

Introduction to Madagascar

Prof. Daniel Leal Macedo

Faculdade de Geofísica - UFPa

31 de julho a 2 de agosto de 2018

MADAGASCAR

Open-source software package for
geophysical data processing and reproducible
numerical experiments.

MADAGASCAR

Open-source software package for geophysical data processing and reproducible numerical experiments.

- Standalone programs (data analysis, processing and imaging);
- A development kit (C, Fortran, Python, Matlab, Octave...);
- A framework for reproducible numerical experiments (SCons);
- A framework for scientific publications (SCons and \LaTeX);
- A collection of reproducible scientific articles;
- A collection of datasets.

MADAGASCAR

Open-source software package for geophysical data processing and reproducible numerical experiments.

- Standalone programs (data analysis, processing and imaging);
- A development kit (C, Fortran, Python, Matlab, Octave...);
- A framework for reproducible numerical experiments (SCons);
- A framework for scientific publications (SCons and \LaTeX);
- A collection of reproducible scientific articles;
- A collection of datasets.

MADAGASCAR

Open-source software package for geophysical data processing and reproducible numerical experiments.

- Standalone programs (data analysis, processing and imaging);
- A development kit (C, Fortran, Python, Matlab, Octave...);
- A framework for reproducible numerical experiments (SCons);
- A framework for scientific publications (SCons and \LaTeX);
- A collection of reproducible scientific articles;
- A collection of datasets.

MADAGASCAR

Open-source software package for geophysical data processing and reproducible numerical experiments.

- Standalone programs (data analysis, processing and imaging);
- A development kit (C, Fortran, Python, Matlab, Octave...);
- A framework for reproducible numerical experiments (SCons);
- A framework for scientific publications (SCons and \LaTeX);
- A collection of reproducible scientific articles;
- A collection of datasets.

MADAGASCAR

Open-source software package for geophysical data processing and reproducible numerical experiments.

- Standalone programs (data analysis, processing and imaging);
- A development kit (C, Fortran, Python, Matlab, Octave...);
- A framework for reproducible numerical experiments (SCons);
- A framework for scientific publications (SCons and \LaTeX);
- A collection of reproducible scientific articles;
- A collection of datasets.

MADAGASCAR

Open-source software package for geophysical data processing and reproducible numerical experiments.

- Standalone programs (data analysis, processing and imaging);
- A development kit (C, Fortran, Python, Matlab, Octave...);
- A framework for reproducible numerical experiments (SCons);
- A framework for scientific publications (SCons and \LaTeX);
- A collection of reproducible scientific articles;
- A collection of datasets.

MADAGASCAR

Open-source software package for geophysical data processing and reproducible numerical experiments.

- Standalone programs (data analysis, processing and imaging);
- A development kit (C, Fortran, Python, Matlab, Octave...);
- A framework for reproducible numerical experiments (SCons);
- A framework for scientific publications (SCons and \LaTeX);
- A collection of reproducible scientific articles;
- A collection of datasets.

Madagascar Community

- 1 Home site (www.ahay.org).
- 2 Mailing list
(<https://lists.sourceforge.net/lists/listinfo/rsf-user>).
- 3 Development blog (<http://ahay.org/blog/>).

Schedule

- ➊ Introduction to Madagascar; command lines.
- ➋ Plotting; scripting with Scons.
- ➌ Modeling with Madagascar.
- ➍ Reading / Writing SEG-Y. Reading / Writing to ASCII.
- ➎ Simple processing workflow with Madagascar.

Introduction to Madagascar

- ❶ command line usage
 - programs
 - file format
- ❷ plotting

Madagascar programs

- 'sf' prefix
- > 1000 programs
- Developed using C, C++, Fortran, Python...
- Applications
 - general data analysis
 - Seismic modeling, processing and imaging
 - visualization
- Documentation based on examples
 - self-documentation
 - on-line documentation

Madagascar programs

List of all programs

```
sfdoc -k .
```

Madagascar programs

List of all programs

```
sfdoc -k .
```

```
bash$ sfdoc -k .
```

sfwave: Rice HPCSS seismic modeling and migration.

sferf: Bandpass filtering using erf function.

sfinfill: Shot interpolation.

sfslice: Extract a slice using picked surface (usually from a stack or a semblance).

sfin: Display basic information about RSF files.

sfdmo: Kirchhoff DMO with antialiasing by reparameterization.

sfic: Imaging condition

sfradstretch: Stretch of the time axis.

sflpef: Find PEF on aliased traces.

sfmul: Add, multiply, or divide RSF datasets.

sfrefer: Subtract a reference from a grid.

sfvplotdiff: Vplot diff - see if 2 vplot files represent “identical” plots.

sflevint: Leveler inverse interpolation in 1-D.

sflorenz: Generate Lorenz attractor.

sfnoise: Add random noise to the data.

Madagascar programs

List of all programs

```
sfdoc -k .
```

List of specific programs

```
sfdoc -k keyword
```


Madagascar programs

List of all programs

```
sfdoc -k .
```

List of specific programs

```
sfdoc -k keyword
```

```
bash$ sfdoc -k inversion
```

```
sfvelinvw: Inverse velocity spectrum with interpolation by modeling from  
inversion result
```

```
sfcgscan: Hyperbolic Radon transform with conjugate-directions inversion
```

```
sfdeblur: Non-stationary deblurring by inversion
```

```
sfconjgrad: Generic conjugate-gradient solver for linear inversion
```

```
sfconjgrad: Generic conjugate-gradient solver for linear inversion with  
complex data
```

```
sfmospray: Inversion of constant-velocity nearest-neighbor inverse NMO.
```

Madagascar programs

Self-documetation

sfprog no arguments

```
bash$ sfawefd2d
NAME
    sfawefd2d
DESCRIPTION
    2D acoustic time-domain FD modeling.
SYNOPSIS
    sfawefd2d < Fwav.rsf vel=Fvel.rsf sou=Fsou.rsf rec=Frec.rsf wfl=Fwfl.rsf
> Fdat.rsf den=Fden.rsf
verb=n snap=n free=n expl=n dabc=n jdata=1 jsnap=nt nqz=sf_n(az) nqx=sf_n(ax)
oqx=sf_o(az) oqx=sf_o(ax)

COMMENTS
    4th order in space, 2nd order in time. Absorbing boundary conditions

PARAMETERS
    bool    dabc=n [y/n]    absorbing BC
    file     den=          auxiliary input file name
    bool     expl=n [y/n]    "exploding reflector"
    bool     free=n [y/n]    free surface flag
    int      jdata=1
    int      jsnap=nt
    int      nqx=sf_n(ax)
    int      nqz=sf_n(az)
```

Madagascar programs

Self-documentation

sfprog no arguments

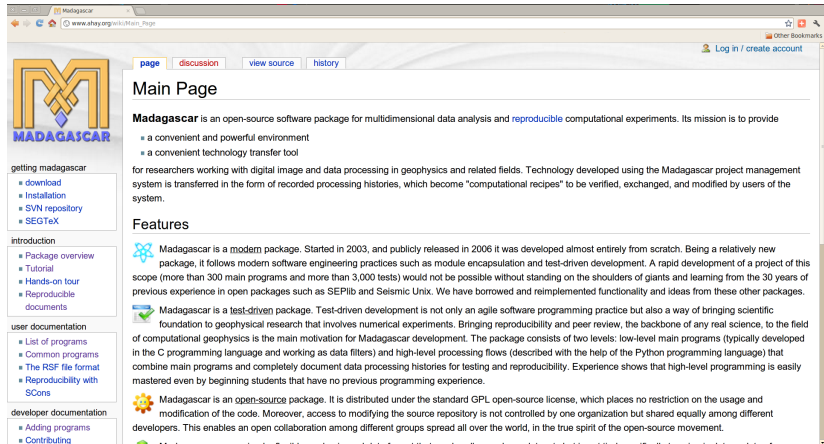
```
...
file      rec=      auxiliary input file name
bool      snap=n [y/n]      wavefield snapshots flag
file      sou=      auxiliary input file name
file      vel=      auxiliary input file name
bool      verb=n [y/n]      verbosity flag
file      wfl=      auxiliary output file name

USED IN
cwp/geo2007StereographicImagingCondition/flat4
cwp/geo2007StereographicImagingCondition/gaus1
cwp/geo2008InterferometricImagingCondition/circle
cwp/geo2008InterferometricImagingCondition/sact1
cwp/geo2008IsotropicAngleDomainElasticRTM/marm2oneA
cwp/geo2011WideAzimuthAngleDecomposition/flatEICangle
cwp/jse2006RWEImagingOverturningReflections/sigsbee
cwp/pept2011MicroearthquakeMonitoring/saf1
cwp/pept2011MicroearthquakeMonitoring/saf2
cwp/pept2011MicroearthquakeMonitoring/saf3
data/amoco/fdmod
data/marmousi/fdmod
data/marmousi2/fdMod
data/pluto/fdmod
data/sigsbee/fdmod2A
```

Madagascar programs

on-line documetation

<http://www.ahay.org>



The screenshot shows the Madagascar project website. At the top, there's a navigation bar with links for 'page', 'discussion', 'view source', and 'history'. The main heading is 'Main Page'. Below it, a paragraph describes Madagascar as an open-source software package for multidimensional data analysis and reproducible computational experiments. It lists two bullet points: 'a convenient and powerful environment' and 'a convenient technology transfer tool'. A paragraph follows, stating the system is transferred in the form of recorded processing histories, which become 'computational recipes'. The 'Features' section is highlighted with a blue icon. It contains two paragraphs: the first describes the package's origin (started in 2003, released in 2006) and its development practices (module encapsulation, test-driven development); the second describes it as a test-driven package, emphasizing reproducibility and peer review. A third paragraph, partially visible, mentions it is an open-source package under the GPL license. On the left side, there are several sidebar menus: 'getting madagascar' (with links for download, installation, SVN repository, and SEGTeX), 'introduction' (with links for package overview, tutorial, hands-on tour, and reproducible documents), 'user documentation' (with links for list of programs, common programs, the RSF file format, and reproducibility with SCons), and 'developer documentation' (with links for adding programs and contributing).

Madagascar

www.ahay.org/wiki/Main_Page

Log in / create account

page discussion view source history


Main Page


Madagascar is an open-source software package for multidimensional data analysis and [reproducible](#) computational experiments. Its mission is to provide


- a convenient and powerful environment
- a convenient technology transfer tool

for researchers working with digital image and data processing in geophysics and related fields. Technology developed using the Madagascar project management system is transferred in the form of recorded processing histories, which become "computational recipes" to be verified, exchanged, and modified by users of the system.

Features

 Madagascar is a modern package. Started in 2003, and publicly released in 2006 it was developed almost entirely from scratch. Being a relatively new package, it follows modern software engineering practices such as module encapsulation and test-driven development. A rapid development of a project of this scope (more than 300 main programs and more than 3,000 tests) would not be possible without standing on the shoulders of giants and learning from the 30 years of previous experience in open packages such as SEPlib and Seismic Unix. We have borrowed and reimplemented functionality and ideas from these other packages.

 Madagascar is a [test-driven](#) package. Test-driven development is not only an agile software programming practice but also a way of bringing scientific foundation to geophysical research that involves numerical experiments. Bringing reproducibility and peer review, the backbone of any real science, to the field of computational geophysics is the main motivation for Madagascar development. The package consists of two levels: low-level main programs (typically developed in the C programming language and working as data filters) and high-level processing flows (described with the help of the Python programming language) that combine main programs and completely document data processing histories for testing and reproducibility. Experience shows that high-level programming is easily mastered even by beginning students that have no previous programming experience.

 Madagascar is an [open-source](#) package. It is distributed under the standard GPL open-source license, which places no restriction on the usage and modification of the code. Moreover, access to modifying the source repository is not controlled by one organization but shared equally among different developers. This enables an open collaboration among different groups spread all over the world, in the true spirit of the open-source movement.

getting madagascar

- [download](#)
- [Installation](#)
- [SVN repository](#)
- [SEGTeX](#)

introduction

- [Package overview](#)
- [Tutorial](#)
- [Hands-on tour](#)
- [Reproducible documents](#)

user documentation

- [List of programs](#)
- [Common programs](#)
- [The RSF file format](#)
- [Reproducibility with SCons](#)

developer documentation

- [Adding programs](#)
- [Contributing](#)

Command-line usage

Single Program

```
[< in.rsfsfprog [par1=] [par2=] [...] [> out.rsfsf]
```

- Single input: <in.rsfsf
- Single output: >out.rsfsf
- Multiple parameters: par=val

multiple Programs

```
[< in.rsfsfprog1 [par=] | ... | sfprogn [par=] [> out.rsfsf]
```

- ONE task per program
- Data passed through pipes

Command-line usage - Example

```
bash$ cd /path/to/madagascar/school/directory/1-Intro
```

Command-line usage - Example

```
bash$ cd /path/to/madagascar/school/directory/1-Intro  
bash$ ls
```

Command-line usage - Example

```
bash$ cd /path/to/madagascar/school/directory/1-Intro
```

```
bash$ ls
```

```
g.asc plot
```


Command-line usage - Example

```
bash$ cd /path/to/madagascar/school/directory/1-Intro
```

```
bash$ ls
```

```
g.asc plot
```

```
bash$ sfspike
```

Command-line usage - Example

```
bash$ cd /path/to/madagascar/school/directory/1-Intro
```

```
bash$ ls
```

g.asc plot

```
bash$ sfspike
```

NAME

sfspike

DESCRIPTION

Generate simple data: spikes, boxes, planes, constants.

SYNOPSIS

```
sfspike < in.rsfsf > spike.rsfmag= nsp=1 k#[0,...] l#[k1,k2,...] p#[0
```

COMMENTS

Spike positioning is given in samples and starts with 1.

PARAMETERS

float d#[0.004,0.1,0.1,...] sampling on #-th axis

ints k#[0,...] spike starting position [nsp]

ints l#[k1,k2,...] spike ending position [nsp]

RSF data structure

- $n1 = 15;$
- $o1 = 0;$
- $d1 = 2;$

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28
0	4	16	64	100	144	...								

$$f(x) = x^2$$

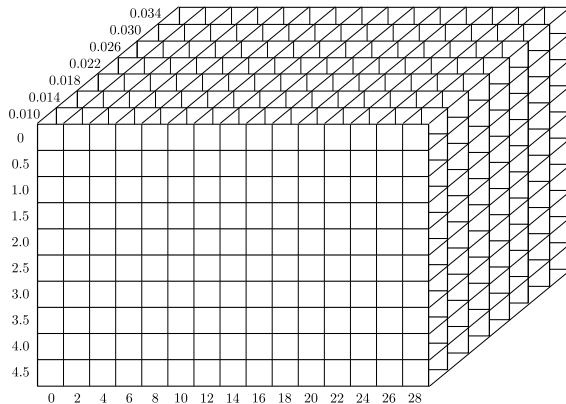
RSF data structure

	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1.0	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28
1.5	⋮	⋮													
2.0	⋮	⋮													
2.5															
3.0															
3.5															
4.0															
4.5															

- $n1 = 15$; $n2 = 10$;
- $o1 = 0$; $o2 = 0$;
- $d1 = 2$; $d2 = 0.5$;

$$f(x, z) = xz$$

RSF data structure



- $n1 = 15;$
 $n2 = 10;$
 $n3 = 7;$
- $o1 = 0;$
 $o2 = 0;$
 $o3 = 0.010;$
- $d1 = 2;$
 $d2 = 0.5;$
 $d3 = 0.004;$

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 > a.rs
```

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 > a.rs
```

- standard in: none
- standard out: a.rs

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 > a.rs
```

- standard in: none
- standard out: a.rs

```
bash$ ls
```


Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 > a.rs
```

- standard in: none
- standard out: a.rs

```
bash$ ls
```

a.rs g.asc plot

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 > a.rs
```

- standard in: none
- standard out: a.rs

```
bash$ ls
```

```
a.rs g.asc plot
```

```
bash$ sfdifil < a.rs
```

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 > a.rs
```

- standard in: none
- standard out: a.rs

```
bash$ ls
```

a.rs g.asc plot

```
bash$ sfdisk < a.rs
```

0: 0 1 0 0 0

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 > a.rsfsf
```

- standard in: none
- standard out: a.rsfsf

```
bash$ ls
```

a.rsfsf g.asc plot

```
bash$ sfdisfil < a.rsfsf
```

```
0:          0          1          0          0          0
```

- standard in: a.rsfsf
- standard out: screen

Command-line usage - Example

```
bash$ sfmath
```

Command-line usage - Example

```
bash$ sfmath
```

NAME

sfmath

DESCRIPTION

Mathematical operations on data files.

SYNOPSIS

```
sfmath > out.rsf nostdin=n n#= d#=(1,1,...) o#=(0,0,...) label#= unit#=
```

COMMENTS

Known functions:

cos, sin, tan, acos, asin, atan,
cosh, sinh, tanh, acosh, asinh, atanh,
exp, log, sqrt, abs,
erf, erfc, sign (for float data),
arg, conj, real, imag (for complex data).

sfmath will work on float or complex data, but all the input and output files must be of the same data type.

An alternative to sfmath is sfadd, which may be more efficient, but is less versatile.

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 | sfmath output="1-input" | sfdifil
```

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 | sfmath output="1-input" | sfdifil
```

0: 1 0 1 1 1

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 | sfmath output="1-input" | sfdifil
```

0:	1	0	1	1	1
----	---	---	---	---	---

```
bash$ sfspike n1=5 k1=4 > b.rsf
```

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 | sfmath output="1-input" | sfdifil
```

0:	1	0	1	1	1
----	---	---	---	---	---

```
bash$ sfspike n1=5 k1=4 > b.rs
```

```
bash$ < sfadd
```

Command-line usage - Example

```
bash$ sfspike n1=5 k1=2 | sfmath output="1-input" | sfdifil
```

```
0:          1          0          1
1          1
```

```
bash$ sfspike n1=5 k1=4 > b.rsfsf
```

```
bash$ < sfadd
```

NAME

sfadd

DESCRIPTION

Add, multiply, or divide RSF datasets.

SYNOPSIS

```
sfadd > out.rsfsf scale= add= sqrt= abs= log= exp= mode= [< file0.rsfsf  
file1.rsfsf file2.rsfsf ...
```

COMMENTS

The various operations, if selected, occur in the following order:

- (1) Take absolute value, abs=
- (2) Add a scalar, add=

Command-line usage - Example

```
bash$ < a.rsfsfadd scale=1,-2 b.rsfsf > c.rsfsf
```

Command-line usage - Example

```
bash$ < a.rsfsfadd scale=1,-2 b.rsfsf > c.rsfsf  
bash$ sfdissfil < a.rsfsf
```

Command-line usage - Example

```
bash$ < a.rsfsfadd scale=1,-2 b.rsfsf > c.rsfsf  
bash$ sfdissfil < a.rsfsf
```

0: 0 1 0 0 0

Command-line usage - Example

```
bash$ < a.rsfsfadd scale=1,-2 b.rsfsf > c.rsfsf  
bash$ sfdisfil < a.rsfsf
```

0: 0 1 0 0 0

```
bash$ sfdisfil < b.rsfsf
```

Command-line usage - Example

```
bash$ < a.rsfsfadd scale=1,-2 b.rsfsf > c.rsfsf
```

```
bash$ sfdisfil < a.rsfsf
```

0:	0	1	0	0	0
----	---	---	---	---	---

```
bash$ sfdisfil < b.rsfsf
```

0:	0	0	0	1	0
----	---	---	---	---	---

Command-line usage - Example

```
bash$ < a.rsfsfadd scale=1,-2 b.rsfsf > c.rsfsf
```

```
bash$ sfdisfil < a.rsfsf
```

0:	0	1	0	0	0
----	---	---	---	---	---

```
bash$ sfdisfil < b.rsfsf
```

0:	0	0	0	1	0
----	---	---	---	---	---

```
bash$ sfdisfil < c.rsfsf
```

Command-line usage - Example

```
bash$ < a.rsfsfadd scale=1,-2 b.rsfsf > c.rsfsf
```

```
bash$ sfdisfil < a.rsfsf
```

0:	0	1	0	0	0
----	---	---	---	---	---

```
bash$ sfdisfil < b.rsfsf
```

0:	0	0	0	1	0
----	---	---	---	---	---

```
bash$ sfdisfil < c.rsfsf
```

0:	0	1	0	-2	0
----	---	---	---	----	---

Command-line usage - auxiliary/extra input/output

Command-line usage - auxiliary/extra input/output

- 1 file enumeration: `sfadd`

Command-line usage - auxiliary/extra input/output

- 1 file enumeration: `sfadd`

```
bash$ sfadd
```

Command-line usage - auxiliary/extra input/output

❶ file enumeration: sfadd

```
bash$ sfadd
```

NAME

sfadd

DESCRIPTION

Add, multiply, or divide RSF datasets.

SYNOPSIS

```
sfadd > out.rsf scale= add= sqrt= abs= log= exp= mode= [< file  
0.rsf] file1.rsf file2.rsf ...
```

COMMENTS

The various operations, if selected, occur in the following order:

- (1) Take absolute value, abs=
- (2) Add a scalar, add=

Command-line usage - auxiliary/extra input/output

- 1 file enumeration: `sfadd`
- 2 via parameters: `sfawefd2d`

Command-line usage - auxiliary/extra input/output

- 1 file enumeration: `sfadd`
- 2 via parameters: `sfawefd2d`

```
bash$ sfawefd2d
```


Command-line usage - auxiliary/extra input/output

- 1 file enumeration: `sfadd`
- 2 via parameters: `sfawefd2d`

```
bash$ sfawefd2d
```

NAME

`sfawefd2d`

DESCRIPTION

2D acoustic time-domain FD modeling

SYNOPSIS

```
sfawefd2d < file_wav.rsfsf vel=file_vel.rsfsf sou=file_src.rsfsf rec
=file_rec.rsfsf > file_dat.rsfsf wfl=file_wfl.rsfsf den=file_den.rsfsf verb=n
snap=n expl=n dabc=n cden=n adj=n fsrf=n optfd=n fdorder=4 hybridbc=n
sinc=n jsnap=nt jdata=1 nqz=sf_n(az) nqx=sf_n(ax) oqz=sf_o(az) oqx=sf_
o(ax) dqz=sf_d(az) dqx=sf_d(ax)
```

.
.
.

Command-line usage - auxiliary/extra input/output

- 1 file enumeration: `sfadd`
- 2 via parameters: `sfawefd2d`

```
bash$ sfawefd2d
```

PARAMETERS

bool	adj=n [y/n]	adjoint flag
bool	cden=n [y/n]	Constant density
bool	dabc=n [y/n]	Absorbing BC
string	den=	auxiliary input file name
float	dqx=sf_d(ax)	Saved wfld window dx
float	dqz=sf_d(az)	Saved wfld window dz
bool	expl=n [y/n]	Multiple sources, one wvlt
int	fdorder=4	spatial FD order
bool	fsrf=n [y/n]	Free surface flag
bool	hybridbc=n [y/n]	hybrid Absorbing BC
int	jdata=1	# of t steps at which to save receiver data
int	jsnap=nt	# of t steps at which to save wavefield

Command-line usage - auxiliary/extra input/output

- 1 file enumeration: `sfadd`
- 2 via parameters: `sfawefd2d`

```
bash$ sfawefd2d
```

int	nqx=sf_n(ax)	Saved wfld window nx
int	nz=sf_n(az)	Saved wfld window nz
bool	optfd=n [y/n]	optimized FD coefficients flag
float	ox=sf_o(ax)	Saved wfld window ox
float	oz=sf_o(az)	Saved wfld window oz
file	rec=	auxiliary input file name
bool	sinc=n [y/n]	sinc source injection
bool	snap=n [y/n]	Wavefield snapshots flag
file	sou=	auxiliary input file name
file	vel=	auxiliary input file name
bool	verb=n [y/n]	Verbosity flag
file	wfl=	auxiliary output file name

Command-line usage - auxiliary/extra input/output

- ❶ file enumeration: `sfadd`
- ❷ via parameters: `sfawefd2d`
- ❸ variable definition: `sfmath`

Command-line usage - auxiliary/extra input/output

- 1 file enumeration: `sfadd`
- 2 via parameters: `sfawefd2d`
- 3 variable definition: `sfmath`

```
bash$ sfmath
```

Command-line usage - auxiliary/extra input/output

- 1 file enumeration: `sfadd`
- 2 via parameters: `sfawefd2d`
- 3 variable definition: `sfmath`

```
bash$ sfmath
```

NAME

`sfmath`

DESCRIPTION

Mathematical operations on data files.

SYNOPSIS

```
sfmath > out.rsf nostdin=n n#= d#=(1,1,...) o#=(0,0,...) label  
#= unit#= type= label= unit= output=
```

COMMENTS

Known functions:

`cos`, `sin`, `tan`, `acos`, `asin`, `atan`,

.

.

Command-line usage - auxiliary/extra input/output

- 1 file enumeration: `sfadd`
- 2 via parameters: `sfawefd2d`
- 3 variable definition: `sfmath`

```
bash$ sfmath
```

Examples:

```
sfmath x=file1.rs f y=file2.rs f power=file3.rs f  
      output='sin((x+2*y)^power)' > out.rs f  
sfmath < file1.rs f tau=file2.rs f output='exp(tau*input)' > out.rs f  
sfmath n1=100 type=complex output="exp(I*x1)" > out.rs f
```

.
. .
.

Regularly Sampled Format

To design a perfect anti-Unix, make all file formats binary and opaque, and require heavyweight tools to read and edit them.

If you feel an urge to design a complex binary file format, or a complex binary application protocol, it is generally wise to lie down until the feeling passes.

Regularly Sampled Format

- 1 Discrete representation of n -d functions
- 2 Uniform sampling
- 3 RSF dataset is n -d matrices with physical dimensions
- 4 Data type `int`, `float`, `double`, `complex`

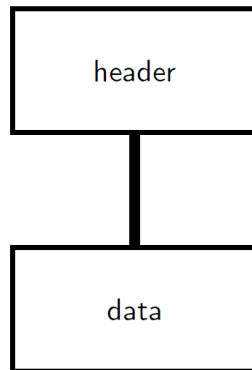
RSF “file” componets

Header file

- Text
- Small
- Portable

Data file

- ASCII or binary (native or XDR)
- Large (Huge)
- Path under \$DATAPATH



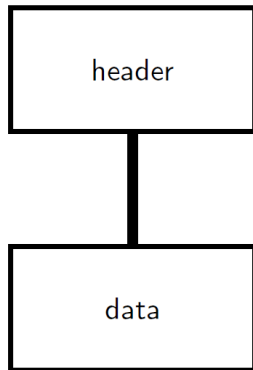
RSF “file” componets

Header file

- Text
- Small
- Portable

Data file

- ASCII or binary (native or XDR)
- Large (Huge)
- Path under \$DATAPATH



```
bash$ echo $DATAPATH
```

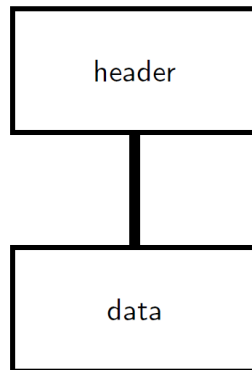
RSF “file” componets

Header file

- Text
- Small
- Portable

Data file

- ASCII or binary (native or XDR)
- Large (Huge)
- Path under \$DATAPATH



```
bash$ echo $DATAPATH
```

```
/home/daniel/rsfdata/
```

Header information

Example: construct Matrix

$$D = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix}$$

Header information

Example: construct Matrix

$$D = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix}$$

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' > d.rs
```

Header information

Example: construct Matrix

$$D = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix}$$

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' > d.rsf  
bash$ < d.rsf sfdifil
```

Header information

Example: construct Matrix

$$D = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix}$$

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' > d.rsf
bash$ < d.rsf sfdifil
```

0:	1	2	3	2	4
5:	6				

Header information

Example: construct Matrix

$$D = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix}$$

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' > d.rsf
bash$ < d.rsf sfdiskfil
```

0:	1	2	3	2	4
5:	6				

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' > d.rsf
```

Header information

Example: construct Matrix

$$D = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix}$$

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' > d.rsf
bash$ < d.rsf sfdifil
```

0:	1	2	3	2	4
5:	6				

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' > d.rsf
bash$ < d.rsf sfdifil col=3
```

Header information

Example: construct Matrix

$$D = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix}$$

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' > d.rsf
bash$ < d.rsf sfdifil
```

0:	1	2	3	2	4
3:	6				

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' > d.rsf
bash$ < d.rsf sfdifil col=3
```

0:	1	2	3
3:	2	4	6

Header information

Print out header

```
sfin file0.rsfs [file1.rsfs] [file2.rsfs] ...
```

```
bash$ sfin d.rsfs
```

```
d.rsfs:
  in="/home/daniel/rsfsdata/d.rsfs@"
  esize=4 type=float form=native
  n1=3          d1=1          o1=1
  n2=2          d2=1          o2=1
  6 elements 24 bytes
```

- n: number of samples
- o: origin of samples
- d: sampling interval
- label: axis label
- unit: axis unit

Header information

Print out header

```
sfin file0.rsfs [file1.rsfs] [file2.rsfs] ...
```

```
bash$ cd $DATAPATH
```

```
bash$ ls -l */**/*
```

```
.  
.  
.
```

RSF dataset attributes

Print out attributes

```
sfattr < file.rsf
```

RSF dataset attributes

Print out attributes

```
sfattr < file.rsf
```

```
bash$ sfattr < d.rsf
```

RSF dataset attributes

Print out attributes

```
sfattr < file.rsrf
```

```
bash$ sfattr < d.rsrf
```

```
*****
      rms =          3.41565
     mean =           3
  2-norm =          8.3666
variance =           3.2
std dev =          1.78885
      max =           6 at 3 2
      min =           1 at 1 1
nonzero samples = 6
  total samples = 6
*****
```


Modify header

(re)write header

```
sfput < in.rsf key1=val1 [...] > out.rsf
```

Modify header

(re)write header

```
sfput < in.rsfs key1=val1 [...] > out.rsfs
```

```
bash$ sfin d.rsfs
```

Modify header

(re)write header

```
sfput < in.rsfs key1=val1 [...] > out.rsfs
```

```
bash$ sfin d.rsfs
```

d.rsfs:

```
in="/home/daniel/rsfsdata/d.rsfs@"  
esize=4 type=float form=native  
n1=3          d1=1          o1=1  
n2=2          d2=1          o2=1  
6 elements 24 bytes
```

Modify header

(re)write header

```
sfput < in.rsfs key1=val1 [...] > out.rsfs
```

```
bash$ < d.rsfs sfput n1=6 n2=1 label1="time" > d2.rsfs
```

Modify header

(re)write header

```
sfput < in.rsfs key1=val1 [...] > out.rsfs
```

```
bash$ < d.rsfs sfput n1=6 n2=1 label1="time" > d2.rsfs  
bash$ sfin d2.rsfs
```

Modify header

(re)write header

```
sfput < in.rsfs key1=val1 [...] > out.rsfs
```

```
bash$ < d.rsfs sfput n1=6 n2=1 label1="time" > d2.rsfs  
bash$ sfin d2.rsfs
```

d2.rsfs:

```
in="/home/daniel/rsfsdata/d2.rsfs@"
```

```
esize=4 type=float form=native
```

```
n1=6          d1=1          o1=1          label1="time"
```

```
n2=1          d2=1          o2=1
```

```
6 elements 24 bytes
```

Memory allocation

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 n3=2 o3=1 output='x1*100 +  
x2*10 + x3' > e.rs
```

Memory allocation

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 n3=2 o3=1 output='x1*100 +  
x2*10 + x3' > e.rs
```

```
bash$ sfdifil < e.rs
```


Memory allocation

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 n3=2 o3=1 output='x1*100 +  
x2*10 + x3' > e.rsfsf
```

```
bash$ sfdisfil < e.rsfsf
```

0:	111	211	311	121	221
5:	321	112	212	312	122
10:	222	322			

Memory allocation

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 n3=2 o3=1 output='x1*100 +  
x2*10 + x3' > e.rsfl
```

```
bash$ sfdisfil < e.rsfl
```

0:	111	211	311	121	221
5:	321	112	212	312	122
10:	222	322			

```
bash$ sfdisfil < e.rsfl col=3
```

Memory allocation

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 n3=2 o3=1 output='x1*100 +  
x2*10 + x3' > e.rsf
```

```
bash$ sfdisfil < e.rsf
```

0:	111	211	311	121	221
5:	321	112	212	312	122
10:	222	322			

```
bash$ sfdisfil < e.rsf col=3
```

0:	111	211	311
3:	121	221	321
6:	112	212	312
9:	122	222	322

Moving RSF dataset

`mv` moves header ONLY

Moving RSF dataset

`mv` moves header ONLY

```
bash$ mv d.rsfs f.rsfs
```

Moving RSF dataset

`mv` moves header ONLY

```
bash$ mv d.rsfs f.rsfs
bash$ sfins f.rsfs
```

Moving RSF dataset

`mv` moves header ONLY

```
bash$ mv d.rsfs f.rsfs
bash$ sfinfo f.rsfs
```

`f.rsfs`:

```
in="/home/daniel/rsfsdata/d.rsfs@"
esize=4 type=float form=native
n1=3          d1=1          o1=1
n2=2          d2=1          o2=1
    6 elements 24 bytes
```

Moving RSF dataset

mv moves header ONLY

```
bash$ mv d.rsfs f.rsfs
bash$ sfinfo f.rsfs
```

```
f.rsfs:
  in="/home/daniel/rsfsdata/d.rsfs@"
  esize=4 type=float form=native
  n1=3          d1=1          o1=1
  n2=2          d2=1          o2=1
    6 elements 24 bytes
```

```
bash$ ls d.rsfs
```


Moving RSF dataset

`mv` moves header ONLY

```
bash$ mv d.rsfs f.rsfs
bash$ sfinfo f.rsfs
```

```
f.rsfs:
  in="/home/daniel/rsfsdata/d.rsfs@"
  esize=4 type=float form=native
  n1=3          d1=1          o1=1
  n2=2          d2=1          o2=1
  6 elements 24 bytes
```

```
bash$ ls d.rsfs
```

```
ls: cannot access 'd.rsfs': No such file or directory
```

Moving RSF dataset

Move header and data

```
sfmv in.rsf out.rsf
```

Moving RSF dataset

Move header and data

```
sfmv in.rsf out.rsf
```

```
bash$ mv f.rsf d.rsf
```

Moving RSF dataset

Move header and data

```
sfmv in.rsf out.rsf
```

```
bash$ mv f.rsf d.rsf
```

```
bash$ sfmv d.rsf f.rsf
```

Moving RSF dataset

Move header and data

```
sfmv in.rsf out.rsf
```

```
bash$ mv f.rsf d.rsf
```

```
bash$ sfmv d.rsf f.rsf
```

```
bash$ sfin f.rsf
```

Moving RSF dataset

Move header and data

```
sfmv in.rsf out.rsf
```

```
bash$ mv f.rsf d.rsf
```

```
bash$ sfmv d.rsf f.rsf
```

```
bash$ sfin f.rsf
```

f.rsf:

```
in="/home/daniel/rsfdata/f.rsf@"
```

```
esize=4 type=float form=native
```

```
n1=3          d1=1          o1=1
```

```
n2=2          d2=1          o2=1
```

```
6 elements 24 bytes
```

Copying and deleting RSF

Copy header and data

```
sfcop in.rsf out.rsf
```

Copying and deleting RSF

Copy header and data

```
sfcp in.rsf out.rsf
```

```
bash$ sfcp a.rsf b.rsf
```


Copying and deleting RSF

Copy header and data

```
sfcop in.rsf out.rsf
```

```
bash$ sfcop a.rsf b.rsf
```

```
bash$ sfin b.rsf
```

Copying and deleting RSF

Copy header and data

```
sfcp in.rsf out.rsf
```

```
bash$ sfcp a.rsf b.rsf
```

```
bash$ sfin b.rsf
```

b.rsf:

```
in="/home/daniel/rsfdata/b.rsf@"  
esize=4 type=float form=native  
n1=5          d1=0.004      o1=0          label1="Time" unit1="s"  
5 elements 20 bytes
```

Copying and deleting RSF

Copy header and data

```
sfcp in.rsfsf out.rsfsf
```

```
bash$ sfcp a.rsfsf b.rsfsf
```

```
bash$ sfin b.rsfsf
```

b.rsfsf:

```
in="/home/daniel/rsfdata/b.rsfsf@"  
esize=4 type=float form=native  
n1=5          d1=0.004          o1=0          label1="Time" unit1="s"  
5 elements 20 bytes
```

```
bash$ sfdiskfil < b.rsfsf
```

Copying and deleting RSF

Copy header and data

```
sfcop in.rsfc out.rsfc
```

```
bash$ sfcop a.rsfc b.rsfc
```

```
bash$ sfin b.rsfc
```

b.rsfc:

```
in="/home/daniel/rsfdata/b.rsfc@"
```

```
esize=4 type=float form=native
```

```
n1=5          d1=0.004      o1=0          label1="Time" unit1="s"  
5 elements 20 bytes
```

```
bash$ sfdiscfil < b.rsfc
```

```
0:          0          1          0          0          0
```

Copying and deleting RSF

Delete header and data

```
sfrm file1.rsfs file2.rsfs [...]
```

Copying and deleting RSF

Delete header and data

```
sfrm file1.rsf file2.rsf [...]
```

```
bash$ rm a.rsf
```

Copying and deleting RSF

Delete header and data

```
sfrm file1.rsfs file2.rsfs [...]
```

```
bash$ rm a.rsfs
```

```
bash$ ls /home/daniel/rsfsdata/a.rsfs@
```

Copying and deleting RSF

Delete header and data

```
sfrm file1.rsfs file2.rsfs [...]
```

```
bash$ rm a.rsfs
```

```
bash$ ls /home/daniel/rsfsdata/a.rsfs@
```

```
/home/daniel/rsfsdata/a.rsfs@
```


Copying and deleting RSF

Delete header and data

```
sfrm file1.rsfs file2.rsfs [...]
```

```
bash$ rm a.rsfs
```

```
bash$ ls /home/daniel/rsfsdata/a.rsfs@
```

```
/home/daniel/rsfsdata/a.rsfs@
```

```
bash$ sfrm b.rsfs
```

Copying and deleting RSF

Delete header and data

```
sfrm file1.rsfs file2.rsfs [...]
```

```
bash$ rm a.rsfs
```

```
bash$ ls /home/daniel/rsfsdata/a.rsfs@
```

```
/home/daniel/rsfsdata/a.rsfs@
```

```
bash$ sfrm b.rsfs
```

```
bash$ ls /home/daniel/rsfsdata/b.rsfs@
```

Copying and deleting RSF

Delete header and data

```
sfrm file1.rsf file2.rsf [...]
```

```
bash$ rm a.rsf
```

```
bash$ ls /home/daniel/rsfdata/a.rsf@
```

```
/home/daniel/rsfdata/a.rsf@
```

```
bash$ sfrm b.rsf
```

```
bash$ ls /home/daniel/rsfdata/b.rsf@
```

```
ls: cannot access '/home/daniel/rsfdata/b.rsf@' : No such file or directory
```

Copying and deleting RSF

Delete header and data

```
sfrm file1.rsfs file2.rsfs [...]
```

```
bash$ rm a.rsfs
```

```
bash$ ls /home/daniel/rsfsdata/a.rsfs@
```

```
/home/daniel/rsfsdata/a.rsfs@
```

```
bash$ sfrm b.rsfs
```

```
bash$ ls /home/daniel/rsfsdata/b.rsfs@
```

```
ls: cannot access '/home/daniel/rsfsdata/b.rsfs@' : No such file or directory
```

```
bash$ sfrm a.rsfs
```

Copying and deleting RSF

Delete header and data

```
sfrm file1.rsfs file2.rsfs [...]
```

```
bash$ rm a.rsfs
```

```
bash$ ls /home/daniel/rsfsdata/a.rsfs@
```

```
/home/daniel/rsfsdata/a.rsfs@
```

```
bash$ sfrm b.rsfs
```

```
bash$ ls /home/daniel/rsfsdata/b.rsfs@
```

```
ls: /home/daniel/rsfsdata/b.rsfs@ : No such file or directory
```

```
bash$ sfrm a.rsfs
```

```
sfrm: build/api/c/files.c: Cannot open file a.rsfs: No such file or directory
```

RSF dataset in a single file

Packing header and data

```
[< in.rsfsfprog [> out.rsfsf] out=stdout
```

RSF dataset in a single file

Packing header and data

```
[< in.rsfsfprog [> out.rsfsf] out=stdout
```

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' out=stdout >  
a.rsfsf
```

RSF dataset in a single file

Packing header and data

```
[< in.rsfsfprog [> out.rsfsf] out=stdout
```

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' out=stdout >  
a.rsfsf
```

```
bash$ sfin a.rsfsf
```


RSF dataset in a single file

Packing header and data

```
[< in.rsfsfprog [> out.rsfsf] out=stdout
```

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' out=stdout >  
a.rsfsf
```

```
bash$ sfin a.rsfsf
```

```
a.rsfsf:  
  in="stdin"  
  esize=4 type=float form=native  
  n1=3          d1=1          o1=1  
  n2=2          d2=1          o2=1  
    6 elements 24 bytes
```

RSF dataset in a single file

Packing header and data

```
[< in.rsff] sfprog [> out.rsff] out=stdout
```

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' out=stdout >  
a.rsff
```

```
bash$ sfin a.rsff
```

```
a.rsff:  
  in="stdin"  
  esize=4 type=float form=native  
  n1=3          d1=1          o1=1  
  n2=2          d2=1          o2=1  
    6 elements 24 bytes
```

in="stdin" indicates standalone RSF dataset

RSF dataset in a single file

Packing header and data

```
[< in.rsfsfprog [> out.rsfsf] out=stdout
```

```
bash$ sfmath n1=3 o1=1 n2=2 o2=1 output='x1*x2' out=stdout >  
a.rsfsf
```

```
bash$ sfin a.rsfsf
```

a.rsfsf:

```
in="stdin"  
esize=4 type=float form=native  
n1=3          d1=1          o1=1  
n2=2          d2=1          o2=1  
6 elements 24 bytes
```

in="stdin" indicates standalone RSF dataset

Exchange dataset between systems

```
< in.rsfsfdd form=xdr out=stdout > out.rsfsf
```

Exercício 1

Construa um vetor com 20 amostras que contenha um período completo de $\sin t$. Visualize na tela seu resultado. Em seguida concatene dois destes períodos para formar um vetor de 40 amostras que contenha dois períodos. Visualize na tela seu resultado.

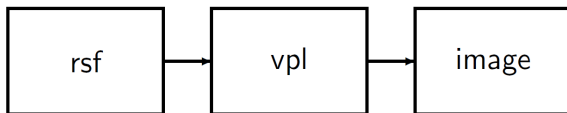
Exercício 2

Construa uma matriz que contenha os valores referentes a um parabolóide circular centrado na origem. A origem deve estar no centro da matriz. Primeiramente faça a matriz quadrada e visualize o resultado na tela. Em seguida faça uma matriz retangular e visualize na tela.

VPLOT

- “.vpl” suffix
- Vector image can be scaled without affecting quality
- Displayed by pen programs
- Compact

VPLOT



Madagascar plotting programs: `sfprog < in.rsf par= > out.vpl`

- `sfgraph`
- `sfgrey`
- `sfgrey3`
- `sfcontour`
- `sfdots`
- ...

pen progrms convert `.vpl` to images (`.eps`, `.gif`, `.png`, ...)

- `vppen`
- `xtpen`
- `popen`
- ...

sfgraph

```
bash$ cd plot
```


sfgraph

```
bash$ cd plot  
bash$ mkdir Fig/
```

sfgraph

```
bash$ cd plot  
bash$ mkdir Fig/  
bash$ ls
```

sfgraph

```
bash$ cd plot
```

```
bash$ mkdir Fig/
```

```
bash$ ls
```

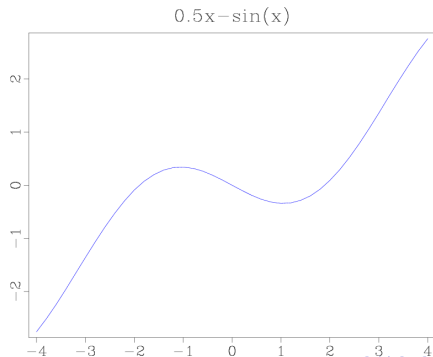
```
data.rsfc  Fig  plot.sh  SConstruct  shots.rsfc
```

sfgraph

```
bash$ sfmath n1=41 o1=-4 d1=.2 output=".5*x1" > y1.rsf
bash$ < y1.rsf sfmath output="sin(x1)" > y2.rsf
bash$ < y1.rsf sfmath sin=y2.rsf output="input-sin" > y3.rsf
bash$ < y3.rsf sfgraph title="0.5x-sin(x)" > Fig/fig1.vpl
bash$ sfpen < Fig/fig1.vpl
```

sfgraph

```
bash$ sfmath n1=41 o1=-4 d1=.2 output=".5*x1" > y1.rsf
bash$ < y1.rsf sfmath output="sin(x1)" > y2.rsf
bash$ < y1.rsf sfmath sin=y2.rsf output="input-sin" > y3.rsf
bash$ < y3.rsf sfgraph title="0.5x-sin(x)" > Fig/fig1.vpl
bash$ sfpen < Fig/fig1.vpl
```

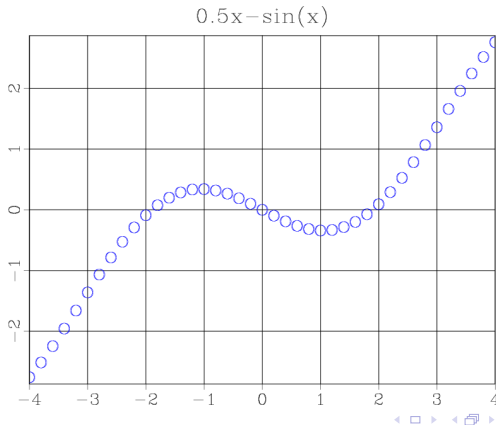


sfgraph

```
bash$ < y3.rsf sfgraph title="0.5x-sin(x)" symbol=o  
symbolsz=12 grid=y min1=-4 max1=4 > Fig/fig2.vpl  
bash$ sfpen < Fig/fig2.vpl
```

sfgraph

```
bash$ < y3.rs sfgraph title="0.5x-sin(x)" symbol=o  
symbolsz=12 grid=y min1=-4 max1=4 > Fig/fig2.vpl  
bash$ sfpen < Fig/fig2.vpl
```

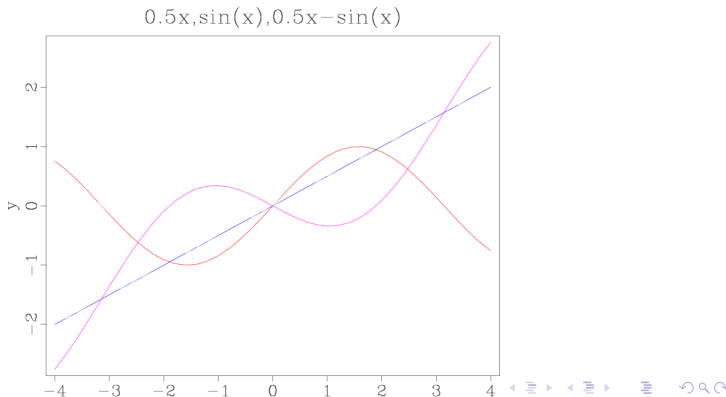


sfgraph

```
bash$ < y1.rsf sfcats y2.rsf y3.rsf axis=2 > y4.rsf
bash$ < y4.rsf sfgraph title="0.5x,sin(x),0.5x-sin(x)"
label1=x label2=y > Fig/fig3.vpl
bash$ sfopen < Fig/fig3.vpl
```


sfgraph

```
bash$ < y1.rsf sfcats y2.rsf y3.rsf axis=2 > y4.rsf
bash$ < y4.rsf sfgraph title="0.5x,sin(x),0.5x-sin(x)"
label1=x label2=y > Fig/fig3.vpl
bash$ sfopen < Fig/fig3.vpl
```



sfgrey

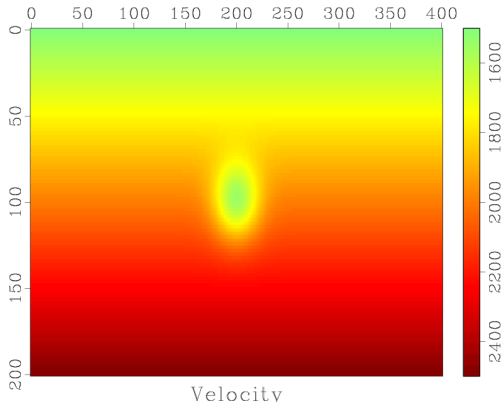
```
bash$ sfmath n1=101 d1=2 n2=201 d2=2 output="1500+5*x1" >
vb.rsf
bash$ < vb.rsf sfmath output="-exp(-.002*((x1-100)*(x1-100)+
(x2-200)*(x2-200)))*450" > v1.rsf
```

sfgrey

```
bash$ sfadd < vb.rsf v1.rsf scale=1,1 > v.rsf  
bash$ < v.rsf sfgrey title=Velocity color=j bias=1500  
scalebar=y barreverse=y > Fig/fig4.vpl  
bash$ sfopen < Fig/fig4.vpl
```

sfgrey

```
bash$ sfadd < vb.rsf v1.rsf scale=1,1 > v.rsf  
bash$ < v.rsf sfgrey title=Velocity color=j bias=1500  
scalebar=y barreverse=y > Fig/fig4.vpl  
bash$ sfopen < Fig/fig4.vpl
```

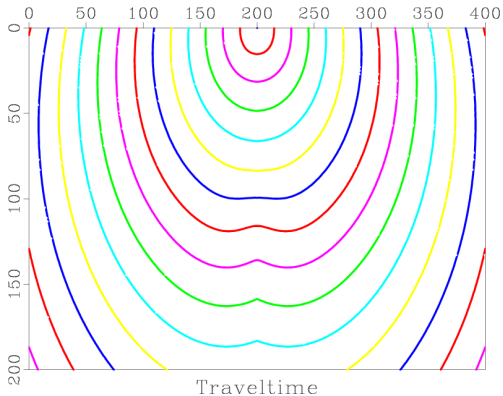


sfcontour

```
bash$ < v.rsfeikonal yshot=200 > eik.rsfe  
bash$ < eik.rsfe sfcontour nc=45 title=Traveltime plotfat=5 >  
Fig/fig5.vpl  
bash$ sfpen < Fig/fig5.vpl
```

sfcontour

```
bash$ < v.rsfsfeikonal yshot=200 > eik.rsfsf  
bash$ < eik.rsfsfcontour nc=45 title=Traveltime plotfat=5 >  
Fig/fig5.vpl  
bash$ sfpen < Fig/fig5.vpl
```

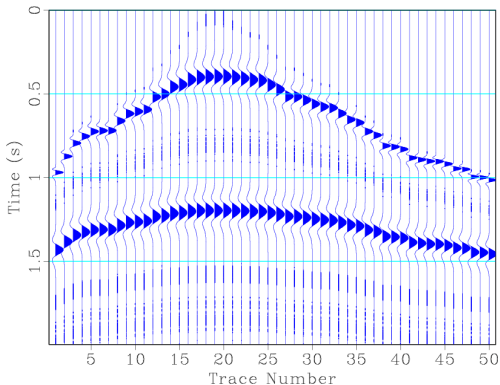


sfwiggle

```
bash$ < data.rsfc sfwiggle title= label2='Trace Number'  
yreverse=y transp=y poly=y > Fig/fig7.vpl  
bash$ sfpen < Fig/fig7.vpl
```

sfwiggle

```
bash$ < data.rsf sfwiggle title= label2='Trace Number'  
yreverse=y transp=y poly=y > Fig/fig7.vpl  
bash$ sfopen < Fig/fig7.vpl
```

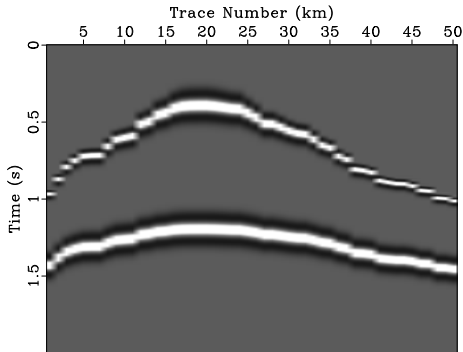


sfgwiggle × sfgrey

```
bash$ < data.rsfsfgrey title= label2='Trace Number' >  
Fig/fig8.vpl  
bash$ sfpen < Fig/fig8.vpl
```

sfwiggle × sfgrey

```
bash$ < data.rsf sfgrey title= label2='Trace Number' >  
Fig/fig8.vpl  
bash$ sfpen < Fig/fig8.vpl
```

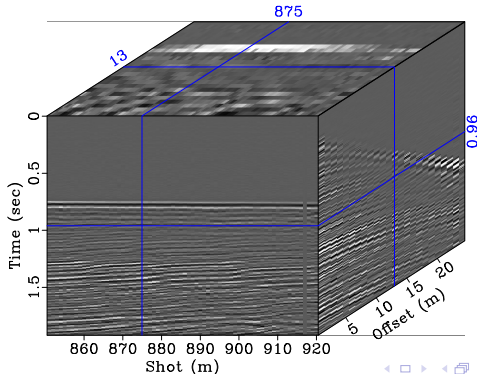


sfgrey3

```
bash$ < shots.rsf sfbyte | sfgrey3 frame1=240 frame2=24  
frame3=12 point1=0.7 point2=0.65 wanttitle=n flat=n  
title='Data' > Fig/fig9.vpl  
bash$ sfopen < Fig/fig9.vpl
```

sfgrey3

```
bash$ < shots.rsf sfbyte | sfgrey3 frame1=240 frame2=24  
frame3=12 point1=0.7 point2=0.65 wanttitle=n flat=n  
title='Data' > Fig/fig9.vpl  
bash$ sfpen < Fig/fig9.vpl
```



Exercício 3

Plote os resultados obtidos no exercício 2 usando `sfcontour` e `sfgrey`.

Exercício 4

Montar um modelo de velocidade de 2000 m inline por 1000 m de profundidade, com amostras espaçadas de 10 m tanto na vertical como na horizontal. A velocidade de fundo é constante igual a 1500 m/s e com uma perturbação retangular com intensidade, posição e tamanho que você quiser.

Exercício 5

Montar um modelo de velocidade com as mesmas dimensões anteriores mas com velocidade de fundo com um gradiente de 1.5 1/s e velocidade inicial de 1500 m/s . Além disso incluir uma perturbação circular com velocidade constante de 3000 m/s com raio e centro aonde você desejar.