

Computational Astrophysics I: Introduction and basic concepts

Helge Todt

Astrophysics
Institute of Physics and Astronomy
University of Potsdam

SoSe 2024, 10.4.2024



Recommended prerequisites:

- basic knowledge of programming, especially in C/C++ → e.g., “Tools for Astronomers”
- basic knowledge in astrophysics

How to get a certificate of attendance / 6 CP/LP/ECTS ($\hat{=}$ 4 semester periods per week):

- without mark, e.g., Master of Astrophysics, module PHY-765: Topics in Advanced Astrophysics (this module has in total 12 CP! and an oral exam at the end):

→ at least 1./3. of the points of the exercises

Attention!

PULS is strict: It is absolutely necessary to enroll for this lecture until **10.05.2024!**

- with a mark (other Master courses):
little programming project at the end of the semester

Please note that the focus for this course is on the exercises!

Specialization track “Computational Astrophysics”

the regulations for obtaining the computational astrophysics specialization *certificate* are as follows:

- 1 Computational Astrophysics I (4 SWS, SoSe) :
Computational Astrophysics: Introduction
Computational Astrophysics: Basic Concepts
- 2 Computational Astrophysics II (3 SWS, WiSe)
Advanced Computational Astrophysics: Concepts and Applications
- 3 Seminar Computational Astrophysics (2 SWS)
Advanced Computational Astrophysics: Seminar
- 4 a 4th course of the computational curriculum, e.g.,
Computational Astrophysics: Advanced Programming (2 SWS)

Aims & Contents of CA I:

- enhance existing basic knowledge in programming (C/C++)
- brief introduction to Fortran → relatively common in astrophysics
- work on astrophysical topics which require computer modeling:
 - solving ordinary differential equations
 - from the two-body problem to N -body simulations
 - stellar structure, the Lane-Emden equation
 - solving equations: linear algebra, root finding, data fitting
 - data analysis
 - data analysis and simulations
 - simulation of physical processes
 - Monte-Carlo simulations and radiative transfer
- + introduction to parallelization (e.g., OpenMP)

What are computers used for in astrophysics?

- control of instruments/telescopes/satellites:

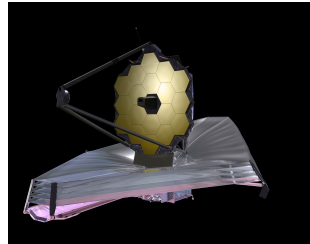
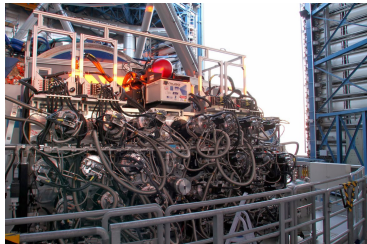


Figure: Multi Unit Spectroscopic Explorer (MUSE), Very Large Array (VLA), James Webb Space Telescope (JWST)

- data analysis / data reduction

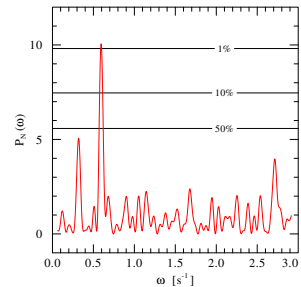
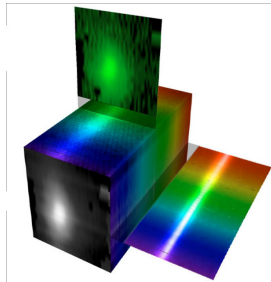
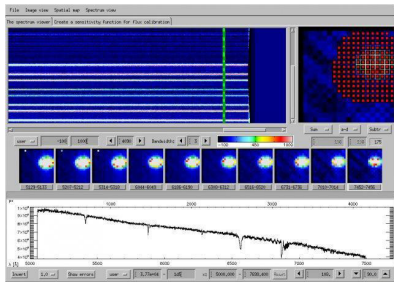


Figure: IDL, 3dCube / FITS, Fourier analysis

- modeling / numerical simulations

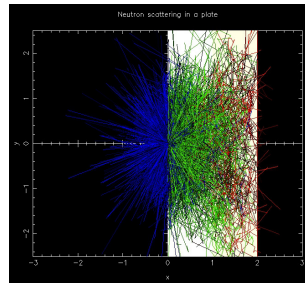
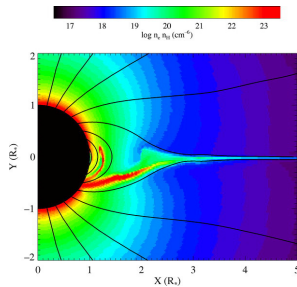
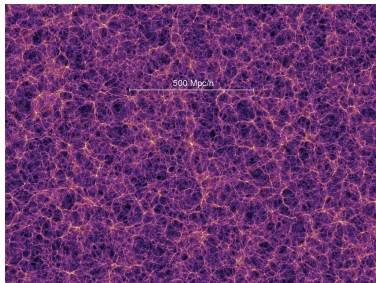


Figure: N-body simulation, hydrodynamics , Monte-Carlo

Computer lab

Meaning of the fonts / shapes

font/shape	meaning	example
xvzf (typewriter)	text to be entered literally (e.g., commands)	man ls
<i>argument</i> (italic)	place holder for own text	file <i>myfile</i>

User accounts

useful for the lecture: your own account **for this computer lab**
(room 0.087 & 1.100)

Please, get your own account!

Sysad: [Helge Todt](#), room 2.004

Guest account

→ see left-hand side whiteboard
only valid per computer and in room 0.087

Attention: Unix/Linux is case sensitive!

Hint: You can choose the session type (e.g., Xfce, IceWM) at login screen.

Security advice

As soon as you got your own account:

`passwd` Change the user password

(Enter the command in a terminal, Xterm, Konsole, or similar.)

Change your initial NIS password(!) to a **strong** password, use

- at least 9 characters, comprising of:
- capital AND lowercase letters, but not single words
- AND numbers
- AND special characters (Attention! Mind the keyboard layout!)

e.g., \$cPhT-25@comP2 or tea4Pollen+Ahead

But: prefer length over complexity!

The initial password expires after 14 days.

Computers:

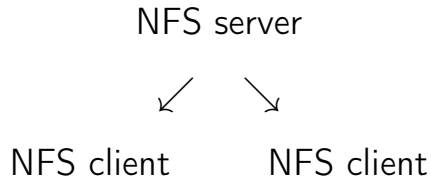
- 17 NFS¹ mounted Linux computers (openSUSE 15.4/15.5), several Intel Core i7-2600K, i7-4770, i7-7700, i7-8700 + 1 Xeon Gold 6152 44-core compute server
- home server (`~user`) always-on:
 - bell
 - mahler
 - weber

room 0.087:

- only for lectures
- Please, do not eat or drink in this room.

student's computer lab in room 1.100:

- open during the day
- b/w printer (500 pages / semester) and color printer (100 pages / semester)



NFS server: provides (home) directories (physical on disk)

NFS clients: mount NFS (home) directories in their root directory

As also other users might have their home directory on your computer

Never switch off the computers!

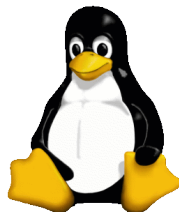
Linux

Linux is a derivative of the operating system **UNIX**. It is a multi-user and multitasking operating system.

It was written in 1991 as a UNIX for PCs, now available for (almost) every platform, e.g., as Android or in Wireless routers and under permanent development.

Linux is . . .

- for free
- open source (program code can also be modified)
- the combination of a *monolithic*¹ kernel and (GNU) software
- dominant in supercomputers (more than 90%)



¹i.e., kernel contains also hardware driver

Important X-Window based environments under Linux: [GNOME](#) and [KDE](#),
here: [Xfce](#)

Desktop environment (session type) can be chosen during local login, e.g., [Xfce](#) (nice) or
[IceWM](#) (simple)

Desktop environment \neq Linux

<i>Desktop environment:</i>	KDE	Xfce	GNOME	...
-----------------------------	-----	------	-------	-----

<i>Linux distributions:</i>	Ubuntu (Debian)	openSUSE	...
-----------------------------	-----------------	----------	-----

`xterm` or terminal: input of Linux shell commands, e.g., `cd`, `ls`

`emacs` or `kate`: editor for ASCII text files, e.g., `hello.cpp`

`g++` or `gfortran`: gcc compilers, e.g., `g++ -o hello hello.cpp`

Shell and shell commands

Unix provides by the *shell* (command line) an extremely **powerful** tool. Within the shell Unix commands are executed.

Unix command syntax

```
command [-option] [argument] <ENTER>
```

Attention! Mind the blanks!

Open an xterm or similar terminal/console and enter following: (finish each line with <ENTER>):

```
echo hello
```

and

```
echo -n hello
```

What's the difference?

Tip 1:

Command history

By ↑ (arrow key up) you can repeat the last commands entered in the shell.

A list of the last commands can be shown via the command

```
history
```

Tip2:

Moreover, you can save typing by using the `TAB` key, it completes commands or file names:

```
ech TAB
```

is completed to

```
echo
```

Tip 3:

Linux: Copy by Selection

mark the text with the mouse:

press left mouse button, keep it pressed

move mouse cursor until end of the region you want to mark

→ marked region will be highlighted

marked text was copied to the *clipboard*

paste the copied text:

move cursor to the intended position

press the middle(!) mouse button (or wheel)

→ the previously copied text was inserted

Directories

The filesystem tree

```
/
|
|-- /home/
|   |
|   |-- /home/weber/
|       |
|       |-- /home/weber/htodt/
|
|
|-- /etc/
|
|-- /dev/
```

→ root of the FS tree

→ Home directories

→ Homes on weber

→ Helge's home

→ et cetera (config)

→ devices

Navigation through directories I

`pwd` shows the current directory path (absolute)
e.g., `/home/weber/htodt`

`cd name` change to directory *name*

`.` means the current directory

`..` the parent directory, e.g., `cd ..`

`/` root of the FS tree

`~` the home directory, e.g., `cd ~` or just `cd`

`~user` the home directory of *user*

`mkdir name` create directory *name*

`rmdir name` remove directory *name*

`ls` show (list) the content of the directory

Navigation through directories III

`ls` show the content of the current directory

`ls -a` also show hidden files (starting with a `.`)

`ls -l` show the file attributes, owner, creation time

File attributes

```
drwxr-xr-x  2 htodt users   4096 14. Oct 13:35 Documents
```

`d` = directory

`r` = readable

`w` = writeable

`x` = executable

`htodt` = owner

`users` = group

`4096` = size in byte `14. Oct 13:35` = creation time

`Documents` = name of the file (here: of the directory)

Hint: `ls -lc` → time of last modification ; `ls -lu` → time of last access

Navigation through directories IV

- `man ls` Manual pages (help for the command `ls`)
- `info ls` Info pages (alternative help for the command `ls`)
- `ls --help` Help for the command `ls`
- `ls --help | less` if more than one screen page

man page navigation – also `less`, `more`

<code>q</code>	quit		
<code><SPACE></code>	next page	<code>b</code>	previous page
<code>/</code>	forward search	<code>?</code>	backward search
<code>n</code>	next occurrence	<code>N</code>	previous occurrence
<code>></code>	jump to the end	<code><</code>	jump to the beginning

→ to create pure ASCII files (e.g., as input for g++)

```
emacs file &
```

Starting programs in background:

The ampersand & at the end of a command let the command run in the background (bg) of the shell.

Hence, the input line of the shell can be still used.

If forgotten: <CTRL>+z followed by bg <ENTER>.

`emacs` available on almost every system (must be installed),
window- or terminal-based (`emacs -nw`)

<code><STRG> + x</code>	<code><STRG> + c</code>	close (quit/exit)
<code><STRG> + x</code>	<code><STRG> + s</code>	save
<code><STRG> + k</code>		kill (cut, from cursor to end of line)
<code><STRG> + y</code>		yank (paste)
<code><STRG> + <SPACE></code>		mark
<code><STRG> + w</code>		cut marked region
<code><ESC> w</code>		copy marked region
<code><STRG> + a</code>		go to beginning of line
<code><STRG> + e</code>		go to end of line

Files

Remark: In Linux almost everything is a file (also directories and devices, see `ls -l /dev/`).

`mv source target` move (rename) files

`cp source target` copy files

`rm file` remove file

`rm -rf directory` remove directory

tar action archive files use *action* on *archive*

tar actions

c	create archive from file/directory
x	extract archive
v	show executed actions (v erbose)
z	zipp archive
t	show content of archive
f	archive is a f ile (default: tape device)

Example: Untar a tarball

```
tar xvzf muCommander.tar.gz
```


Connecting to other
hosts (computers)

`hostname`

this command shows the name of the host
you're currently logged in

Connection to another host (*remote host*) under Linux/Unix with the *secure shell*, within the same domain (e.g., within the computer lab cluster)

```
ssh host name
```

After successful *login*, in the same terminal/window a shell is shown that runs on the remote host.

The SecureShell

Client-server system for establishing a secure connection (encrypted), login on the remote host (remote host = [SSH server](#))

if SSH client and SSH server support X11:

```
ssh host name -Y
```

allows the SSH server to open a graphical window (e.g., for `evince` or `kate`) on the [SSH client](#)

Besides the interactive use of the SSH one can also just let a program run on the remote host via ssh:

```
ssh hostname "ls -l"
```

The connection will be automatically closed after program/command is finished.

Login from outside (e.g., from home):

```
ssh username@bell.stud.physik.uni-potsdam.de
```

There are SSH clients for Windows that are for free, e.g., [PuTTY](#). Moreover, [MobaXterm](#), [Xming](#), [X2Go](#) (requires also installation on the server) or with help of the Windows Subsystem for Linux (requires the installation of Linux distribution) it is also possible to perform a graphical SSH login from Windows to Unix/Linux.

Hint:

With help of the graphical login you can, e.g., use [Mathematica](#) on the computer lab cluster at home.

For login without password:

- 1 run `ssh-keygen` on the client, answer all question just with `<ENTER>`
- 2 add the resulting `~/.ssh/id_rsa.pub` from the client host to `~/.ssh/authorized_keys` on the *remote host*

With help of the SSH protocol it is also possible to transfer files between different computers:

```
scp document.txt username@bell.stud.physik.uni-potsdam.de:
```

secure copy **to** the remote host

```
scp username@bell.stud.physik.uni-potsdam.de:document.txt .
```

secure copy **from** the remote host (mind the dot!)

After the colon `:` is a path given, either absolute or relative to the home directory

To copy only files that have been modified (comparison of source and target):

```
rsync -rtvz username@host.domain:directory/ .
```

secure copy **from** the remote host, only modified files

some useful options:

- r recursive: also directories
- t time: keep time stamps of transferred files
- v verbose: print information during transfer
- z zip: compressed file transfer (faster for slow connections)
- c checksum: use check sums (instead of time stamps)
for comparison

Copy files via konqueror from other hosts

konqueror allows to show directories of remote hosts with help of the `fish` protocol. So, enter in the address bar, e.g.,

```
fish://user@weber.stud.physik.uni-potsdam.de
```


- `df -h` shows free space on hard drive
- `du -hs` shows total size of current directory
- `ps ux` shows running processes of the current user
- `top` shows load and running processes (interactive)
- `htop`
- `kill -9 PID` “kills” the process with the given process ID (PID)

The resources of the stud cluster (CPUs, RAM, disk space) can be used by all users, the users share these resources.

→ Therefore, please, think of other users:

- **Log out**, when leaving the computer, do not just lock the screen. Never switch off/shutdown the computer.
- Have an eye on the **disk usage of you home directory** (see below), delete regularly data that are no longer required.
- If you intend to start a **job that will run a bit longer**, then *nice* this job (see below).

Nice and renice jobs/processes

The command `top` shows the priority and the consumption of resources of running processes:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
26054	htodt	39	19	1103308	179636	8648	R	100,0	0,552	0:28.93	53_steal.exe
14763	htodt	20	0	1749032	358808	74328	S	3,322	1,102	12:23.34	Xvnc

Jobs that might run longer than just a few minutes and put some load on the CPU, should be niced when started, e.g.:

```
nice -19 ./53_steal.exe
```

→ The priority is decreased from 20 (default) by 19 to 39 (higher values mean actually lower priority).

A job can also be niced when already running with help of `renice` and the process ID (PID), e.g.:

```
renice +19 26054
```

The program `top` in its interactive mode can also renice a process by pressing the key `r`.

Checking disk space

The command `df -hT` shows an overview of the available and used disk space on the current host:

```
weber/htodt> df -hT
```

Filesystem	Type	Size	Used	Avail	Use%	Mounted on
/dev/sdb1	xfs	3,7T	981G	2,7T	27%	/home
bell:/home/bell	nfs4	1,8T	1,1T	702G	60%	/nfs/bell

Moreover, the command `du -hs ~` displays the disk usage of your own home directory.

```
weber/htodt> du -hs ~
```

```
30G /home/weber/htodt
```

Instead of the tilde `~` you can also use other (own) subdirectories as an argument, to check their disk usage.

→ If a disk shows a use of 100%, you cannot longer write on this disk or copy data to it.

Show CPU performance

The type, its parameters, and its current clock rate(s) of the installed CPU are shown in the file `/proc/cpuinfo`, which you can read with help of `cat` or `less` (info is duplicated for each thread/logical core):

```
weber/htodt>cat /proc/cpuinfo
model name      : Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
cpu MHz        : 3491.946
```

The number of logical cores/threads (= either number of physical cores or number of physical cores $\times 2$ for Hyperthreading) can be also seen in the program `top`, if you press the key `1`.

Show available RAM

The command `free -h` shows the amount of free/available RAM:

```
weber/htodt> free -h
```

	total	used	free	shared	buff/cache	available
Mem:	187Gi	66Gi	51Gi	3,2Gi	69Gi	116Gi