

The Rough Road to Real-Life Reproducible Research

Christophe Pouzat, Arnaud Legrand, Konrad Hinsén

3 septembre 2018

Outline

M4-S0 : The Rough Road to Real-Life Reproducible Research

M4-S1 : Data Hell

M4-S2 : Software Hell

M4-S3 : Numerics Hell

M4-S4 : Conclusion

Where are we?

M4-S0 : The Rough Road to Real-Life Reproducible Research

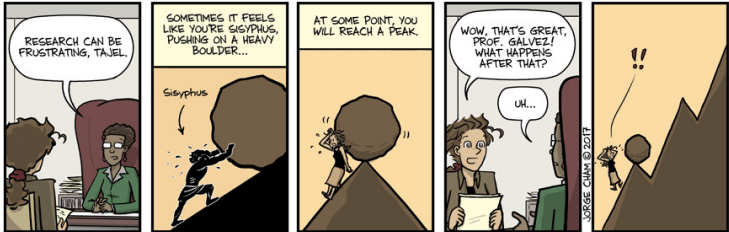
M4-S1 : Data Hell

M4-S2 : Software Hell

M4-S3 : Numerics Hell

M4-S4 : Conclusion

Reproducible Research Hell



Module 4. The Rough Road to Real-Life Reproducible Research

1. Data Hell
2. Software Hell
3. Numerics Hell
4. Conclusion

Where are we?

M4-S0 : The Rough Road to Real-Life Reproducible Research

M4-S1 : Data Hell

M4-S2 : Software Hell

M4-S3 : Numerics Hell

M4-S4 : Conclusion

Two new problems

When we start to work on real data, we typically have to deal with two problems :

- ▶ the data are of diverse nature
- ▶ the data occupy a lot of memory

Non-homogeneous data

- ▶ The influenza-like illness data from module can easily be presented as a table (2 dimensional object)
- ▶ Often the table form must be **abandoned** because
 - ▶ the columns don't all have the same length
 - ▶ the data can be a time series **and** a set of images, etc.

Big data

- ▶ **Text formats** are not always appropriate for numbers
- ▶ Choice of a **binary format** because
 - ▶ Numbers occupy less memory
 - ▶ Numbers in text format must be converted to binary anyway for computation

Text format features we wish to keep : metadata

- ▶ Text permits storing the data **and** all the rest. . .
- ▶ ⇒ add information about the data :
 - ▶ provenance
 - ▶ recording date
 - ▶ source
 - ▶ etc.
- ▶ This information about the data is what is called **metadata**
- ▶ They are vital for doing reproducible research

Text format features we wish to keep : endianness

- ▶ Text format is universal
- ▶ Binary formats depend on hardware architecture and operating system
- ▶ The four-bit sequence 1010 can be read as
 - ▶ $1 \times 1 + 0 \times 2 + 1 \times 4 + 0 \times 8 = 5$, which is **little-endianness**
 - ▶ $1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 = 10$, which is **big-endianness**
- ▶ A binary storage for reproducible research must specify endianness

Binary formats for composite data allow storing metadata

Wanted : binary formats for

- ▶ working with big datasets of diverse nature
- ▶ storing metadata along with the data
- ▶ having endianness fixed **once and for all**

'FITS' and 'HDF5'

- ▶ The [Flexible Image Transport System](#) ('FITS'), developed in 1981 and still regularly updated
- ▶ The [Hierarchical Data Format](#) ('HDF'), developed at the [National Center for Supercomputing Applications](#), is at its fifth version, 'HDF5'

'FITS'

- ▶ 'FITS' introduced and updates by the astrophysics community
- ▶ Format sufficiently general for use in different contexts

The anatomy of a 'FITS' file

- ▶ One or more segments : **Header/Data Units** (HDUs)
- ▶ A HDU is made up of :
 - ▶ a header (**Header Unit**) followed **optionally** by
 - ▶ the data (**Data Unit**)
- ▶ Header = key-value pair → **metadata**
- ▶ Data stored as binary tables (one to 999 dimensions) or as tables (text or binary)

Manipulation of 'FITS' files

- ▶ The developers of the format offer a 'C' library and associated programs that are easy to use
- ▶ 'PyFITS' for Python users
- ▶ 'FITSio' for R users

'HDF5'

- ▶ Hierarchical organization, resembles a filesystem tree
- ▶ Structuring element : a **group** (similar to a directory) contains one or more **datasets**
- ▶ **Groups** can be nested
- ▶ No structure imposed on metadata
- ▶ No structure imposed on data - they can be text

Manipulation of 'HDF5' files

- ▶ More flexible format \Rightarrow the 'C' library is more complex than its 'FITS' equivalent
- ▶ The library is distributed with 'HDFView', a powerful tool for exploring and visualizing data
- ▶ 'h5py' is a very complete 'Python' interface
- ▶ Three 'R' packages : 'h5', 'hdf5r' et 'rhdf5'

Archiving

Git (hub, lab, ...) : not well suited for data storage



Conclusions

- ▶ Real data \Rightarrow size and structure problems
- ▶ Read data are complex \Rightarrow metadata
- ▶ 'FITS' and 'HDF5' = **practical** solutions
- ▶ In terms of complexity and flexibility : 'FITS' < 'HDF5'
- ▶ Archiving platforms \Rightarrow persistent storage accessible for everyone

Where are we?

M4-S0 : The Rough Road to Real-Life Reproducible Research

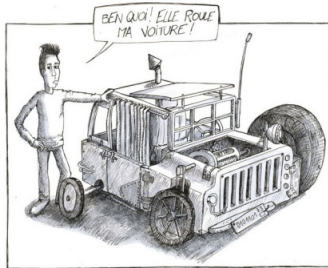
M4-S1 : Data Hell

M4-S2 : Software Hell

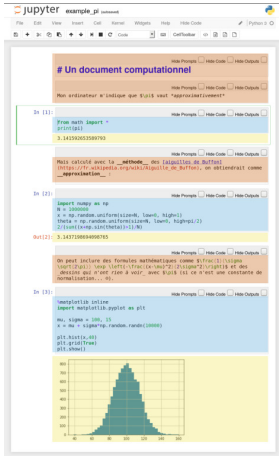
M4-S3 : Numerics Hell

M4-S4 : Conclusion

Scaling up



Complex code. . .



The screenshot shows a Jupyter Notebook window titled "exemple_jf - notebook". The notebook contains several cells:

- A title cell: "# Un document computationnel".
- A text cell: "Mon ordinateur n'indique que 7, il s'agit tout 'approximativement'".
- A code cell (In [1]):

```
from math import *\nprint(pi)
```

Output: 3.141592653589793
- A text cell: "Mais calculé avec la _methode_ des tangentes de Buffon!" with a URL: https://fr.wikipedia.org/wiki/Tangente_de_Buffon, and a note: "on obtiendrait comme _approximation_".
- A code cell (In [2]):

```
import numpy as np\nN = 100000\ns = np.random.uniform(size=N, low=0, high=1)\nx = np.random.uniform(size=N, low=0, high=np.pi/2)\nz = 2 * (s * np.sin(2 * x))
```

Output: 3.1437198894888765
- A text cell: "On peut inclure des formules mathématiques comme $\frac{1}{2} \frac{d}{dx} x^2 = x$ et des choses qui n'ont rien à voir, avec tout ça ce n'est une constante de normalisation...".
- A code cell (In [3]):

```
import matplotlib\nimport matplotlib.pyplot as plt\n\nmu, sigma = 100, 15\nx = mu + sigma * np.random.randn(10000)\n\nplt.hist(x, 40)\nplt.grid(True)\nplt.show()
```

The final cell displays a histogram of the generated data, showing a bell-shaped distribution centered around 100.

- ▶ A real spaghetti bowl
 - ▶ No global view
 - ▶ Interaction between multiple languages = danger

Complex code...

The screenshot shows a JupyterLab notebook with a title bar that reads "analyze-syndrome-grippal" and "surveillance-epi". The main content area contains a large R script. At the top, there is a header in French: "Incidence du syndrome grippal". Below this, there is a multi-line R script that includes data loading, variable assignment, and a large table of data. The table has many columns, including dates and various numerical values. The script ends with a call to `write.csv` to save the data to a file.

This screenshot shows a continuation of the R script from the previous notebook. It features a large table with columns for dates (e.g., "2010-01-01", "2010-02-01") and various numerical values. The script continues with more data processing and a final `write.csv` command. The interface includes a toolbar at the top with options like "Run", "Clear", and "Save".

This screenshot shows the final part of the R script, which includes several plots. The first plot is a line graph showing a time series of data. The second plot is a bar chart with the title "Espace de l'incidence annuelle". Below the plots, there is more R code and a final plot at the bottom of the page. The interface shows the same JupyterLab toolbar and navigation options.

- ▶ A real spaghetti bowl
 - ▶ No global view
 - ▶ Interaction between multiple languages = danger

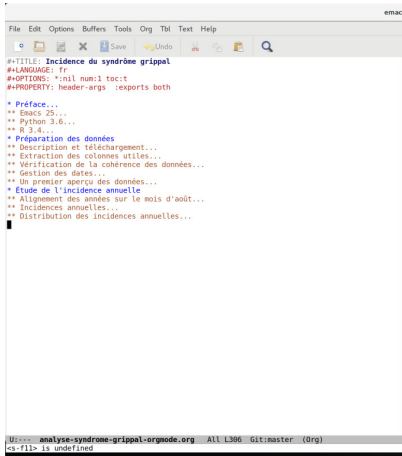
Complex code. . .

The collage consists of several panels:

- Estimating Color Names by Web Image Searches:** A Python script using BeautifulSoup and color analysis libraries.
- Chromatid plots and chromosomal interactions:** Two colorful plots showing interactions between chromosomes.
- Heatmap of correlation matrix:** A square heatmap with a diagonal line, showing relationships between variables.
- Scatter plot of training sample number:** A plot with 'Number of samples' on the y-axis and 'Number of features' on the x-axis, showing a positive correlation.
- Code snippets:** Various blocks of code in Python, R, and JavaScript, some with comments and function definitions.

- ▶ A real spaghetti bowl
 - ▶ No global view
 - ▶ Interaction between multiple languages = danger

Complex code. . .



```
emacs
File Edit Options Buffers Tools Org Tbl Text Help
[Icons] Save Undo [Icons]
#*=TITLE: Incidence du syndrome grippal
#*=LANGUAGE: fr
#*=OPTIONS: *nil num:1 toc:t
#*=PROPERTY: header-args :exports both

* Préface...
** Emacs 25...
** Python 3.6...
** R 3.4...
* Préparation des données
** Description et téléchargement...
** Extraction des colonnes utiles...
** Vérification de la cohérence des données...
** Gestion des dates...
** Un premier aperçu des données...
* Étude de l'incidence annuelle
** Alignement des années sur le mois d'août...
** Incidences annuelles...
** Distribution des incidences annuelles...

U:--- analyse-syndrome-grippal-orgmode.org All L306 Git:master (Org)
<*-fill> is undefined
```

- ▶ A real spaghetti bowl
 - ▶ No global view
 - ▶ Interaction between multiple languages = danger

Complex code. . .


```
emac
File Edit Options Buffers Tools Org Tbl Text Help
[Icons] Save Undo [Icons] Search

##+TITLE: Incidence du syndrome grippal
##+LANGUAGE: ff
##+OPTIONS: *nil num:1 toc:1
##+PROPERTY: header-args :exports both

* Préface...
** Emacs 25...
** Python 3.6...
** R 3.4...
* Préparation des données
** Description et téléchargement...
** Extraction des colonnes utiles...
** Vérification de la cohérence des données...
** Gestion des dates...
** Un premier aperçu des données
Nous passons au langage R pour avoir un résumé statistique de notre jeu de données.
##+BEGIN_SRC R :results output :var data=date-inc-sorted
data$date <- as.Date(data$date)
summary(data)
##+END_SRC

##+RESULTS:
:      date      inc
: Min. :1984-12-31 Min. : 0
: 1st Qu.:1993-02-25 1st Qu.: 5164
: Median :2001-04-16 Median : 16188
: Mean   :2001-04-15 Mean   : 63053
: 3rd Qu.:2009-06-04 3rd Qu.: 49576
: Max.   :2017-07-24 Max.   :1801824

Regardons enfin à quoi ressemblent nos données !
##+BEGIN_SRC R :results output graphics :file inc-plot.png :var data=date-inc-sorted
data$date <- as.Date(data$date)
plot(data, type="l", xlab="Date", ylab="Incidence hebdomadaire")
##+END_SRC

##+RESULTS:

U:--- analyse-syndrome-grippal-orgnode.org Top L1 Git:master (Org)
Beginning of buffer
```

- ▶ A real spaghetti bowl
- ▶ No global view
- ▶ Interaction between multiple languages = danger

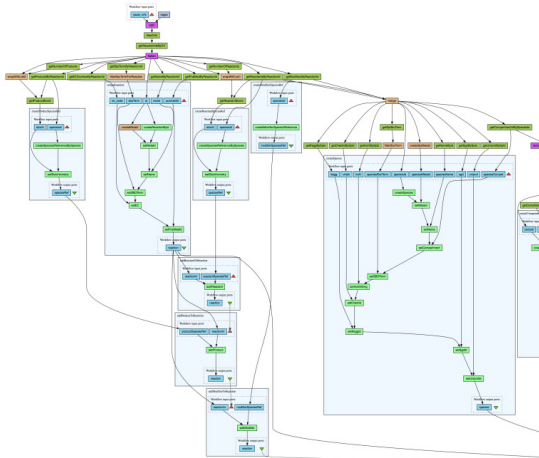
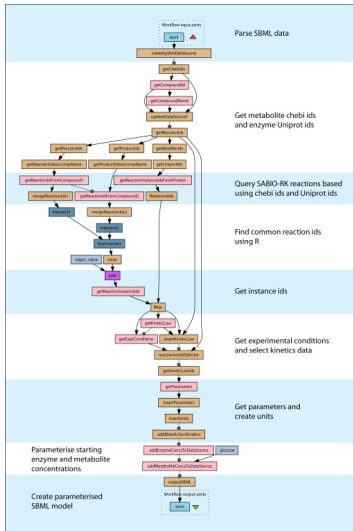
Complex code...

The image displays five screenshots of a complex code repository, illustrating a 'spaghetti bowl' of code. The screenshots show:

- 1. A large block of C++ code with many lines of text.
- 2. A table with columns for 'Name', 'Age', 'Sex', 'Height', 'Weight', 'Blood Pressure', and 'Cholesterol'.
- 3. A code snippet in a language like Python or JavaScript.
- 4. A table with columns for 'Name', 'Age', 'Sex', 'Height', 'Weight', 'Blood Pressure', and 'Cholesterol'.
- 5. A table with columns for 'Name', 'Age', 'Sex', 'Height', 'Weight', 'Blood Pressure', and 'Cholesterol'.

- ▶ A real spaghetti bowl
 - ▶ No global view
 - ▶ Interaction between multiple languages = danger

... that is difficult to orchestrate



... that is difficult to orchestrate

Workflows :

- ▶ Clearer high-level view
- ▶ Composition of codes and data movement made explicit
- ▶ Safer sharing, reusing, and execution
- ▶ Notebooks are a variant that is both impoverished and richer
- ▶ No simple/mature path from a notebook to a workflow

Examples :

- ▶ Galaxy, Kepler, Taverna, Pegasus, Collective Knowledge, VisTrails
- ▶ Light-weight : dask, drake, swift, snakemake, ...
- ▶ Hybrids : SOS-notebook, ...

The mess of expensive computations

Long-running computations and big datasets

- ▶ JupyterHub and supercomputers : under development
- ▶ Checkpoints and caching
- ▶ Workflows permit scaling up

Complex ecosystems

What is hiding behind the simple

1

```
import matplotlib
```

Package: python3-matplotlib

Version: 2.1.1-2

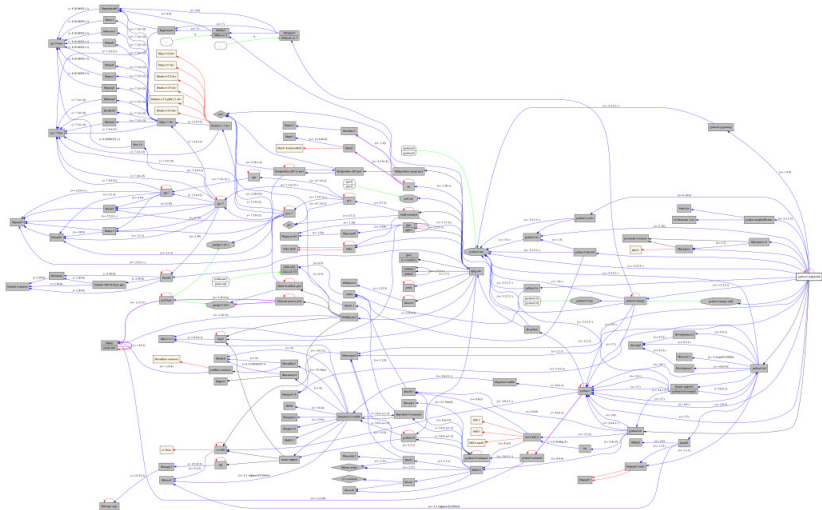
Depends: python3-dateutil, python-matplotlib-data (>= 2.1.1-2),
python3-pyparsing (>= 1.5.6), python3-six (>= 1.10), python3-tz,
libjs-jquery, libjs-jquery-ui, python3-numpy (>= 1:1.13.1),
python3-numpy-abi9, python3 (<< 3.7), python3 (>= 3.6~),
python3-cycler (>= 0.10.0), python3:any (>= 3.3.2-2~), libc6 (>=
2.14), libfreetype6 (>= 2.2.1), libgcc1 (>= 1:3.0), libpng16-16 (>=
1.6.2-1), libstdc++6 (>= 5.2), zlib1g (>= 1:1.1.4)

Complex ecosystems

What is hiding behind the simple

1

```
import matplotlib
```



Complex ecosystems

No standard :

- ▶ Linux (apt, rpm, yum), MacOS X (brew, McPorts, Fink), Windows (?)
- ▶ Neither for installation nor for retrieving the information... 😞

```
1 import sys
2 print(sys.version)
3 import matplotlib
4 print(matplotlib.__version__)
5 import pandas as pd
6 print(pd.__version__)
```

```
3.6.3 (default, Oct 3 2017, 21:16:13)
[GCC 7.2.0]
2.1.1
0.20.3
```

```
1 library(ggplot2)
2 sessionInfo()
```

```
R version 3.4.3 (2017-11-30)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Debian GNU/Linux buster/sid
```

Matrix products: default

```
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
```

locale:

```
[1] LC_CTYPE=fr_FR.UTF-8      LC_NUMERIC=C
[3] LC_TIME=fr_FR.UTF-8      LC_COLLATE=fr_FR.UTF-8
[5] LC_MONETARY=fr_FR.UTF-8  LC_MESSAGES=fr_FR.UTF-8
```

other attached packages:

```
[1] ggplot2_2.2.1
```

loaded via a namespace (and not attached):

```
[1] colorspace_1.3-2 scales_0.5.0 compiler_3.4.3 laz
[5] plyr_1.8.4 pillar_1.1.0 gtable_0.2.0 tib
[9] Rcpp_0.12.15 grid_3.4.3 rlang_0.1.6 mun
```

Controlling one's environment

A controlled environment :

- ▶ Work in a virtual machine (heavy) or a Docker container (light)

Preserve the mess

- ▶ Automatic capture of the environment
- ▶ CDE, ReProZip, CARE

Cleaning up

- ▶ Start with a clean environment
- ▶ Install only what's strictly necessary (and document it)
- ▶ Docker, Singularity, Guix, Nix, ...

The test of time



aleksandrmarkin@geg

Backwards compatibility

▶ Python and its rapidly evolving environment

```
1 python2 -c "print(10/3)"  
2 python3 -c "print(10/3)"
```

```
3  
3.3333333333333335
```

Backwards compatibility

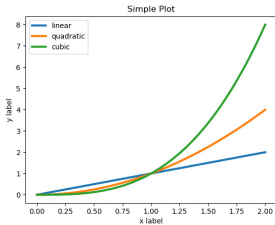
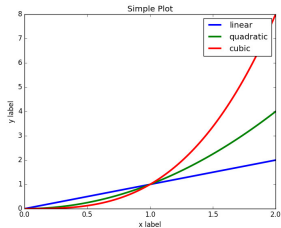
► Python and its rapidly evolving environment

```
1 python2 -c "print(10/3)"
```

```
2 python3 -c "print(10/3)"
```

```
3
```

```
3.3333333333333335
```

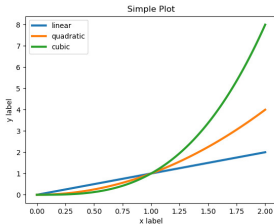
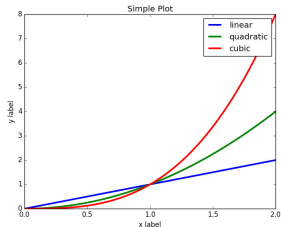


Backwards compatibility

▶ Python and its rapidly evolving environment

```
1 python2 -c "print(10/3)"  
2 python3 -c "print(10/3)"
```

```
3  
3.3333333333333335
```



- ▶ Cortical Thickness Measurements (PLOS ONE, June 2012) : *FreeSurfer* : differences were found between the Mac and HP workstations and between Mac OSX 10.5 and OSX 10.6.
- ▶ Format incompatibility between orgmode 7.*, 8.*, 9.*, etc.

Rapid development tools

Rapid but also **fragile** and **unstable** :

- ▶ Correction or introduction of bugs
- ▶ It becomes necessary to check regularly if environments can still be reconstructed and work (continuous integration, regression testing)



Popper : <http://falsifiable.us/>

Another option :

- ▶ Limit onself to what is manageable (C for example)

Archiving

Source code management

- ▶ Git (hub, lab, ...) : stable, open, ... durable?
 - ▶ ~~Google Code, Gitorious, Code Spaces~~



Software Heritage

HAL
archives-ouvertes.fr

Environment management

- ▶ Longevity of access to Docker Hub, Nix repository, Code Ocean, ... ?
- ▶ Once an environment is frozen, what's the lifetime of a virtual machine, a Docker image, ... ?

Preserve as much information as possible automatically

- ▶ Software, versions, installation procedures

Where are we?

M4-S0 : The Rough Road to Real-Life Reproducible Research

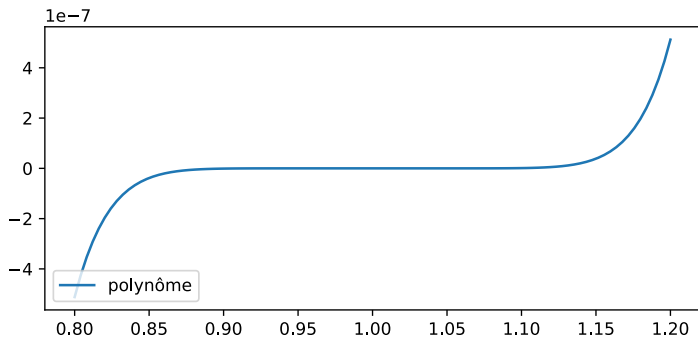
M4-S1 : Data Hell

M4-S2 : Software Hell

M4-S3 : Numerics Hell

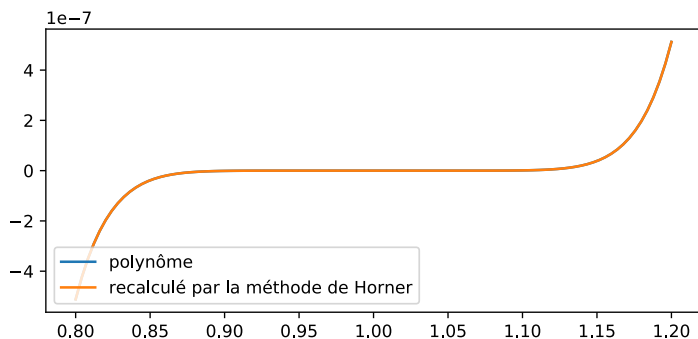
M4-S4 : Conclusion

Floating-point arithmetic



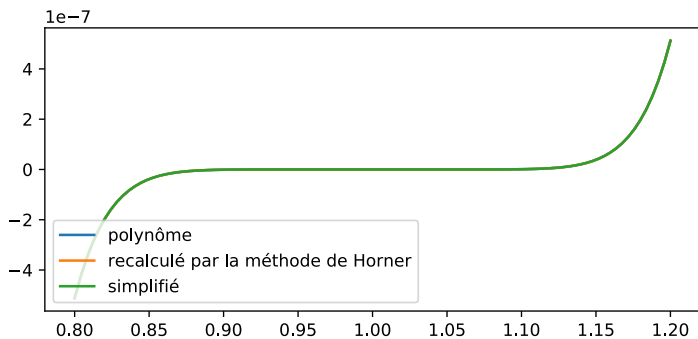
```
1 def polynome(x):  
2     return x**9 - 9.*x**8 + 36.*x**7 - 84.*x**6 + 126.*x**5 \  
3         - 126.*x**4 + 84.*x**3 - 36.*x**2 + 9.*x - 1.
```

Floating-point arithmetic



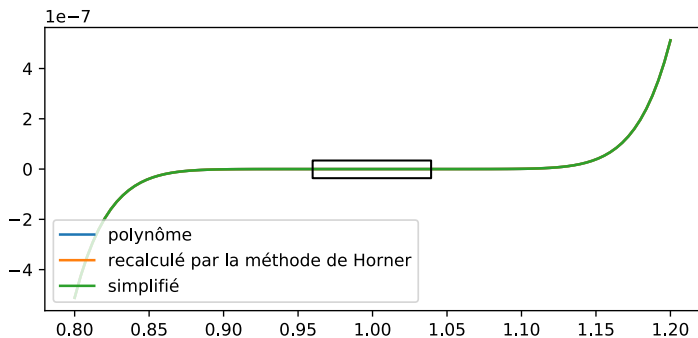
```
1 def horner(x):  
2     return x*(x*(x*(x*(x*(x*(x - 9.) + 36.) - 84.) + 126.) \  
3         - 126.) + 84.) - 36.) + 9.) - 1.
```

Floating-point arithmetic

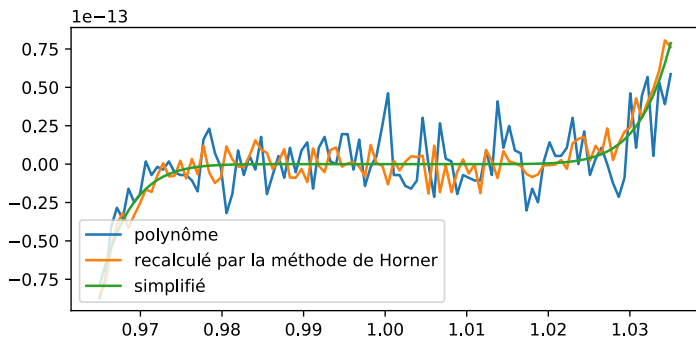


```
1 def simple(x):  
2     return (x-1.)**9  
3     # trop facile!
```

Floating-point arithmetic



Floating-point arithmetic



Rounding

- ▶ Every operation includes implicit rounding.
- ▶ $a+b$ is actually $\text{arrondi}(a+b)$.
- ▶ Unfortunately :
$$\text{arrondi}(\text{arrondi}(a+b)+c) \neq \text{arrondi}(a+\text{arrondi}(b+c)).$$
- ▶ Operation order therefore matters.

For a reproducible computation, operation order must be preserved !!!

How to explain it to my compiler ?

To speed up computations, compilers may change operation order, and thus results.

Two options for computing reproducibly :

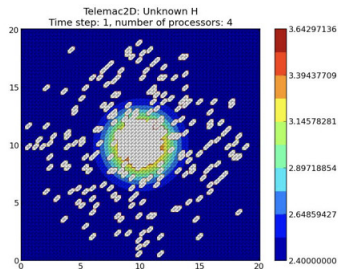
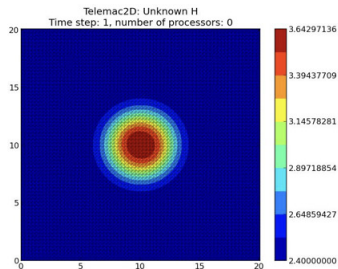
- ▶ Insist on the preservation of operation order,
 - ▶ if the language permits it.
 - ▶ Example : Module 'ieee_arithmetic' in Fortran 2003
- ▶ Make compilation reproducible :
 - ▶ Record the precise compiler version
 - ▶ Record all compilation options

Parallel computation

- ▶ Goal : get results sooner → Minimize communications between processors → Adapt data distribution → ... hence the operation order 😞
- ▶ Consequence : results depends on the number of processors !

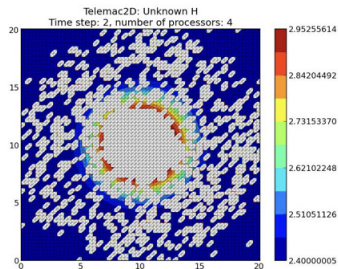
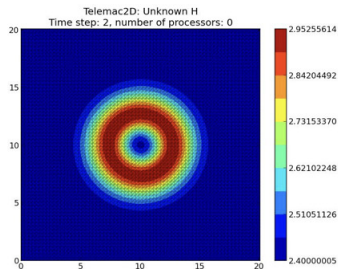
Minimizing the impact of parallelism is an active research topic.

Parallel computation : example



Source : Rafife Nheili, PhD. Thesis, University of Perpignan, 2016

Parallel computation : example



Source : Rafife Nheili, PhD. Thesis, University of Perpignan, 2016

Computing platforms

- ▶ Computing platform : hardware + infrastructure software
- ▶ Computation = platform + software + data
- ▶ The platform defines the interpretation of the software.
- ▶ Platform and software define the interpretation of the data.
- ▶ Other platform-defined aspects :
 - ▶ integer representation (16/32/64 bits)
 - ▶ error handling

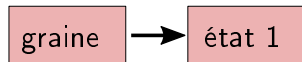
Random numbers

- ▶ Used to simulate stochastic processes.
- ▶ In practice : **pseudo-** random numbers.
- ▶ Series of numbers that **appear** to be random. . .
- ▶ . . . but are generated by a deterministic algorithm.

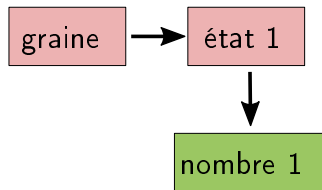
Pseudo-random number generators

graine

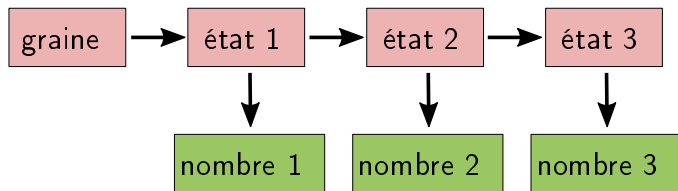
Pseudo-random number generators



Pseudo-random number generators



Pseudo-random number generators



Reproducibility in theory

- ▶ Principle : same seed + same algorithm \rightarrow same series
- ▶ The seed is often chosen as a function of time
- ▶ It must be defined in the application code

Reproducibility in practice

- ▶ Same seed + same algorithm \rightarrow same series :
not obvious with floating-point arithmetic !
- ▶ A simple trick to permit verification :
test the first values of the series.

Example : the Python language

```
1 import random
2
3 random.seed(123)
4 for i in range(5):
5     print(random.random())
```

```
0.0523635988509
0.0871866775226
0.40724176367
0.107700234938
0.901198877952
```

Example : the Python language

```
1 import random
2
3 random.seed(123)
4 assert random.random() == 0.052363598850944326
5 assert random.random() == 0.08718667752263232
6 assert random.random() == 0.4072417636703983
```

Take-home message

- ▶ The results of a numerical computation depend on
 - ▶ the software
 - ▶ the input data
 - ▶ the computing platform : hardware, compilers, ...
- ▶ Platform influence is important for floating-point arithmetic. Record all parameters on which your results may depend :
 - ▶ compiler version, compilation options
 - ▶ hardware (processor type, GPU, ...)
 - ▶ number of processors
- ▶ When using a random number generator, define your own seed and verify the first elements of the series.

Where are we?

M4-S0 : The Rough Road to Real-Life Reproducible Research

M4-S1 : Data Hell

M4-S2 : Software Hell

M4-S3 : Numerics Hell

M4-S4 : Conclusion

The take-home message of the MOOC

A major concern

- ▶ Scientific method
- ▶ Inspectability and reusability

Tools exist

- ▶ Computational documents and workflows, version control and archives, software environments, continuous integration, . . .
- ▶ These tools evolve constantly
 - ▶ Choose those that are best adapted to your context
 - ▶ Find a compromise between modern and durable tools

Use in practice, don't get discouraged !

- ▶ Takes notes rigorously
- ▶ Make information useable and accessible
- ▶ Improve in small steps