Short Introduction to Scientific Computing

Prof. Eric Polizzi
The Role of Computing in Science and Engineering

- “Scientific Computing” == “Computational Science and Engineering” == ”Modeling and Simulations” ==> Computing
- Computing is a primary tool for discovery and innovation in science & engineering disciplines, it is critical to advancing a diverse collection of applications
- Truly interdisciplinary field focusing on the whole computational modeling process for solving large-scale/scope problems in science & engineering
The Role of Computing in Science and Engineering

- Computing as the relative new third component of science

- Open new perspectives for engineering, critical for advancing a diverse collection of applications. Examples in ECE alone: EM, Nano, Climate, etc.
Scientific Computing - How to proceed?

Numerical modeling is a multiple-step process:

- Physical Modeling
- Mathematical Modeling
- Numerical Modeling
- Computing

How to describe the physics?
What are the equations to solve?
How to discretize them?
Which algorithms to use?

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

- Math View → Numerical Analysis
  - The overall 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 (multicore workstations to petascale), there is a strong need for rethinking the computational strategies and numerical algorithms
Parallel computing

- 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)
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.
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, Schrödinger, 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

- **Problem types (Matrix Computations)**
  - Matrix-Vector multiplications $Ax$
  - Linear systems $Ax=f$ (find $x$)
  - Eigenvalue problems $Ax=\lambda x$

- **Data formats**
  - Dense, Banded, Sparse

- **Implementation**
  - Data locality, low-level numerical libraries (e.g. BLAS, LAPACK)
  - Array-based implementation

- **Language**
  - Scripting: Matlab/Python (Education or special purpose)
  - C, C++, Fortran (for performance/research/production)
  - Parallelism
    - Shared-memory (OpenMP)
    - Distributed memory (MPI)
'OpenMP' vs 'MPI'

Multiple Threads (threading)

Start

Main code

... 

Start

Main code

Synchronization

... 

Start

Main code

Synchronization

... 

The End

The End