HIGH-PERFORMANCE SCIENTIFIC COMPUTING (HPSC) ME522

Instructor: Gaurav Bhutani

SMME, IIT Mandi

Feb-Jun 2023 semester

Venue: to be announced

Online: https://meet.google.com/xyn-osvu-yys

Lecture plan

- Course essentials
- Goals and learning outcomes
- Pre-requisites and software requirements
- Short demonstration

Course essentials

Notes

- Class: 4 hours per week (2 sessions).
- Venue to be announced.
- One hr lecture, followed by 1 hour lab / hands-on session. Use your own laptop.
- Teaching supporter: Ayush Sahu (SMME Metch
  Help with labs, installing software, obtaining material, announcements
- Announcements: Course mailing list / Google chat group
- Learn from peers
- Moodle – all course slides and link to material
- Codes on Github as we create
- Virtual machines on IIT Mandi Cloud

Evaluation

- Lab/class attendance and random lab viva (20)
- Midsem (30)
  - Written (10)
  - Lab exam (20)
- Endsem (50)
  - Written (20)
  - Lab exam (30)

Goals

HPC

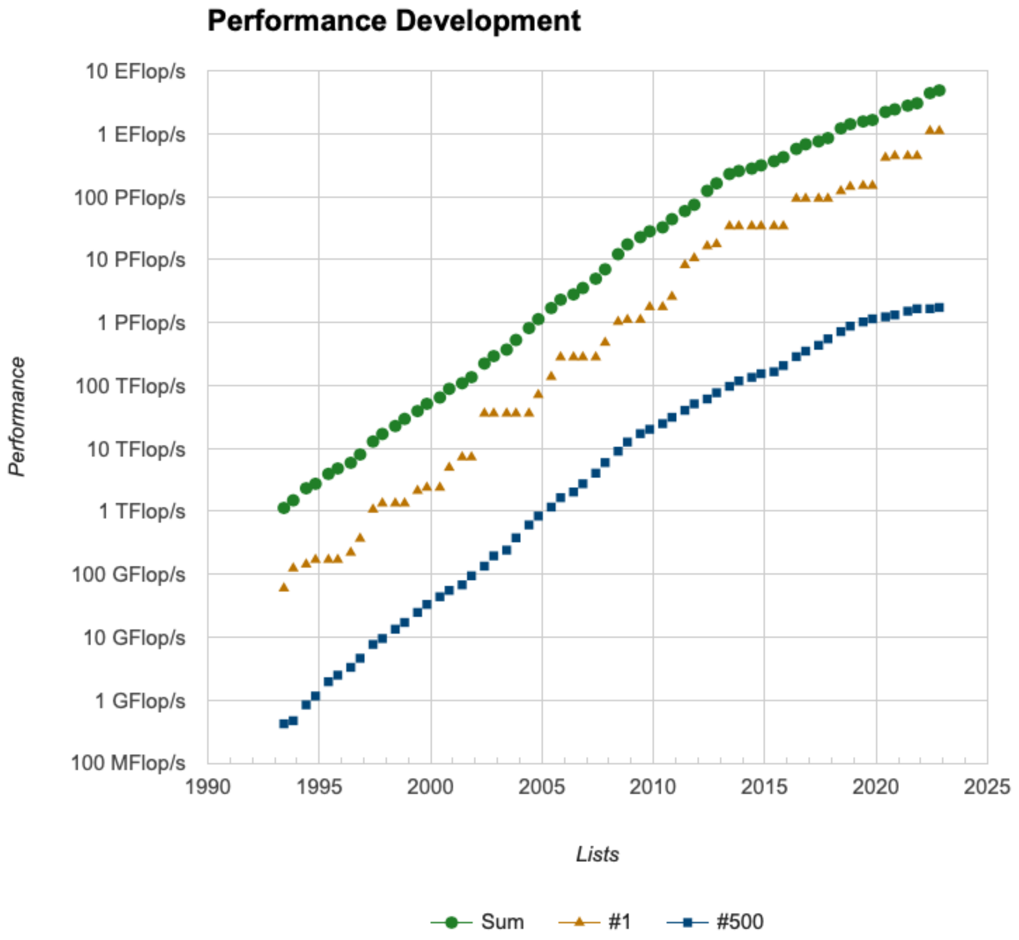
- **High Performance Computing (HPC)** often means heavy-duty computing on clusters or supercomputers with 100s of thousands of cores.
- *“World’s fastest computer”*
- #1. Frontier – HPE Cray (Oak Ridge National Lab, USA)
  8.7M cores ≈ 1100 Petaflops; 21MW power
- #3. Leonardo – Atos (CINECA, Italy)
  1.4M cores ≈ 174 Petaflops; 5.6MW
- Param Himalaya – Atos (IIT Mandi, India)
  800 Teraflops; 150kW

See http://top500.org for current list.

Increasing speed

- Moore's Law: Processor speed doubles every 18 months.
  ⇒ factor of 1024 in 15 years.
- Going forward: Number of cores doubles every 18 months

- Top: Total computing power of top 500 computers
- Middle: #1 computer
- Bottom: #500 computer



http://www.top500.org

Our focus

- Our focus is more modest, but we will cover material that is:
  - Essential to know if you eventually want to work on supercomputers
  - Extremely useful for any scientific computing project, even on a laptop.
- Focus on scientific computing as opposed to other computationally demanding domains, for which somewhat different tools might be best.

Learning outcomes

Efficient processing

- Basic computer architecture, e.g. floating point arithmetic, cache hierarchies
- Using Unix (or Linux)
- Language issues, e.g. compiled vs. interpreted, object oriented, etc.
- Specific languages: Python (for scripting), Fortran 90/95 (for fast processing)
- Parallel computing with OpenMP, MPI
- Not included: GPU-based parallelisation

Good software practices

- Version control using Git, github
- Makefiles
- Debuggers, code testing
- Reproducability
- Use of high-performance computing (HPC) clusters

Strategy

- So much material, so little time…
- Concentrate on basics, simple motivating examples.
- Get enough hands-on experience to be comfortable experimenting further and learning much more on your own.
- Learn what's out there to help select what's best for your needs. New languages are introduced with time – similar ideas though.
- Teach many things "by example" as we go along.
- You'll be expected to read supplementary notes when they are provided.
- No specific book for the course. Internet search for help will be useful.

Pre-requisites & software requirements

Pre-requisites

- Some programming experience in some language, e.g., Matlab, C.
- You should be comfortable:
  - editing a file containing a program and executing it,
  - using basic structures like loops, if-then-else, input-output,
  - writing subroutines or functions in some language
- You are not expected to know Python or Fortran.
- Some basic knowledge of linear algebra, e.g.:
  - what vectors and matrices are and how to multiply them
  - How to go about solving a linear system of equations
- Comfort level for learning new software.

Software requirements

- You will need access to a computer with a number of things on it. All open-source software.
- Note: Unix is often required for scientific computing.
- Windows: Many tools we'll use can be used with Windows, but learning Unix is part of this class.
- Options:
- Install everything you'll need on your own computer.
- Install VirtualBox and use the Virtual Machine (VM) created for this class.
- Use Amazon Web Services with an Amazon Machine Image (AMI) which will be created for this class. Other cloud computing services are also available.

Know the class

- A short intro by various class members
  - Name and program enrolled in
  - What are your academic interests and what research projects you are working on? For PG students please mention your research group at IIT Mandi.
  - Why interested in this course? What are your expectations from the course?

Short demonstration

Demo code

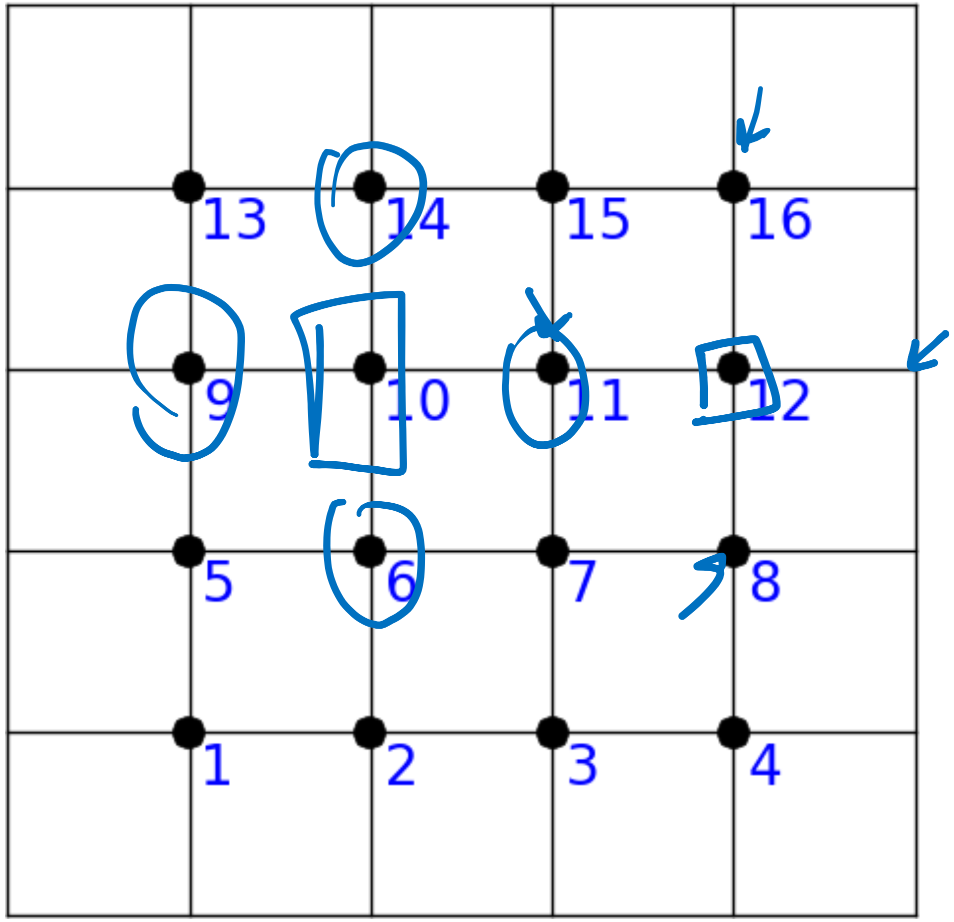
```
$ cd <folder name>
$ export HPSC=$PWD
$ cd $HPSC/lecture1

$ make plots

$ display *.png
```

Steady-state heat conduction

- Discretize on an $N \times N$ grid with N^2 unknowns
- Assume temperature is fixed (and known) at each point on boundary.
- At interior points, the steady state value is (approximately) the average of the 4 neighbouring values.

$$[A]_{N^2 \times N^2}\{x\}_{N^2} = \{b\}_{N^2}$$

Row-wise ordering



$\nabla^2 u = 0$

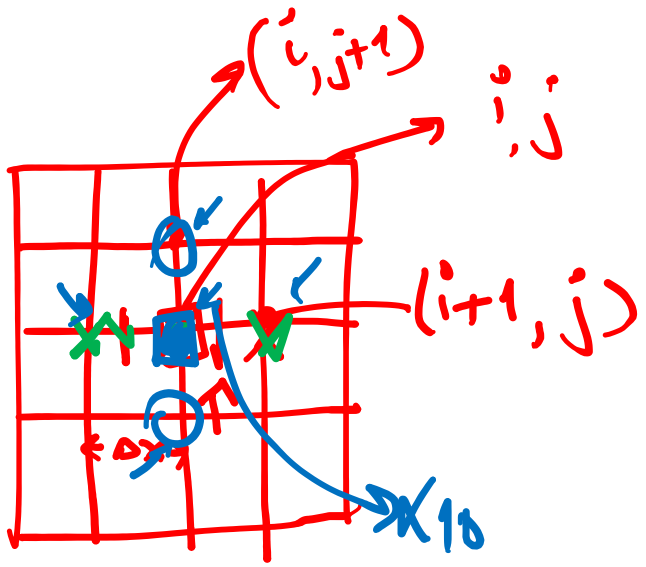
2-D $\quad \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = 0$

$\underbrace{[A]}_{N^2 \times N^2} \{x\} = \{b\}$

$\begin{Bmatrix} x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{Bmatrix}$

$N^2 \quad 120 \times 120$

14400



$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1,j} - 2\,\underbrace{u_{i,j}} + u_{i-1,j}}{(\Delta x)^2}$$

$$u_{i,j} = f(u_{i-1,j},\ u_{i+1,j},\ u_{i,j-1},\ u_{i,j+1})$$

Steady-state heat conduction

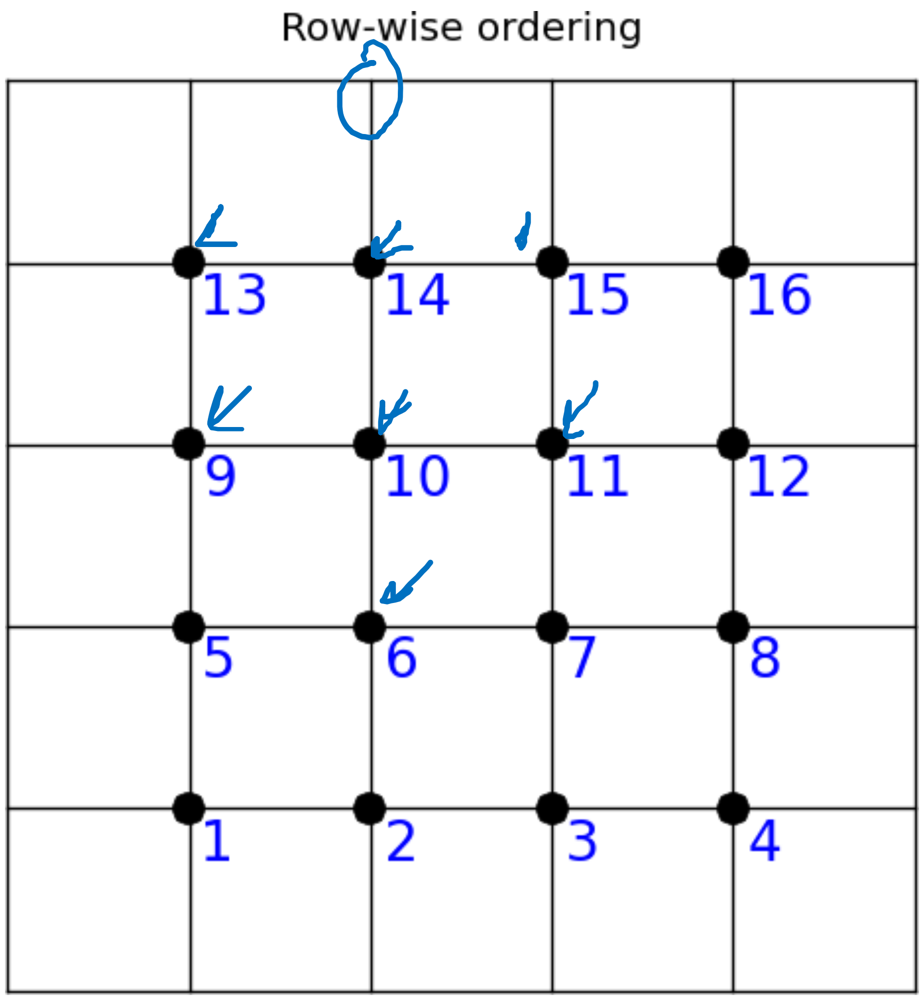
$$u_{i,j} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}\right)$$

- Holds for i, j = 1, 2, . . . , N with $u_{0,j}$ known on boundary. Gives a linear system Au = b, with N^2 equations N^2 unknowns. Matrix A is $N^2 \times N^2$, for $N = 120$, $N^2 = 14400$.

$x = A^{-1} b$

- Very sparse: each row of matrix A has at most 5 nonzeros. Gaussian elimination is not the best approach.
- Jacobi method (not the best method)

$$u_{(i,j)}^{[k+1]} = \frac{1}{4}\left(u_{(i-1,j)}^{[k]} + u_{(i+1,j)}^{[k]} + u_{(i,j-1)}^{[k]} + u_{(i,j+1)}^{[k]}\right)$$



Speedup of linear solvers

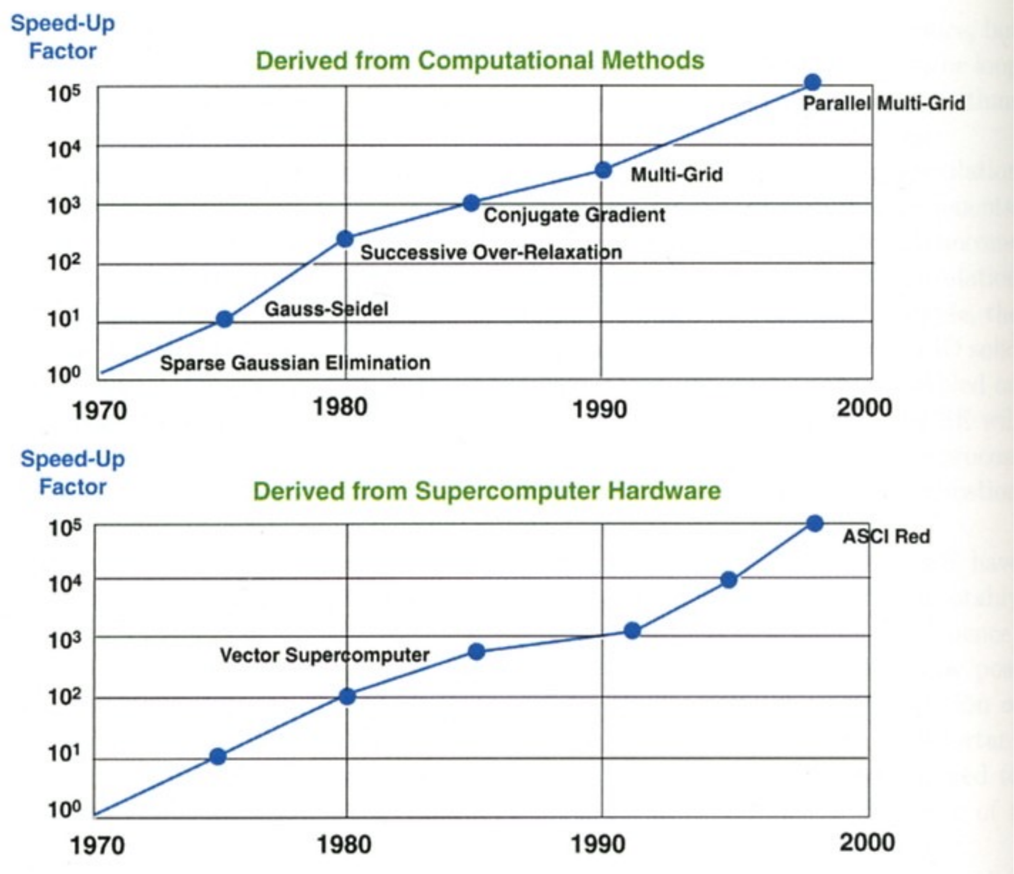


Fig. 2 *Comparison of the contributions of mathematical algorithms and computer hardware.*

Source: SIAM Review 43(2001), p. 168.

- Exponential increase in the speed of numerical algorithms.
- Exponential increase in the computer hardware performance with time.

Class virtual machine

- Available on Google Drive - `https://drive.google.com/drive/folders/1mnMJoJNqiOpcuFzVZeW8EfoxqlqHWWVG?usp=sharing`
- Username: hpsc; password: me522
- This file is large! About 5 GB compressed. After unzipping, about 10 GB.

References

- Slides are adapted from HPSC, AM483, Uni of Washington by Randall J Leveque (https://faculty.washington.edu/rjl/teaching.html)
  Refer to his page for more useful material and notes.