Introduction to Computational Mathematics
Introduction

Computational Mathematics:

- Concerned with the design, analysis, and implementation of algorithms for the numerical solution of problems that have no tractable analytical solution.

- Combines:
  1. Numerical analysis
  2. Mathematical modeling
  3. Computer science
  4. Applied mathematics
  5. Science and engineering.

- Recognized as a genuine field of the mathematical sciences.
• Why is computational mathematics important?
• Consider the following simplified model of the scientific process:

Scientific process

- Physical system
- Observe and collect data
- Conceptual interpretation
- Make predictions (get rich)
- Interpret results and compare to experimental data
- Solve the model
- Refine based on results
- Apply physical laws
- Mathematical model
- Success

• Computational math fits in the solution phase, and often in the interpretation phase.
Scientific process

- Why is computational mathematics important?
- Consider the following simplified model of the scientific process:

  Physical system
  
  Observe and collect data
  
  Conceptual interpretation

  Make predictions (get rich)
  
  Refine based on results

  Interpreting results and compare to experimental data

  Model for fluid dynamics:
  \[
  \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}
  \]
  \[
  \nabla \cdot \mathbf{u} = 0
  \]

  Why: The resulting models can essentially never be solved completely using analytical (pencil and paper) methods.
Simple example with no analytical solution

- Consider the function (called the error function):
  \[ f(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, dt \]

- Suppose some set of measurements follow a normal distribution with mean zero and standard deviation \( \sigma \).

- Then the probability that the error of a measurement is within \( \pm \varepsilon \) is given by
  \[ f \left( \frac{\varepsilon}{\sigma \sqrt{2}} \right) \]

- The definite integral defining \( f \) cannot be determined in terms of elementary functions for a general \( \varepsilon \).

- One must result to numerical approximation!
Much more complicated examples

\[2^{43,112,609} - 1\]
“Computational science now constitutes what many call the third pillar of the scientific enterprise, a peer alongside theory and physical experimentation.”

Report to the President : Computational Science : Ensuring America’s Competitiveness”, June 2005.
Algorithms

- Algorithms are the main product of numerical analysis.
- A mathematical algorithm is a formal procedure describing an ordered sequence of operations to be performed a finite number of times.
- Algorithms are like recipes with the basic building blocks of addition, subtraction, multiplication, and division, as well as programming constructs like for, while, and if.

Simple Example: Compute the \((N+1)\)-term Taylor series approximation to \(e^x\)

\[
e^x \approx \sum_{k=0}^{N} \frac{x^k}{k!}
\]

Algorithm written in pseudo code

Input: \(x, N > 0\)
Output: \((N + 1)\)-term Taylor series approximation to \(e^x\)

\[
taylor=1; \\
factorial=1; \\
xpowk=1; \\
for \ k = 1 \ to \ N \ do \\
\quad factorial = factorial \times k \\
\quad xpowk = xpowk \times x \\
\quad taylor = taylor + xpowk/factorial \\
end for
\]
• Three primary concerns for algorithms:
  - **Accuracy**: How good is the algorithm at approximating the underlying quantity.
  - **Stability**: Is the output of the algorithm sensitive to small changes in the input data.
  - **Efficiency**: How much time does it take the algorithm to obtain a reasonable approximation.

• We will briefly discuss these for the algorithms considered in this course; a more thorough discussion and analysis is part of a more advanced course in numerical analysis.

• Some other important concerns include robustness, storage, and parallelization.
Algorithms

- Algorithms can be classified into two types:
  
  - **Direct methods**: Obtain the solution in a finite number of steps, assuming no rounding errors.
    
    *Example*: Solving a linear system with Gaussian elimination
  
  - **Iterative methods**: Generate a sequence of approximation that converge to the solution as the number of steps approaches infinity.
    
    *Example*:

    We learn later that the square root of a positive number $a$ can be obtained by the sequence: $x_{n+1} = \frac{1}{2} \left( x_n + \frac{a}{x_n} \right)$.

    For example, let $a = 2$ and $x_0 = 3$. The table below shows the results

    | $n$ | $x_n$ | $|x_n - \sqrt{2}|$ |
    |-----|-------|-------------------|
    | 0   | 3.000000000000000000 | 1.58578643762690 × 10^0 |
    | 1   | 1.833333333333333333 | 4.19119770960238 × 10^-1 |
    | 2   | 1.462121212121212121 | 4.79076497481170 × 10^-2 |
    | 3   | 1.414998429894803     | 7.84867521707922 × 10^-4 |
    | 4   | 1.414213780047198     | 2.17674102520604 × 10^-7 |
    | 5   | 1.414213562373112     | 1.66533453693773 × 10^-14 |
    | 6   | 1.414213562373095     | 2.22044604925031 × 10^-16 |
Errors

• Major sources of errors in computational math:
  
  - **Truncation errors**: Result from the premature termination of an infinite computation.
    
    Example:
    \[
    e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}
    \]
    
    These are the primary concern of computational math.
  
  - **Round off errors**: Result from using floating point arithmetic.

    Less significant than truncation errors, but nevertheless can result in catastrophic problems (some examples).

• Other errors that must be accounted for:
  
  Human errors, modeling errors, and measurement errors.
Measuring errors

- This course is about learning numerical methods for approximating solutions to problems.

- Let $p$ be an approximation to $p^*$, then we have two ways of measuring the error:
  - **Absolute error:** $|p - p^*|$
  - **Relative error:** $\frac{|p - p^*|}{|p^*|}$

- Relative error is typically the best, but it depends on the problem.

- To illustrate this point consider the following simple example
  
  **Example:**

  (a) $p = 3.100$ and $p^* = 3.000$
  (b) $p = 3.100 \times 10^{-4}$ and $p^* = 3.000 \times 10^{-4}$
  (c) $p = 3.100 \times 10^{3}$ and $p^* = 3.000 \times 10^{3}$

  What are the absolute and relative errors in these cases? Which value makes the most sense to use?
Overview of the course

- We will cover the following material:
  - Floating point arithmetic
  - Solving large linear systems of equations
  - Interpolation and curve fitting
  - Solving non-linear equations and optimization
  - Numerical integration and differentiation
  - Least squares methods for over/underdetermined problems.
  - Numerical solutions of initial and boundary value problems.

- We will discuss the methods associated with these topics and the corresponding MATLAB routines.

- You will develop your own MATLAB codes for solving some applied problems.