
\begin{bmatrix}
0.0000 & 1.0000 \\
1.0000 & 1.0000
\end{bmatrix}
\]

\[
b = [1; 0] \\
b = 
\begin{bmatrix}
1 \\
0
\end{bmatrix}
\]

\[
x = \text{LS\_by\_QR}(A,b) \\
x = 
\begin{bmatrix}
-1.0000 \\
1.0000
\end{bmatrix}
\]
Root Finding

Wilkinson’s polynomial

\[ p(x) = (x - 1)(x - 2) \ldots (x - 10) + \delta x^{10} \]

Roots of \( p \) computed for two values of \( \delta \)

\[ \delta = 10^{-5} \]

\[ \delta = 10^{-3} \]

Roots are very sensitive to errors in the coefficients
Implications

Possible algorithm for finding eigenvalues

1. Find the coefficients of the characteristic polynomial
2. Find its roots

This is unstable and should never be used

- Problem is badly conditioned
- Small errors in the coefficients of the polynomial will be amplified, even if root finding is done perfectly!