MADAGASCAR
command line usage

Paul Sava

Center for Wave Phenomena
Colorado School of Mines
psava@mines.edu

MADAGASCAR **mission**

- **a research environment**

- **a technology transfer tool**

MADAGASCAR **mission**

- **a research environment**
  –open-source software package–

- **a technology transfer tool**

MADAGASCAR **mission**

- **a research environment**
  –open-source software package–

- **a technology transfer tool**
  –reproducible numeric experiments system–

why open source software?

freedom

to use

to study

to modify

to redistribute

to improve

reproducibility

why numeric reproducibility?

collaborative research

technology transfer

education

peer review

MADAGASCAR system

software
documents
blog http://www.ahay.org
merchandise
...

- standard GPL license
- distributed development (academia/industry)
- version control (`subversion`)
- installation and flows (`scons`)

MADAGASCAR **developers**

- University of Texas (Austin)
- Colorado School of Mines
- King Abdullah University of Science and Technology
- University of Western Australia
- ...

MADAGASCAR heritage

software system

- SEPlib (Stanford University)
- SU (Colorado School of Mines)
- DDS (Amoco/BP)

reproducible documents

- SEP document system

MADAGASCAR architecture

documents

flows 3 overlapping layers

programs

MADAGASCAR **architecture**

documents

flows

programs

- provide basic processing modules
- C, C++, F90, Python, Matlab ...
- communicate by pipes

MADAGASCAR architecture

documents

flows

programs

- provide processing history
- Python (SCons)
- combine processing modules

MADAGASCAR architecture

documents

flows

programs

- provide reproducible documents
- LaTeX and SCons
- assemble text and numeric results

programs

- independent processing modules
- combined using pipes
- “sf” prefix
- documented by examples (“books”)
- implement SMP parallelism
- program count: 1028 on 4/1/2014

program list

sfdoc -k .

```
sfofpwd: Objective function of dip estimation with PWD filters.
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.
sfradstretch: Stretch of the time axis.
sflpef: Find PEF on aliased traces.
sfrefer: Subtract a reference from a grid.
sflevint: Leveler inverse interpolation in 1-D.
sfnoise: Add random noise to the data.
...
```

self documentation

- run program without arguments
- find program purpose
- find execution parameters
- find execution examples

example

sfspike

```
NAME
        sfspike
DESCRIPTION
        Generate simple data: spikes, boxes, planes, constants.
SYNOPSIS
        sfspike > spike.rsf mag= nsp=1 k#=[0,...] l#=[k1,k2,...] p#=[0,...] n#=
o#=(0,...) d#=(0.004,0.1,0.1,...) label#=(Time,Distance,Distance,...) unit#=[s,km,km,...] title=
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]
        string  label#=(Time,Distance,Distance,...)     label on #-th axis
        floats  mag=    spike magnitudes  [nsp]
        int     n#=     dimension of #-th axis
        int     nsp=1   Number of spikes
        float   o#=(0,...)      origin on #-th axis
        floats  p#=[0,...]      spike inclination (in samples)  [nsp]
        string  title=  title for plots
        string  unit#=[s,km,km,...]     unit on #-th axis
USED IN
        bei/conj/causint
        bei/dpmv/matt
        bei/dpmv/yalei
        bei/dwnc/vofz
...
```

program execution

single input, single output

```
< i.rsf sfprog arguments > o.rsf
```

- sfprog = MADAGASCAR program
- arguments = program arguments
- input from `stdin` (<)
- output to `stdout` (>)

demo

sfspike `n1=100 o1=0 d1=0.01 n2=50 o2=1000 d2=10 > f1.rsf`

- standard in: none
- standart out: `f1.rsf`

file format

header:

- text file (description of data)
- description of regularly-sampled format
- small, can be archived

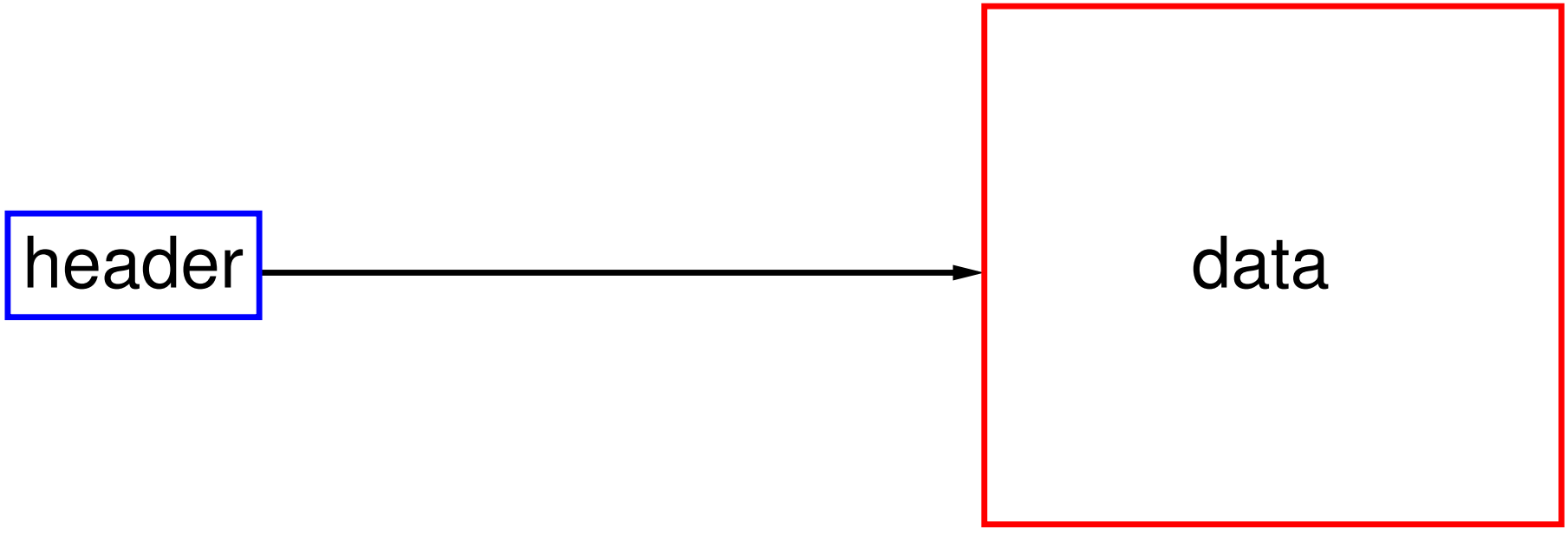
binary:

- binary file (actual data)
- regularly-sampled data (native binary or XDR binary)
- large, can be stored on a different file system
- path to binary set with environment variable `DATAPATH`

file format

home file system

scratch file system



example

sfin

```
NAME
        sfin
DESCRIPTION
        Display basic information about RSF files.
SYNOPSIS
        sfin info=true check=2. trail=true file1.rsf file2.rsf ...
COMMENTS
        n1,n2,... are data dimensions
        o1,o2,... are axis origins
        d1,d2,... are axis sampling intervals
        label1,label2,... are axis labels
        unit1,unit2,... are axis units

PARAMETERS
        float   check=2.        Portion of the data (in Mb) to check for zero values.
        bool    info=y [y/n]    If n, only display the name of the data file.
        bool    trail=y [y/n]   If n, skip trailing dimensions of  one
USED IN
        data/sigsbee/fs2B
        data/sigsbee/nfs2B
...
SOURCE
        filt/main/in.c
```

demo

sfin `f1.rsf`

```
f1.rsf:
    in="/scratch/f1.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.01         o1=0          label1="Time" unit1="s"
    n2=50           d2=10           o2=1000       label2="Distance" unit2="km"
        5000 elements 20000 bytes
```

dataset described by:

- header: `f1.rsf`
- binary: `in="/scratch/f1.rsf@"`

axis described by:

- *n*: number of samples
- *o*: sampling origin
- *d*: sampling rate (delta)
- *label*: axis label
- *unit*: axis unit

demo

```
< f1.rsf sfattr
```

```
****************************************
rms = 1
mean value = 1
norm value = 70.7107
variance = 0
standard deviation = 0
maximum value = 1 at 1 1
minimum value = 1 at 1 1
number of nonzero samples = 5000
total number of samples = 5000
****************************************
```

compatibility

- SEPlib: identical format

  In `f1.rsf`

- SU: use converters

  sfsegyread `tape=f1.su su=y tfile=tfile.rsf endian=0`

  `> f1.rsf`

  sfsegywrite `tape=f1.su su=y tfile=tfile.rsf endian=0`

  `< f1.rsf`

demo

```
< f1.rsf sfwindow n2=25 min2=1200 > f2.rsf
```

sfin f1.rsf

```
f1.rsf:
    in="/scratch/f1.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.01         o1=0          label1="Time" unit1="s"
    n2=50           d2=10           o2=1000       label2="Distance" unit2="km"
        5000 elements 20000 bytes
```

sfin f2.rsf

```
f2.rsf:
    in="/scratch/f2.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.01         o1=0          label1="Time" unit1="s"
    n2=25           d2=10           o2=1200       label2="Distance" unit2="km"
        2500 elements 10000 bytes
```

program execution

multiple inputs, multiple outputs

```
< i.rsf sfprog arguments l1=f1.rsf l2=f2.rsf > o.rsf
```

- input from `stdin` (<)
- output to `stdout` (>)
- `f1.rsf` can be open for input and/or output
- `f2.rsf` can be open for input and/or output

example

sfbandpass

```
NAME
        sfbandpass
DESCRIPTION
        Bandpass filtering.
SYNOPSIS
        sfbandpass < in.rsf > out.rsf flo= fhi= phase=n verb=n nplo=6 nphi=6
PARAMETERS
        float   fhi=    High frequency in band, default is Nyquist
        float   flo=    Low frequency in band, default is 0
        int     nphi=6  number of poles for high cutoff
        int     nplo=6  number of poles for low cutoff
        bool    phase=n [y/n]   y: minimum phase, n: zero phase
        bool    verb=n [y/n]    verbosity flag
SOURCE
        system/generic/Mbandpass.c
```

pipes

- MADAGASCAR programs can be piped
- `stdout` from one program becomes `stdin` for the next
- no intrinsic limit for the number of pipes

demo

```
< f1.rsf sfwindow n2=25 min2=1200 | sftransp > f3.rsf
```

sfin f1.rsf

```
f1.rsf:
    in="/scratch/f1.rsf@"
    esize=4 type=float form=native
    n1=100          d1=0.01         o1=0            label1="Time" unit1="s"
    n2=50           d2=10           o2=1000         label2="Distance" unit2="km"
        5000 elements 20000 bytes
```

sfin f3.rsf

```
f3.rsf:
    in="/scratch/f3.rsf@"
    esize=4 type=float form=native
    n1=25           d1=10           o1=1200         label1="Distance" unit1="km"
    n2=100          d2=0.01         o2=0            label2="Time" unit2="s"
        2500 elements 10000 bytes
```

useful utilities

example
sfmath

```
NAME
        sfmath
DESCRIPTION
        Mathematical operations on data files.
SYNOPSIS
        sfmath > out.rsf type= unit= output=
COMMENTS

        Known functions: cos,  sin,  tan,  acos,  asin,  atan,
                         cosh, sinh, tanh, acosh, asinh, atanh,
                         exp,  log,  sqrt, abs, conj (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.

        Examples:

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

        See also: sfheadermath.

PARAMETERS
        string  output=         Mathematical description of the output
        string  type=   output data type [float,complex]
        string  unit=
USED IN
        bei/dpmv/matt
        bei/dwnc/sigmoid
        bei/ft1/autocor
...
```

demo

sfmath n1=1000 output='sin(0.5*x1)' > s1.rsf

sfin s1.rsf

```
s1.rsf:
    in="/scratch/s1.rsf@"
    esize=4 type=float form=native
    n1=1000         d1=1            o1=0
        1000 elements 4000 bytes
```

demo

sfmath n1=300 n2=200 output='sin(0.25*x1+1*x2)' > s2.rsf

sfin s2.rsf

```
s2.rsf:
    in="/scratch/s2.rsf@"
    esize=4 type=float form=native
    n1=300          d1=1            o1=0
    n2=200          d2=1            o2=0
        60000 elements 240000 bytes
```

plotting

- **sfgraph**: 1D graphs
- **sfgrey**: 2D/3D grayscale graphs
- **contour**: contour plots
- **sfgrey3**: cube plots
- ...

demo

sfin s1.rsf

```
s1.rsf:
    in="/scratch/s1.rsf@"
    esize=4 type=float form=native
    n1=1000         d1=1            o1=0
        1000 elements 4000 bytes
```

< s1.rsf **sfgraph** title='' | **xtpen**

demo

sfin s2.rsf

```
s2.rsf:
    in="/scratch/s2.rsf@"
    esize=4 type=float form=native
    n1=300          d1=1            o1=0
    n2=200          d2=1            o2=0
        60000 elements 240000 bytes
```

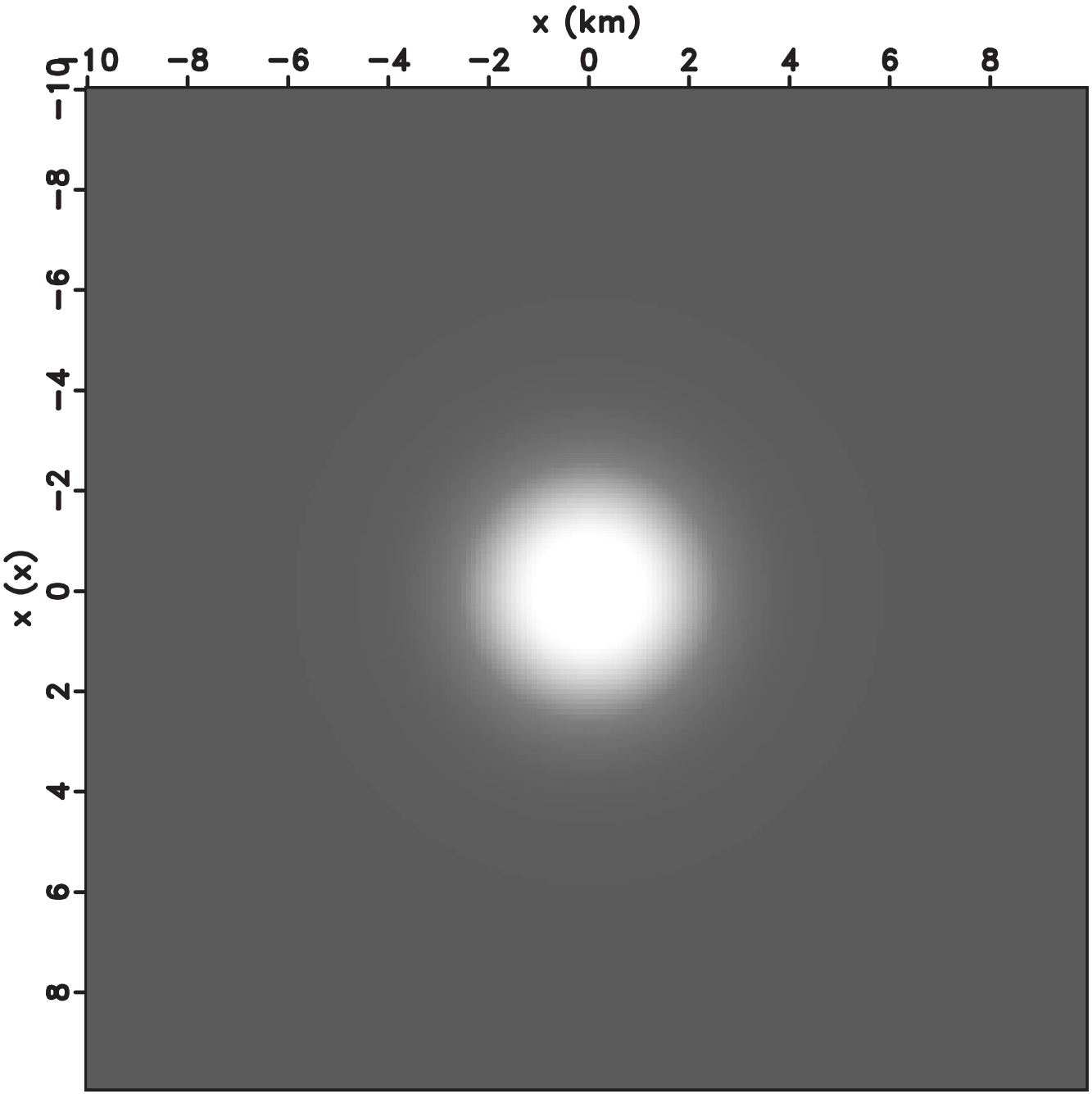
< s2.rsf **sfgrey** title='' | **xtpen**

exercise

create 2D Gaussian function

```
sfmath output="exp(-(x1*x1+x2*x2)/(2*1.5*1.5))"
n1=200 d1=0.1 o1=-10.  n2=200 d2=0.1 o2=-10.  |
sfput label1=z unit1=km label2=x unit2=km > gg.rsf
```

```
< gg.rsf sfgrey pclip=100 screenratio=1 title='' | xtpen
```

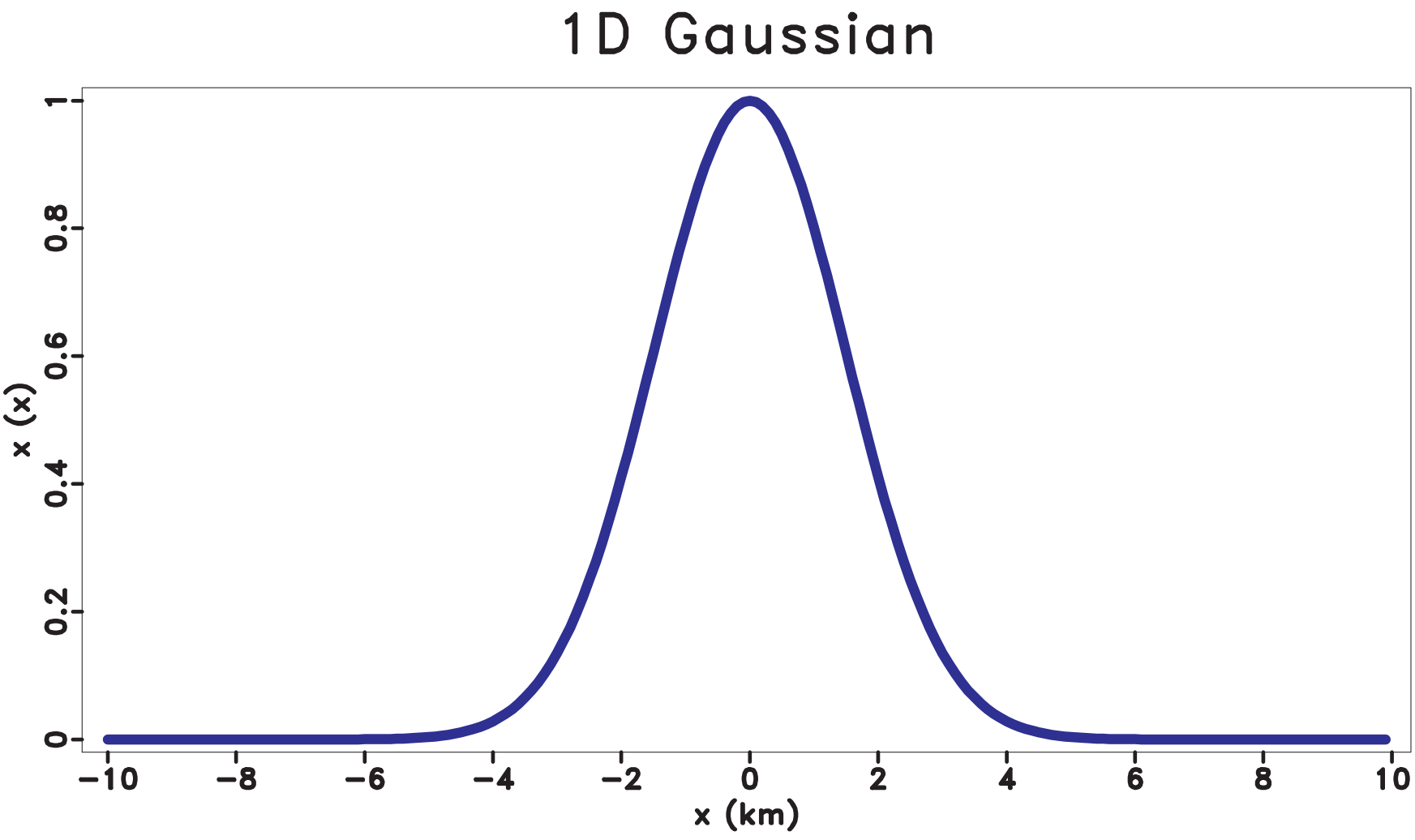


2D Gaussian

extract 1D subset

```
< gg.rsf sfwindow n2=1 f2=100 > gg0.rsf
```

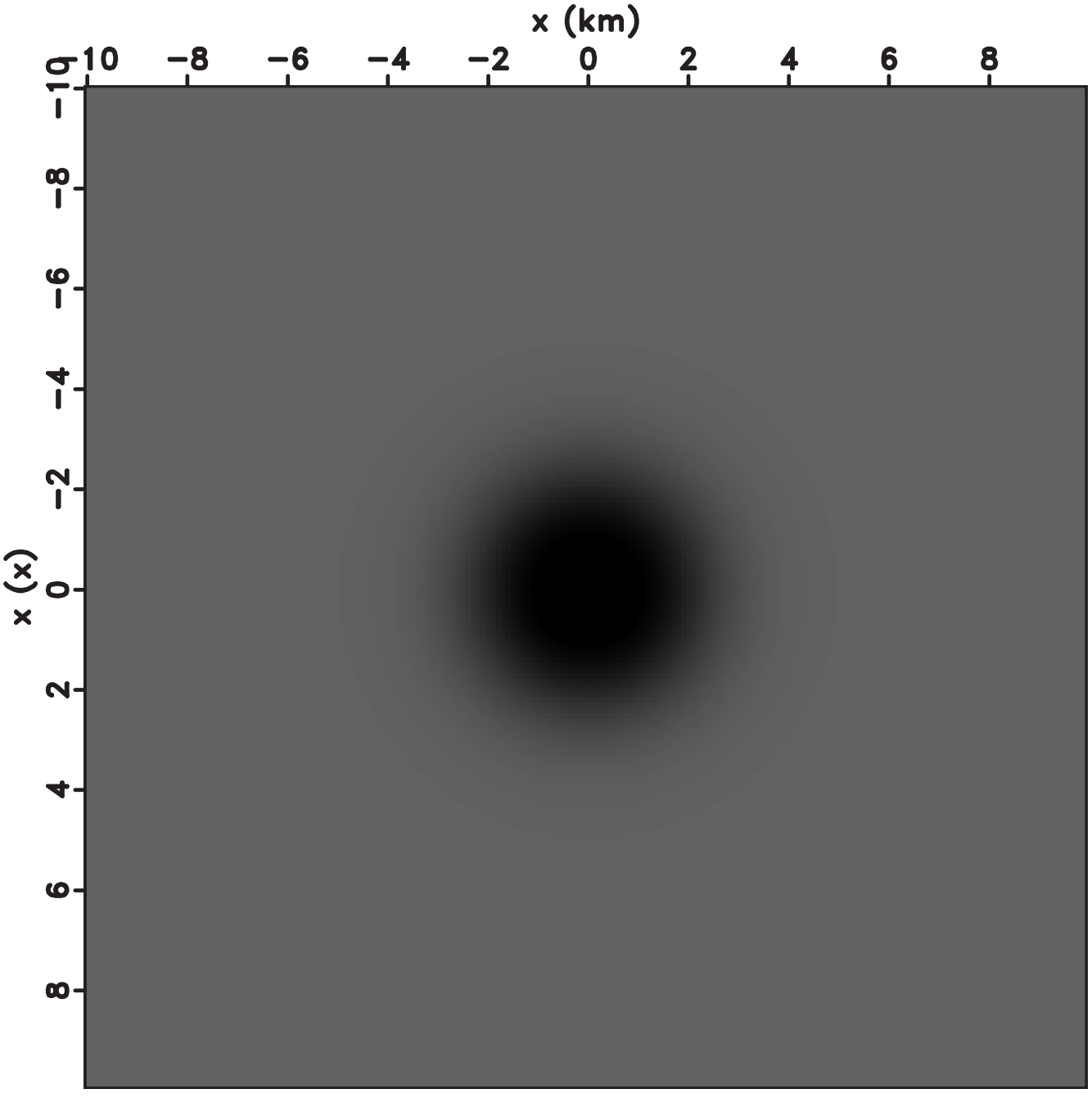
```
< gg0.rsf sfgraph title='' | xtpen
```


1D Gaussian
x (x)
0
0.2
0.4
0.6
0.8
1
−10
−8
−6
−4
−2
0
2
4
6
8
10
x (km)

create a velocity model

```
< gg.rsf sfmath output="3-input" > vel.rsf
```

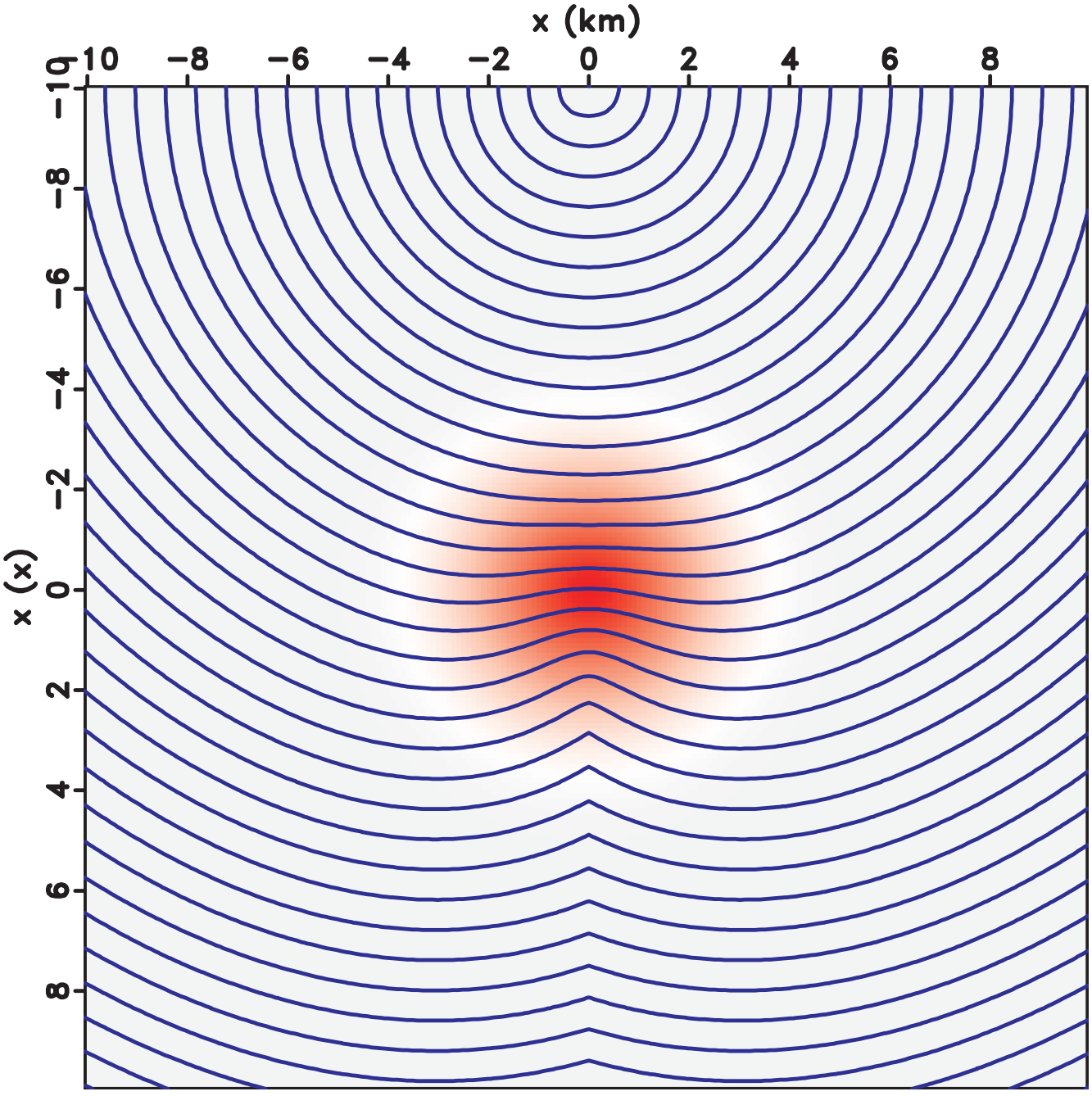
```
< vel.rsf sfgrey title='' pclip=100 screenratio=1 | xtpen
```


x (km)
−10
−8
−6
−4
−2
0
2
4
6
8
x (x)
−10
−8
−6
−4
−2
0
2
4
6
8

compute traveltimes

```
< vel.rsf sfeikonal zshot=-10 yshot=0 > fme.rsf
```

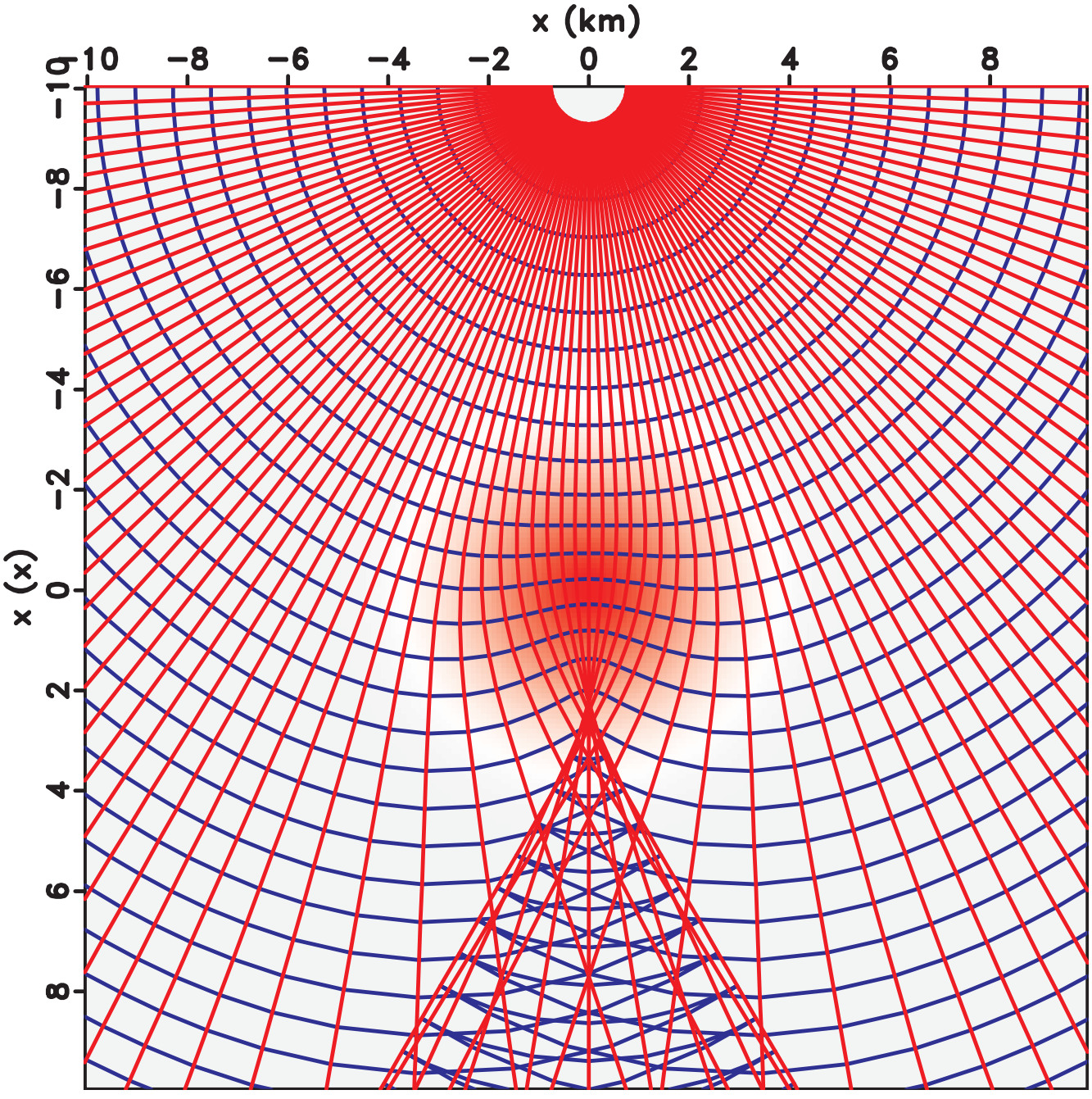
```
< fme.rsf sfcontour title='' nc=200 screenratio=1 | xtpen
```


x (km)
−10
−8
−6
−4
−2
0
2
4
6
8
x (x)
−10
−8
−6
−4
−2
0
2
4
6
8

compute rays and wavefronts

```
< vel.rsf sfhwt2d xsou=0 zsou=-10
nt=1000 ot=0 dt=0.01 ng=1801 og=-90 dg=0.1 > hwt.rsf
```

```
< hwt.rsf sfwindow j1=20 j2=20 |
sfgraph title='' yreverse=y screenratio=1
min1=-10 max1=+10 min2=-10 max2=+10 | xtpen
```


x (km)
−10
−8
−6
−4
−2
0
2
4
6
8
x (x)
−10
−8
−6
−4
−2
0
2
4
6
8

http://www.ahay.org

$RSF/book/rsf/school