

User's manual for SLIM programs in Madagascar

*Gilles Hennenfent*¹

ABSTRACT

This guide documents the contributions to Madagascar made by authors from the Seismic Laboratory for Imaging and Modeling (SLIM) at the University of British Columbia (UBC).

COPYRIGHT

Copyright (c) The University of British Columbia at Vancouver, 2005-2007.

LICENSE

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

DISCLAIMER

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

¹*Seismic Laboratory for Imaging and Modeling, Dept. of Earth and Ocean Sciences, the University of British Columbia, Vancouver, BC, Canada*

¹**e-mail:** ghenhenfent@eos.ubc.ca

UTILITIES

sfthr: Threshold float/complex inputs given a constant/varying threshold level.

```
sfthr < in.rsrf > out.rsrf fthr=fthr.rsrf thr= mode=
```

Methods available:

- soft
- hard
- non-negative Garrote (nng)

Written by: Gilles Hennenfent & Colin Russell, UBC

Created: February 2006

<u>string</u>	fthr=	varying threshold level (positive number) (auxiliary input file name)
<u>string</u>	mode=	'soft', 'hard', 'nng' (default: soft)
<u>float</u>	thr=	threshold level (positive number)

Consider the vector $\mathbf{x} := \{x_i\}_{0 \leq i < m} \in \mathbb{R}^m$. Soft thresholding is defined as

$$\mathcal{S}_\gamma(\mathbf{x}) := \{\text{sign}(x_i) \cdot \max(|x_i| - \gamma, 0)\}_{0 \leq i < m}, \quad (1)$$

with γ a positive threshold level. Hard thresholding is defined as

$$\mathcal{H}_\gamma(\mathbf{x}) := \{\max(|x_i| - \gamma, 0) \cdot x_i\}_{0 \leq i < m}. \quad (2)$$

Finally, non-negative Garrote (nng) thresholding is defined as

$$\mathcal{T}_\gamma^{\text{nng}}(\mathbf{x}) := \{\max(|x_i| - \gamma, 0) \cdot (x_i - \gamma^2/x_i)\}_{0 \leq i < m}. \quad (3)$$

The extension to positive varying threshold level is straightforward by replacing γ by γ_i in Eq.'s (1),(2), and (3).

In Madagascar, to soft threshold a dataset with a constant, use e.g.

```
bash$ sfmath n1=100 n2=1 output='1' | sfnoise rep=y >data.rsrf
bash$ sfthr <data.rsrf thr=2 >res1.rsrf
```

or replace the last command by

```
bash$ sfthr <data.rsrf thr=2 method=soft >res2.rsrf
```

This is also equivalent to soft thresholding `data.rsrf` with a vector of same size `mythr.rsrf` whose entries are all set to 2.

```
bash$ sfmath n1=100 n2=1 output='2' >mythr.rsrf
bash$ sfthr <data.rsrf fthr=mythr.rsrf method=soft >res3.rsrf
```

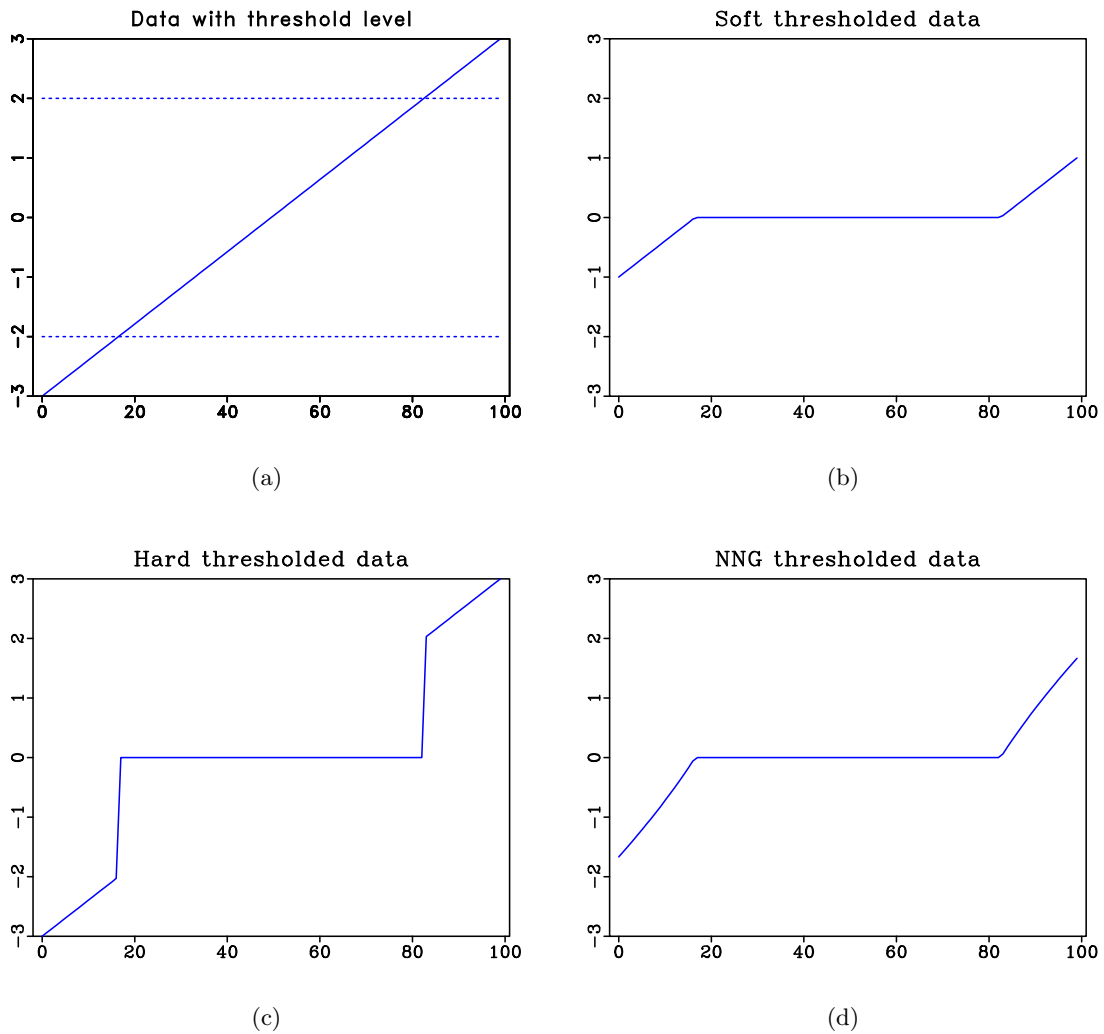


Figure 1: Thresholding example. Line whose range is symmetric about the origin (a) thresholded using soft (b), hard (c), and NNG (d) methods.

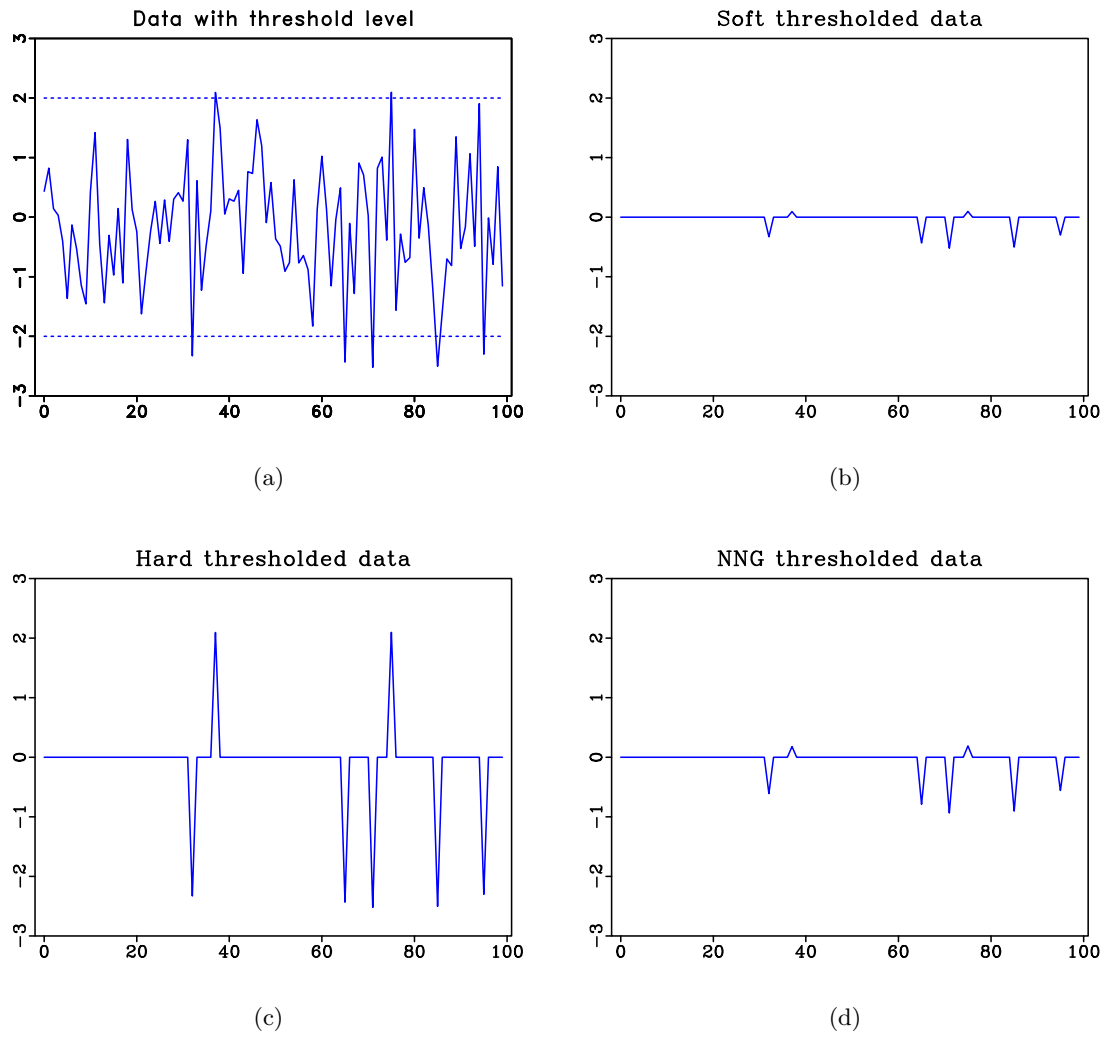


Figure 2: Random vector thresholding example.

If `thr=.5` and `fthr=mythr.rsfsf` are specified at the same time, the effective threshold level is 1, obtained by multiplying `mythr.rsfsf` entries by 0.5

```
bash$ sfthr <data.rsfsf thr=.5 fthr=mythr.rsfsf method=soft >res4.rsfsf
```

Note that thresholding is an element-wise operation. `sfthr` can thus deal with arbitrarily large datasets.

sfsort: Sort a float/complex vector by absolute values.

```
sfsort < in.rsfsf > out.rsfsf memsize=sf_memsize() ascmode=n
```

Written by: Gilles Hennenfent & Henryk Modzelewski, UBC
Created: February 2006

<u>bool</u>	ascmode=n	[y/n]	y=ascending; n=descending
<u>int</u>	memsize=sf_memsize()		Max amount of RAM (in Mb) to be used

`sfsort` is useful for sorting Madagascar vectors either in ascending or descending order with respect to their amplitudes. The sorting is done using `qsort` from `stdlib.h`. This function is an implementation of the quicksort algorithm. `sfsort` has two modes: 1) in-core if the user-specified `memsize` is big enough to load the full dataset in memory and sort it, and else 2) out-of-core. In the latter case, we implemented a divide and conquer approach. The large dataset is first divided into pieces that fit in memory. These pieces are sorted and written to disk in temporary files. The second step is a merge process of the temporary files.

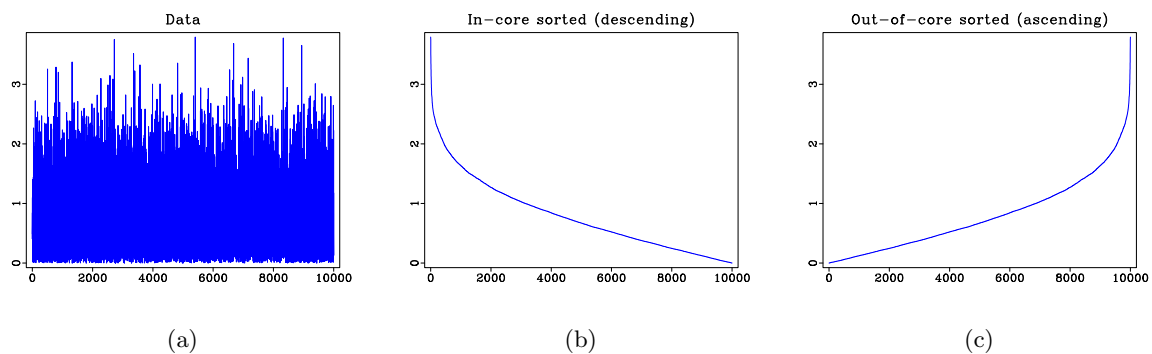


Figure 3: Sorting example.

TRANSFORMS

sffdct:

```
sffdct nbs=4 nba=8 ac=1 adj=n
```

Madagascar wrapper to the Fast Discrete Curvelet Transform (FDCT)

Requirements:

- Python API enable in Madagascar
- PyCurveLab (<https://wave.eos.ubc.ca/Software/Licenced/>)
- CurveLab (<http://www.curvelet.org/>)

<u>bool</u>	ac=1	[y/n]	curvelets at finest scale
<u>bool</u>	adj=n	[y/n]	adjoint transform
<u>int</u>	nba=8		number of angle at the 2nd coarsest scale
<u>int</u>	nbs=4		number of scale for the decomposition

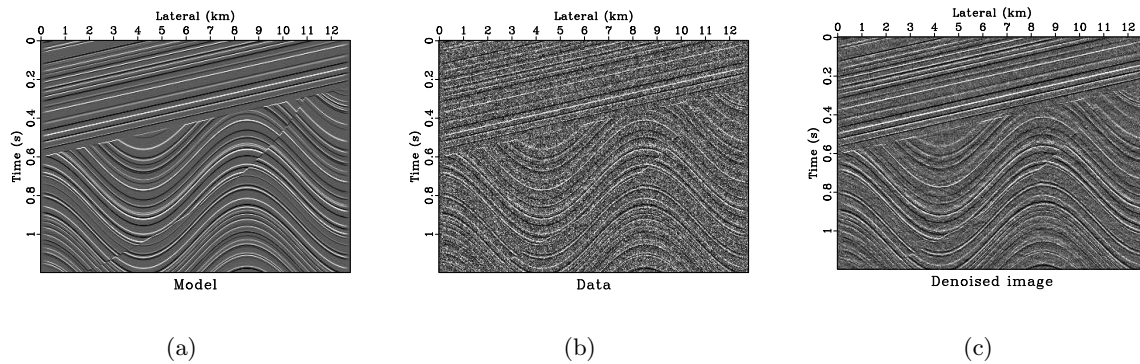


Figure 4: Data denoising. (a) Noise-free data, (b) noisy data, and (c) denoised data using sffdct.