Reproducible Computational Experiments Using SCons

Sergey Fomel

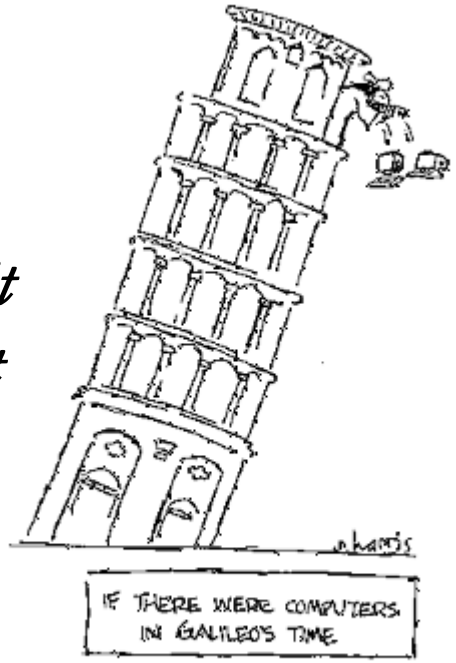
University of Texas at Austin

Gilles Hennenfent

University of British Columbia

Problem

- **How to publish computational results?**
  - **scientific publication**
  - **technology transfer**
- *Within the world of science, computation is now rightly seen as a third vertex of a triangle complementing experiment and theory. However, as it is now often practiced, one can make a good case that computing is the last refuge of the scientific scoundrel.*
  – Randall LeVeque



Outline

- Reproducible research at Stanford
  - lessons
- Reproducible computational experiments
  - Implementation in MADAGASCAR
  - test-driven development using SCons
- Road to the future
  - community effort

Reproducible Research at Stanford

- **Donald Knuth**
  - literate programming
  - A computer program should be written with human readability as a primary goal.

- **Jon Claerbout**
  - reproducible research
  - The purpose of reproducible research is to facilitate someone going a step further by changing something.

- **David Donoho**
  - reproducible research using Matlab
  - An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship.

Lessons of Reproducible Research

- Motivation
  - scientific integrity
  - robust software development
  - technology transfer
- Tools
  - computational experiments
  - publication
- Maintenance
  - test-driven development

Implementation in MADAGASCAR

- Open-source package (GPL)
  - released in 2006
  - 16 developers
- Three levels
  - command-line modules
    - C, Fortran, Python, Matlab
  - signal processing scripts
    - SCons
  - publications
    - LaTeX + latex2html + SCons

http://rsf.sourceforge.net/

Monday	Get an idea
Tuesday	Implement it
Wednesday	Test it
Thursday	Communicate it
Friday	Apply it

SCons (Software Construction)

- Python-based replacement for “make”
  - reliable, automatic, and extensible dependency analysis
  - winner of the Software Carpentry competition
  - configuration files are Python scripts
- Features
  - support for different programming languages
  - support for parallel builds.
  - configuration support analogous to autoconf
  - cross-platform
  - open-source

http://www.scons.org

SConstruct File for Compilation

```
Program('program', ['main.c', 'file1.c', 'file2.c'])
```

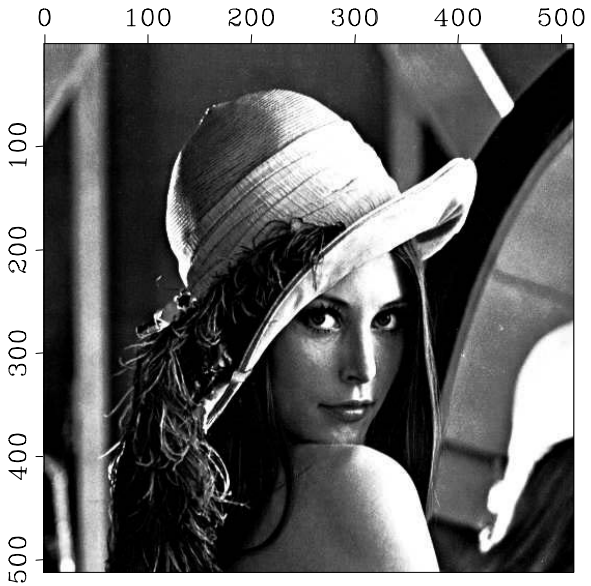
```
bash$ scons
scons: Building targets ...
cc -c -o file1.o file1.c
cc -c -o file2.o file2.c
cc -c -o main.o main.c
cc -o program main.o file1.o file2.o
scons: done cleaning targets.
```

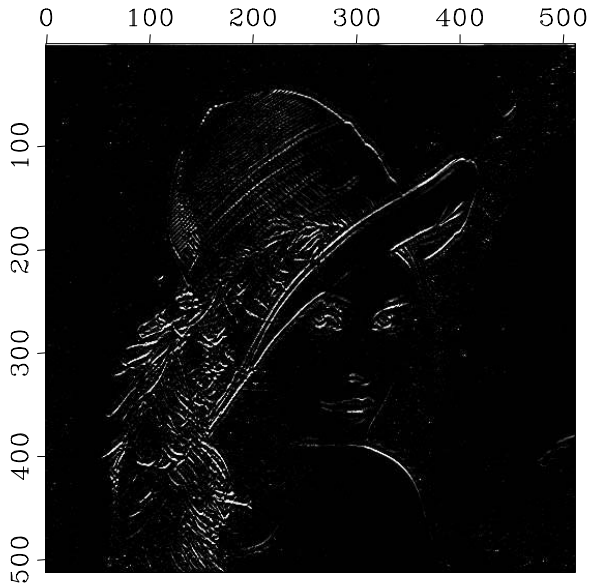
SConstruct File for Signal Processing

```
import rsfproj as p

# Download data
p.Fetch('lena.img','imgs')

# Convert to a floating-point format
p.Flow('lena','lena.img',
       '''echo n1=512 n2=513 in=$SOURCE
       data_format=native_uchar | transp |
       window f1=1 | dd type=float''')

# Bandpass filtering
p.Flow('band','lena','bandpass flo=0.1')

# Plotting
for img in ('lena','band'):
    p.Plot(img,'grey allpos=y wanttitle=n screenratio=1')
```

Execution

```
bash$ scons
scons: Building targets ...
retrieve(["lena.img"], [])
< lena.img echo n1=512 n2=513 in=lena.img data_format=native_uchar | \
 /path/transp | /path/window f1=1 | /path/dd type=float > lena.rsf
< lena.rsf /path/bandpass flo=0.1 > band.rsf
< band.rsf /path/grey allpos=y wanttitle=n screenratio=1 > band.vpl
< lena.rsf  /path/grey allpos=y wanttitle=n screenratio=1 > lena.vpl
scons: done building targets.
```





Experimentation

```
bash$ sed s/flo=0.1/flo=0.2/ < SConstruct > SConstruct2
bash$ mv SConstruct2 SConstruct
bash$ scons
scons: Building targets ...
< lena.rsf /path/bandpass flo=0.2 > band.rsf
< band.rsf /path/grey allpos=y wanttitle=n screenratio=1 > band.vpl
scons: done building targets.
```

```
bash$ sed s/wanttitle=n/title=Lena/ < SConstruct > SConstruct2
bash$ mv SConstruct2 SConstruct
bash$ scons
scons: Building targets ...
< band.rsf /path/grey allpos=y title=Lena screenratio=1 > band.vpl
< lena.rsf  /path/grey allpos=y title=Lena screenratio=1 > lena.vpl
scons: done building targets.
```

SConstruct File for Publication

```
import rsftex as t

# Compile paper from LaTeX source in icassp.tex
t.Paper('icassp',options='12pt',use='hyperref,amsmath')
```

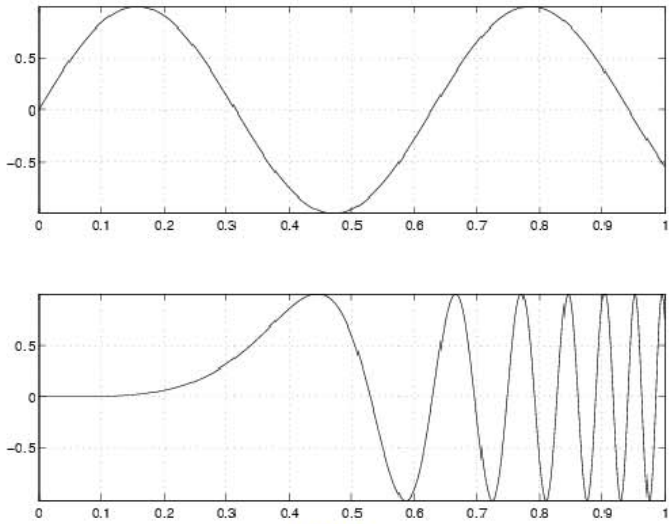
```
bash$ scons icassp.pdf
...
bash$ scons icassp.html
...
bash$ scons icassp.install
...
```

http://rsf.sourceforge.net/Reproducible_Documents

bandpassed, as is often the case with seismic data, with the largest frequency present in the data ($f_{\max}$) smaller than the Nyquist frequency given by the sampling rate (f_{Ny}). Thus, we can replace in our calculations Δx with

$$\Delta x_{\max} = \frac{1}{2 f_{\max}},$$

which will result in a $\Delta y_{\max}$ larger than that computed using Δx, the sampling rate in the x space.



strali

Figure 2. Illustration of how aliasing can occur while stretching: if the same sampling rate is used for the y-space (lower plot) as for the x-space (upper plot), serious aliasing will occur when transforming back to x-space. This will not happen if the sampling rate in the y-space is smaller than or equal to $\Delta y_{\max}$

In order to compute $\Delta y_{\max}$, we will consider two points in the x space, as seen in Fig. 2, such as

$$x_b = x_a + \Delta x_{\max} \tag{7}$$

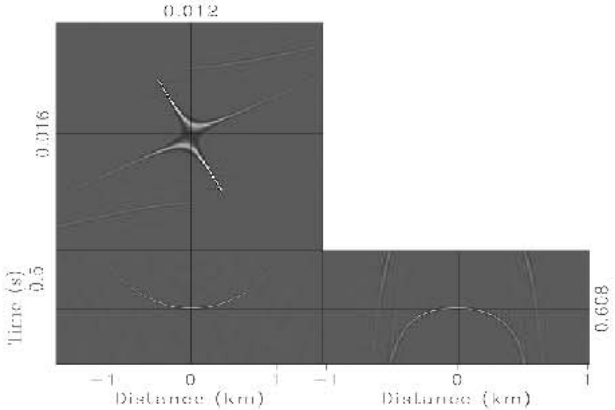
$$\Phi_j = \begin{cases} 0, & for\ \vec{k}\cdot\vec{h}=0 \\ \vec{k}\cdot\vec{h}, & for\ \Omega=0 \\ \frac{\Omega}{2}\left\{\sqrt{1+\left(\frac{2\vec{k}\cdot\vec{h}}{\Omega}\right)^2}-1-\ln\left[\frac{\sqrt{1+\left(\frac{2\vec{k}\cdot\vec{h}}{\Omega}\right)^2}+1}{2}\right]\right\} & otherwise \end{cases} , \tag{4}$$

$$where\ \vec{k}\cdot\vec{h} = k_x h_x + k_y h_y \tag{5}$$

and j can take the values 1 or 2. The frequency domain variables must have incorporated in their value a 2π constant (they are defined according to equation (2))

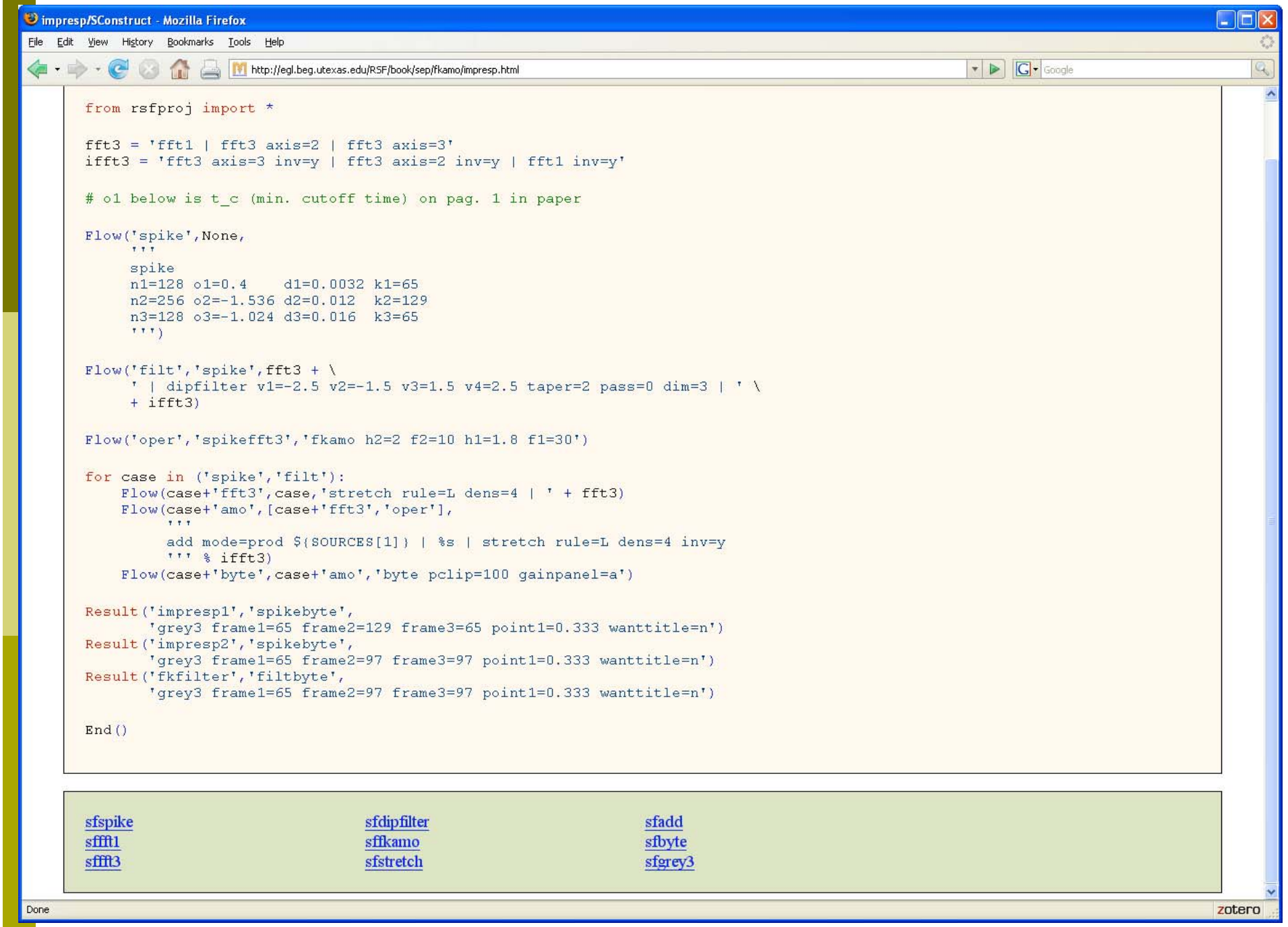
4. Do reverse 3D FFT in order to obtain the $P(\tau, m_x, m_y)|_{\vec{h}_2}$ cube.
5. Do reverse log stretch along the time axis and affix to the top of the cube the slices from times smaller than t_c. The final result is a $P(t, m_x, m_y)|_{\vec{h}_2}$ cube.

Figure 1 shows the impulse response of the above described AMO.



impresp1

Figure 1. AMO impulse response


impresp/SConstruct - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://egl.beg.utexas.edu/RSF/book/sep/fkamo/impresp.html
Google
from rsfproj import *
fft3 = 'fft1 | fft3 axis=2 | fft3 axis=3'
ifft3 = 'fft3 axis=3 inv=y | fft3 axis=2 inv=y | fft1 inv=y'
# o1 below is t_c (min. cutoff time) on pag. 1 in paper
Flow('spike',None,
'''
spike
n1=128 o1=0.4 d1=0.0032 k1=65
n2=256 o2=-1.536 d2=0.012 k2=129
n3=128 o3=-1.024 d3=0.016 k3=65
''')
Flow('filt','spike',fft3 + \
' | dipfilter v1=-2.5 v2=-1.5 v3=1.5 v4=2.5 taper=2 pass=0 dim=3 | ' \
+ ifft3)
Flow('oper','spikefft3','fkamo h2=2 f2=10 h1=1.8 f1=30')
for case in ('spike','filt'):
Flow(case+'fft3',case,'stretch rule=L dens=4 | ' + fft3)
Flow(case+'amo',[case+'fft3','oper'],
'''
add mode=prod ${SOURCES[1]} | %s | stretch rule=L dens=4 inv=y
''' % ifft3)
Flow(case+'byte',case+'amo','byte pclip=100 gainpanel=a')
Result('impresp1','spikebyte',
'grey3 frame1=65 frame2=129 frame3=65 point1=0.333 wanttitle=n')
Result('impresp2','spikebyte',
'grey3 frame1=65 frame2=97 frame3=97 point1=0.333 wanttitle=n')
Result('fkfilter','filtbyte',
'grey3 frame1=65 frame2=97 frame3=97 point1=0.333 wanttitle=n')
End()
sfspike
sffft1
sffft3
sfdipfilter
sffkamo
sfstretch
sfadd
sfbyte
sfgrey3
Done
zotero

Future: Community Support

- Special issue journals
  - Computing in Science & Engineering
- Web 2.0 publications
  - http://www.insight-journal.org/
- Wikipedia entry for reproducible research
- Motivation
  - scientific integrity
  - robust software development
  - technology transfer