OLD DOMINION UNIVERSITY

Tools of computational physics

A. Godunov

1. Major tools of computational physics
2. Computers: hardware
3. Computers: operating systems
4. Computers: programming languages

updated 31 August 2022

1

Part 1:

Tools of computational physics

2

Computational Physics

Computing has become a central tool in physics!

From R. Landau "Computational Physics"

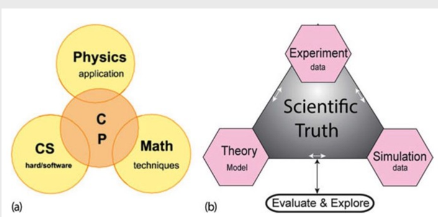


Figure 1.1 (a)A representation of the multidisciplinary nature of computational physics as an overlap of physics, applied mathematics and computer science, and as a bridge among them. (b) Simulation has been added to experiment and theory as a basic approach in the search for scientific truth. Although this book focuses on simulation, we present it as part of the scientific process.

3

3

Toolbox

- Physics: methods for computer simulations
- Math: solving numerical problems
- Hardware and Software

4

4

Hardware and software

The basic ideas behind computational physics are hardware and software independent

However, for solving a problem in hand one may need to consider advantages and disadvantages various choices.

Critical or important

- Development time
- Computational time
- Computational memory (RAM)
- Storage memory

Use right tools for your problem!

5

5

Software

Principal parts

1. Operating system
2. Ideally IDE integrated development environment + language(s)
   a) Editor
   b) Compiler(s) (for high-level languages)
   c) Libraries (modules, toolboxes,...)
3. *Optional:*
   a) Lint and other static code analyzers
   b) Debugger
   c) Profilers
   d) Report generators

6

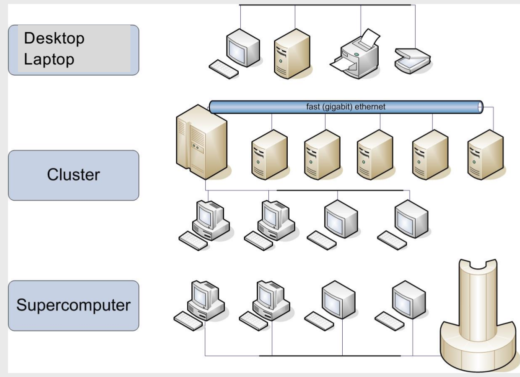
6

Part 2:

Tools: Hardware

7

Hardware 1.



8

8

Internal hardware

Most important for computing

- CPU(s) - central processing unit (how many + how fast)
  also - cache memory: cache 1, cache 2
- RAM -random-access memory (GB)
  communication with CPU by bus
- GPU – Graphic Processor Unit (how many + how fast)
  Can speed up calculations considerably!
- Network Interface (MB/GB per sec)
- *HDD – Hard Disk Drive (TB)*

9

9

Top 500

The TOP500 lists the 500 most powerful commercially available computer systems https://top500.org

TOP500 LIST - JUNE 2021

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

R_{peak} values are calculated using the advertised clock rate of the CPU. For the efficiency of the systems you should take into account the Turbo CPU clock rate where it applies.

← 1-100 101-200 201-300 301-400 401-500 →

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371

TOP500 LIST - JUNE 2022

R_{max} and R_{peak} values are in PFlop/s. For more details about other fields, check the TOP500 description.

R_{peak} values are calculated using the advertised clock rate of the CPU. For the efficiency of the systems you should take into account the Turbo CPU clock rate where it applies.

← 1-100 101-200 201-300 301-400 401-500 →

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	1,110,144	151.90	214.35	2,942
4	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096
5	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94.64	125.71	7,438
6	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93.01	125.44	15,371

10

Part 3:

Software: operating systems

11

Two major families

1. Unix and Unix-like
   a) Unix - commercial versions: AIX, HP-UX, Solaris, …
   b) Apple - macOS
   c) Linux – many distributions
2. Windows

Share of operating systems for desktops/laptops

Share of operating systems for clusters/supercomputers:
Linux now runs on all the fastest 500 supercomputers in the world

12

12

UNIX family

Developed in around 1970 in Bell Labs research center

1. Powerful beyond imagination.
2. ALL the Top 500 supercomputers in the world run on Linux.
3. Linus is based on Unix
4. Robust and small kernel.
5. Very safe: sandboxing and rich file permission system.
6. Plenty of tools
   editors, programming languages, …

Philosophy of Unix/Linux: “Building blocks” + “glue”

- Building blocks: programs do only one thing, but do it well
- Glue: easy combine various blocks.

13

13

Windows

Generally available since 1992

1. Most popular OS in the world for personal computers and laptops.
2. The latest version is Windows 11.

14

14

Part 4:

Software: Programming languages

15

Programming languages

Important questions

1. Which language to learn?
2. Which language to use?
3. Do I need to learn new language(s)?

Most common in physics

- C and C++ (current standard C++20)
- Python
- Fortran
- Matlab
- *Julia*
- *R*

16

16

Computer languages

Three basic modes to run a code

1. Interpreted: Python, Matlab, Mathematica, R.
2. Compiled: Fortran, C/C++.
3. JIT (Just-in-Time) compilation: Julia

Interpreted languages can we used with:

1. A command line
2. A script file.

17

17

C

Created in the early 1970s by Dennis Ritchie at Bell Labs,

- General purpose, multi-paradigm, compiled language
- By design, C's features reflect the capabilities of the targeted CPUs
- It was designed to be compiled to provide low-level access to memory and language constructs that map efficiently to machine instructions
- Many languages have based directly or indirectly on C, including C++, C#, Java, JavaScript, Julia, Perl, PHP, Python, Ruby, Swift, …

18

18

C++

Developed by Bjarne Stroustrup at Bells Labs in the early 1980s

- General purpose, multi-paradigm, compiled language
- C/C++ is the infrastructure of much of the modern computing world.
- Powerful language: you can code anything in C++
- Easy integration with multiprocessor programming
  OpenMP, MPI, CUDA, OpenCL, ...
- If you know Unix/Linux and C/C++, you can master everything else
- Excellent compilers (including open-source) and tools.
- Top performance in terms of speed.

19

19

C++

Some disadvantages

- Hard language to learn, even harder to master
- Large specification: C++20 (This causes, at times, portability issues)
- Matrix indexing starts at zero

20

20

FORTRAN (formula translator)

Grandfather of all modern languages – developed in 1957 (IBM)

- General purpose, multi-paradigm, compiled language
- Last version Fortran 2018
- Lot of high-quality libraries (both numerical and applications)
- Still widely used in science in engineering
  weather forecast, nuclear weapon research and development, …
- Easy to learn, portable, nice array support, easy to parallelize
- Generally available on clusters and supercomputers

21

21

FORTRAN (formula translator)

Some disadvantages

- Small community of users
- Most Fortran compilers are proprietary

22

22

Python

Designed by Guido von Rossum around 1991

- General purpose, multi-paradigm, interpreted language
- Open source
- Intuitive – easy to learn
- Scientific computation modules: NumPy, SciPy, and SymPy
- Plotting modules: matplotlib and ggplot.
- Preinstalled on many systems (e.g. macOS)

23

23

Python

Some disadvantages

- Considerable time penalty
- Python’s memory usage is high
- Python’s functional programming can be difficult to read
- Runtime Errors: One of the major drawbacks of this language is that its design has numerous issues

24

24

MATLAB

Started in the late 1970s, released commercially in 1984.

https://www.mathworks.com/products/matlab.html

- General purpose, multi-paradigm, interpreted language
- Widely used in engineering and industry
- Plenty of codes around
  for science, engineering and economics.
- Many useful toolboxes
- Great IDE (Integrated Development Environment)
- Interacts reasonably well with C/C++, Fortran, and R

Some disadvantages

- Can be expensive
- Tight integration with Java

25

25

Other languages

Julia

- Modern, high-performance programming language designed for scientific computation and data manipulation.
- Designed for parallelism and cloud computing. Syntax close to Matlab. However, at early stages of life (can be unstable)

R

- High level, open-source language for statistical computation
- Widely used for big data, easy to parallelize

Mathematica

- Mainly oriented toward symbolic computation
- Programming approach is different from other languages.

And more: C#, Javascript, PHP, Perl, Swift, Ruby, …

26

26

Compare times of calculation …

Results depends on a model/test but here are some average numbers

```
C++         1.00
Fortran     0.90
Python      50.0
Matlab      10.0
Mathematica from 4.0 (idiomatic) to 900 (base)
Julia       3.0
R           250
```

27

27

Summary

- C++ good to learn (most powerful general programming language) if you master C++ you can quickly learn anything else.
- Fortran very powerful but learn only if needed (legacy codes or libraries)
- Python easy to learn, open source, but generally much slower than C++ and Fortran
- MATLAB convenient with great IDE, making graphs, multiple toolboxes available
- Mathematica good problem-solving environment but programming approach is different from other languages
- Java – rather no, unless the use of Virtual machine is important

28

28

C++ compilers and IDEs

Microsoft Visual C++ compiler
https://visualstudio.microsoft.com/vs/features/cplusplus/
Windows (IDE included)

Xcode (from Apple)
https://developer.apple.com/xcode/
macOS (IDE included)

Intel C++ compiler
www.intel.com/content/www/us/en/developer/tools/oneapi/dpc-compiler.html
Windows, macOS, Linux (works with Microsoft Visual Studio)

Dev-C++
http://www.bloodshed.net
Windows (IDE included) (open source)

29

29

Fortran compilers and IDEs

ABSoft
https://www.absoft.com
Windows, macOS, Linux (IDE included)

NAG (Numerical Algorithmic Group)
https://www.nag.com/content/nag-fortran-compiler
Windows, macOS, Linux (IDE included)

Intel Fortran compiler
www.intel.com/content/www/us/en/developer/tools/oneapi/fortran-compiler.html
Windows (IDE: Microsoft Visual Studio), macOS (IDE: Xcode),
Linux (IDE: Eclipse)

30

30

Python

Python.org
https://www.python.org
Windows, macOS

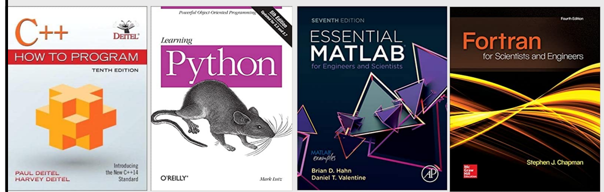
Anaconda
https://www.anaconda.com/products/individual
Windows, macOS

IDEs (for Windows, macOS and Linux)
- IDLE https://docs.python.org/3/library/idle.html
- Visual studio https://visualstudio.microsoft.com/vs/features/python/
- Spyder https://www.spyder-ide.org
- Atom https://atom.io
- And many more …

31

31

Books … so many



32

32

Example: a circle using C++

```
// calculation: the diameter, circumference, and area of a circle
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
    const double pi=3.1415926;
    double radius, diameter, circumference, area;
    cout << "enter radius as float " << endl;
    cin >> radius;
    diameter = 2.0*radius;
    circumference = 2.0*pi*radius;
    area = pi*pow(radius,2);
    cout.setf(ios::fixed | ios::showpoint);
    cout.width(10);
    cout.precision(5);
    cout << "radius   = " << radius        << endl;
    cout << "diameter = " << diameter      << endl;
    cout << "circumf. = " << circumference << endl;
    cout << "area     = " << area          << endl;
    return 0;
}
```

33

33

Example: a circle using MATLAB

```
% calculation: the diameter, circumference, and area
% of a circle with a given radius

Pi = 3.1415926;

prompt = 'Enter radius of a circle \n';
radius = input(prompt);

diameter = 2.0*radius;
circumference = 2.0*Pi*radius;
area = pi*radius*radius;

fprintf(' radius        %8.4f   \n',radius);
fprintf(' diameter      %8.4f  \n',diameter);
fprintf(' circumference %8.4f  \n',circumference);
fprintf(' area          %8.4f  \n',area);

%end
```

34

34

Example: a circle using Python

```
# -*- coding: utf-8 -*-
""" From "COMPUTATIONAL PHYSICS" & "COMPUTER PROBLEMS in PHYSICS"
    by RH Landau, et all."""

# Area.py: Area of a circle, simple program
from math import pi

N = 1
r = 1.3
C = 2.* pi* r
A = pi * r**2

print ('Program number =', N, '\n r, C, A = ', r, C, A)
```

35

35

Example: Fortran – Fibonacci prime numbers

```
 program fibonacci
! the program generates Fibonacci numbers and chooses only prime numbers
! f(0) = 0
! f(1) = 1
! f(n) = f(n-1) + f(n-2) for n>1
implicit none
integer :: f(0:100)
integer :: i, j
character :: prime*5
 f(0) = 0
 f(1) = 1
 do i=2,40
   f(i) = f(i-1) + f(i-2)
!  check for prime numbers
   prime = 'prime'
   do j=2,f(i)-1
     if (f(i) == (f(i)/j)*j) then
       prime = '     '
       exit
     end if
   end do
   write (*,102) i, f(i), prime
 end do
102 format(i3, i12, a6)
 stop
 end
```

36

36

Other items

1. Version Control
2. Backups
3. Dynamic notebooks (Jupiter, Markdown, …)

37

37

Part 4b:

High Performance Computing (HPC)

38

High-performance computing (HPC)

Deals with scientific problems that require substantial computational power.

Usually, but not always, HPC involves the use of several processors:

- Multi-core/many-core CPUs (in a single machine or networked).
- Many-core coprocessors.
- GPUs (graphics processing units).
- TPUs (tensor processing units).
- FPGAs (field-programmable gate arrays)

"Amateurs talk about the speed of their processors, but professionals study coding techniques" from Gen. Robert H. Barrow, USMC (27th Commandant of the US Marine Corps)

39

39

High-performance computing (HPC)

Resources

- Livermore National Lab https://hpc.llnl.gov/training/
- HPC carpentry https://www.hpc-carpentry.org
- More https://pages.tacc.utexas.edu/~eijkhout/istc/istc.html
- …

And many books

40

40

Few quotes

"Spend your intellectual energies on the current problem - not on fancy tools. When the volume and sophistication of your problems demand these weapons you will know it. That is the time to learn a new tool - and learn it by re-doing an already-solved problem, not a new one."
F.S. Acton

"I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs."
Maurice Wilkes, after the first attempts to write programs for the EDSAC computer

"I conclude that there are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies."
Charles Hoare, inventor of the QuickSort algorithm, in his 1990 ACM Turing Award Lecture

41

41