ECE 697NA – MATH 697NA
Numerical Algorithms

Introduction

Prof. Eric Polizzi

Department of Electrical and Computer Engineering,
Department of Mathematics and Statistics,
University of Massachusetts, Amherst, USA
Connections to a “Data Structures & Algorithms” class?

Every computer program uses **data structures and algorithms**

---

**Data Structures**

**Why?** organize your data in computer's memory to efficiently store and retrieve **information**

**How?** using arrays, linked lists, stacks, queues, trees, graphs, etc.

---

**Algorithms**

**Why?** manipulate the data in various way to perform **computation**

**How?** using strategy/method, operations and analysis → Importance of Big O notation

---

**Data-centric view of the world**

- **Complex data structures** (from arrays to trees and graphs)
- **Basic 'non-numerical' algorithms act on database** (insert, remove, search, sort, traverse)

---

**Scientific Computing**

- **Basic data structure** (arrays)
- **Complex numerical algorithms** (computational modeling, HPC)
The Role of Computing in Science and Engineering

- Computing as the relative new third component of science

- Computing is a primary tool for discovery and innovation in science & engineering disciplines, it is critical to advancing a diverse collection of applications.
The Role of Computing in Science and Engineering

- Truly interdisciplinary field focusing on the whole computational modeling process for solving large-scale/scope problems in science & engineering.
Scientific Computing- How to proceed?

**Scientific modeling is a multiple-step process:**

- **Physical Modeling**
  - How to describe the physics?
- **Mathematical Modeling**
  - How to formulate the equations?
- **Numerical Modeling**
  - How to discretize the equations?
- **Numerical Algorithms**
  - How to compute the solutions?

Two difficulties:
(i) answering the questions above, *multidisciplinary approach*
(ii) making links between communities, *interdisciplinary activity*
Scientific Computing - Modeling procedure

- **Science/Engineering Application**
  - All core sciences and Engineering applications:
    - Fundamental Representation- equations (continuous mathematics, PDE)
    - Physical assumptions and modeling parameters(Input, Output)
    - Philosophical questions

- **Mathematical analysis**
  - Mathematical Treatment
    - Computational domain, Boundary Conditions
    - Variational techniques, Perturbative approach, Integral equation
    - Theorem of existence/uniqueness of solutions
    - Non-linear equations and techniques

- **Numerical Discretization**
  - Discretization Representation/ Treatment
    - Mesh, Grid, FDM, FEM, Spectral
    - Multiscale techniques
    - Domain-decomposition approach

- **Numerical Algorithms**
  - Matrix Computations
    - Eigenvalue Problems, Singular value decomposition
    - Linear Systems Solvers, Linear least square
    - Matrix Storage
    - Direct/Iterative methods
    - Sequential/ Parallel algorithms
    - Programming (language and techniques, numerical libraries)

- **SIMULATIONS**
The equation is:

\[
\left( -\frac{1}{2m} \nabla^2 + V_{\text{eff}}[n(r)] \right) \psi_i(r) = E_i \psi_i(r),
\]

\[
n(r) = 2 \sum_{i=1}^{N_F} |\psi_i(r)|^2,
\]

\[
V_{\text{eff}}[n(r)] = V_{\text{ion}}[n(r)] + V_H[n(r)] + V_{XC}[n(r)],
\]

\[
-\nabla^2 V_H(r) = \frac{\rho(r)}{\varepsilon}.
\]

The equation is:

\[
A \mathbf{x} = \lambda B \mathbf{x}
\]

<<Large>>

[\lambda_{\text{min}}, \lambda_{\text{max}}]

\[
[Q^H A Q, Q^H B Q]
\]

>>Small<<

- Pick \( Y_{N \times M_0} = \{y_1, y_2, \ldots, y_{M_0} \} \) (\( M_0 > M \))
- Repeat
  1. Compute \( Q_{N \times M_0} = \alpha (B^{-1} A) Y_{N \times M_0} \)
  2. Orthogonalize \( Q \)
  3. Rayleigh-Ritz \( A_Q \tilde{X}_Q = B_Q \tilde{X}_Q \tilde{\Lambda} \)
     \( A_Q = (Q^H A Q)_{M_0 \times M_0} \)
     \( B_Q = (Q^H B Q)_{M_0 \times M_0} \)
     \( \tilde{X}_Q = I_{(M_0 \times M_0)} \)
  4. Check convergence of \( X = Q \tilde{X}_Q \) and \( \tilde{\Lambda} \)
  5. Go back step 1 if needed using \( Y = X \)

The graph shows the weak scaling of PFEAST.
Graphs and Matrices

Model 'Real World' Problem:
- Internet traffic, social network, etc.

Model 'Mathematical' Problem:
- Solving differential equations via discretization (set of linear equations)
- Equation examples: wave, Maxwell, heat, Schrodinger, Poisson, etc.

Numerical Algorithms
- Basics: Root Finding, Numerical integrations, Fast Fourier Transform
- **Matrix Computations** - Numerical Linear Algebra

Non-Numerical Algorithms
- Searches, MST, Shortest Path, Coloring, etc.
Scientific Computing

- **Math View → Numerical Analysis**
  - “Numerical analysis is the study of algorithms for the problem of continuous mathematics”, N. Trefethen
  - *The goal of the field of numerical analysis is the design and analysis of numerical techniques to give approximate but accurate solutions to mathematical equations whose complexity make their intractable otherwise*
  - “practical mathematical calculations”

- **CS view → High-Performance Computing (HPC)**

With the significant advances in parallel architectures, there is a strong need for rethinking the computational strategies and numerical algorithms.
Computing Paradigms

Old “conventional wisdom”
- Multiply is slow, but load & store is fast
- Do not bother “parallelizing” your application; just wait a little while and run on a much faster uniprocessor
- Less than linear scaling for a multiprocessor application is a failure

New “conventional wisdom”
- “memory wall” – load & store is slow but multiply is fast
- it will be a long wait for a faster uniprocessor (end of free-speedup)
- any speed improvement via parallelism is a success.
New computational power at hand for the end user,

Most existing software is not ready to directly utilize the power of parallel (multicore) processors since they are written in traditional sequential programming languages. The user/developer had the limited scope of only one processor in mind, expecting after the Moore’s law to continuously provide optimization of his sequential code.

The hardware scaling limit (the end of Moore’s law) is going to precipitate the need for the end user for new efficient parallel numerical libraries (suitable for multicore).
Language?

- **In general:** No such thing as “best programming language”; Particular language more convenient to use in particular situation/application

- **Criteria:** Portability, Sustainability, Performance/Optimization, Scripting (matlab, python) or Compiled (C, C++, Fortran)

- Two programming approaches:
  - **1- Procedural** (with or without data objects) – codes divided into methods (procedure, subroutines)- intuitive programming - Price to pay: flexibility
  - **2- OOP** – objects contains both data and methods- offers more flexibility for some real world applications – Price to pay: abstraction, planned-ahead programming, performance

- **Fortran:** HPC workhorse/simplicity/performance/great management of data locality/intuitive interactions with BLAS/LAPACK libraries

- **Parallelism:** Shared-memory (OpenMP); Distributed memory (MPI)
'OpenMP' vs 'MPI'

Multiple Threads (threading)

Main code

Start

Main code

The End

Synchronization

Start

Main code

The End