2015 Madagascar Advanced School Discussion

---What Madagascar makes different

Heng Luo, Long Teng, Yong Liu, Kai Xu

Sinopec Geophysical Research Institute, Nanjing, CN

Before Madagascar comes to me

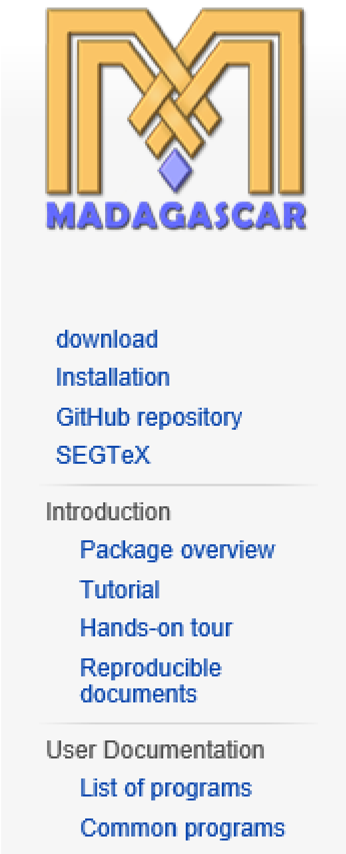
- I used to code C programs and run them directly before. (No parameter card at all)
- I used Word to write papers and some abstracts before.
- I used SeismicUnix Software to plot seismic graph.
- The subroutines used in my programs are mostly coded by myself before.
- It’s painful to use somebody else’s program, even I’m been allowed to use them.
- I nearly need to read through the whole codes when I come back to use the codes being coded several months before.

Then Madagascar

- A classmate talked about Madagascar after he's participated the Madagascar school maybe in 2011.
- In 2013, I entered Sinopec to do the research work. I found it's very difficult to use others' program, because of different platform, different language, different datatype and the parameters is too flexible.
- I started to use Madagascar and proposed to use it in our group, then we can share our programs and save lots of time.
- After that, I found more powerful tools compatible with Madagascar-Latex, and we're trying to use it now.

Learning Madagascar

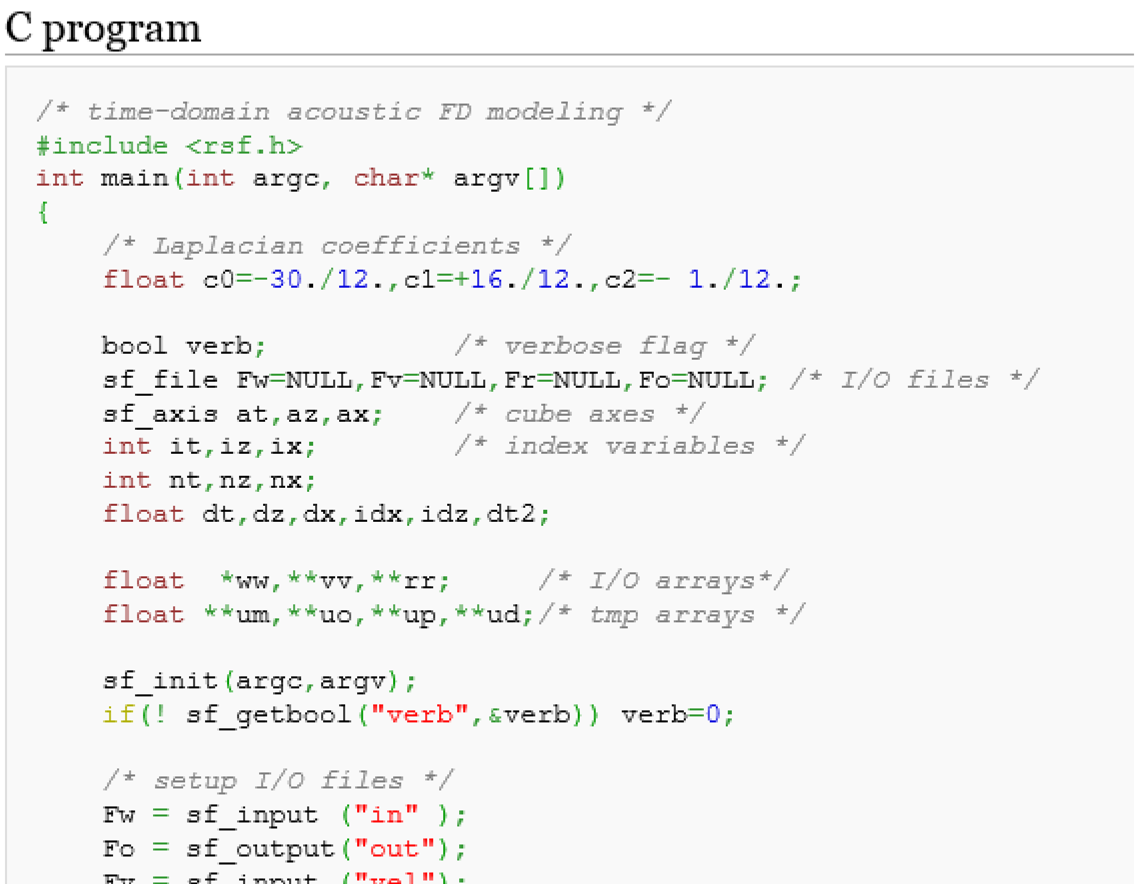
- My colleagues and I don’t have enough time to take Madagascar training courses, so we almost learn it on the internet.



download
Installation
GitHub repository
SEGTeX

Introduction
- Package overview
- Tutorial
- Hands-on tour
- Reproducible documents

User Documentation
- List of programs
- Common programs

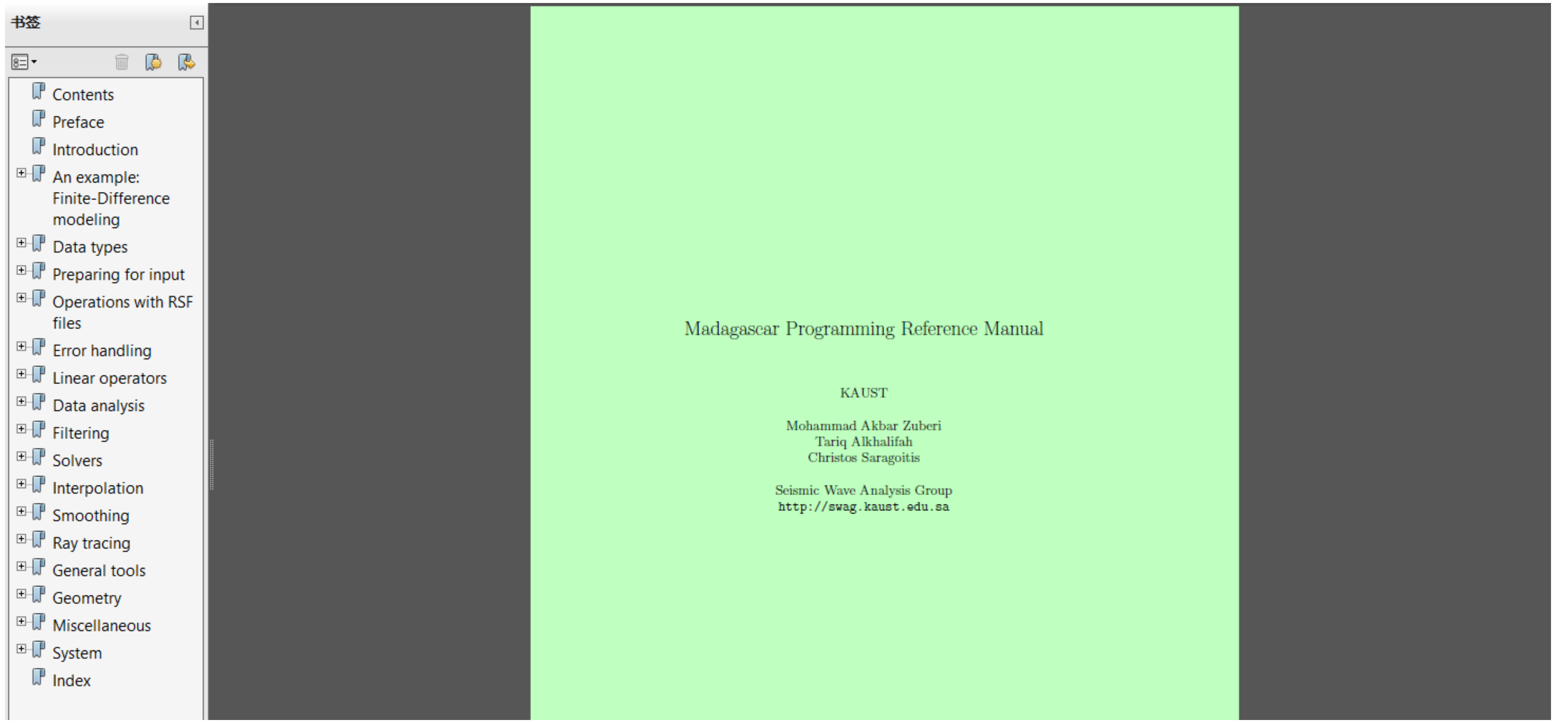


C program

```
/* time-domain acoustic FD modeling */
#include <rsf.h>
int main(int argc, char* argv[])
{
    /* Laplacian coefficients */
    float c0=-30./12.,c1=+16./12.,c2=- 1./12.;

    bool verb;           /* verbose flag */
    sf_file Fw=NULL,Fv=NULL,Fr=NULL,Fo=NULL; /* I/O files */
    sf_axis at,az,ax;    /* cube axes */
    int it,iz,ix;        /* index variables */
    int nt,nz,nx;
    float dt,dz,dx,idx,idz,dt2;

    float  *ww,**vv,**rr;     /* I/O arrays*/
    float **um,**uo,**up,**ud;/* tmp arrays */

    sf_init(argc,argv);
    if(! sf_getbool("verb",&verb)) verb=0;

    /* setup I/O files */
    Fw = sf_input ("in" );
    Fo = sf_output("out");
    Fv = sf_input ("vel");
```

Learning Madagascar

- I started to do some coding from:



Madagascar Programming Reference Manual

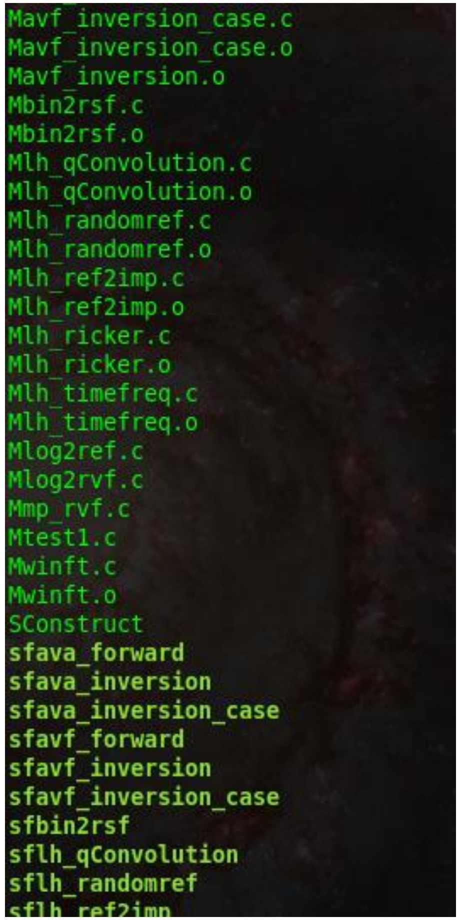
KAUST

Mohammad Akbar Zuberi
Tariq Alkhalifah
Christos Saragoitis

Seismic Wave Analysis Group
http://swag.kaust.edu.sa

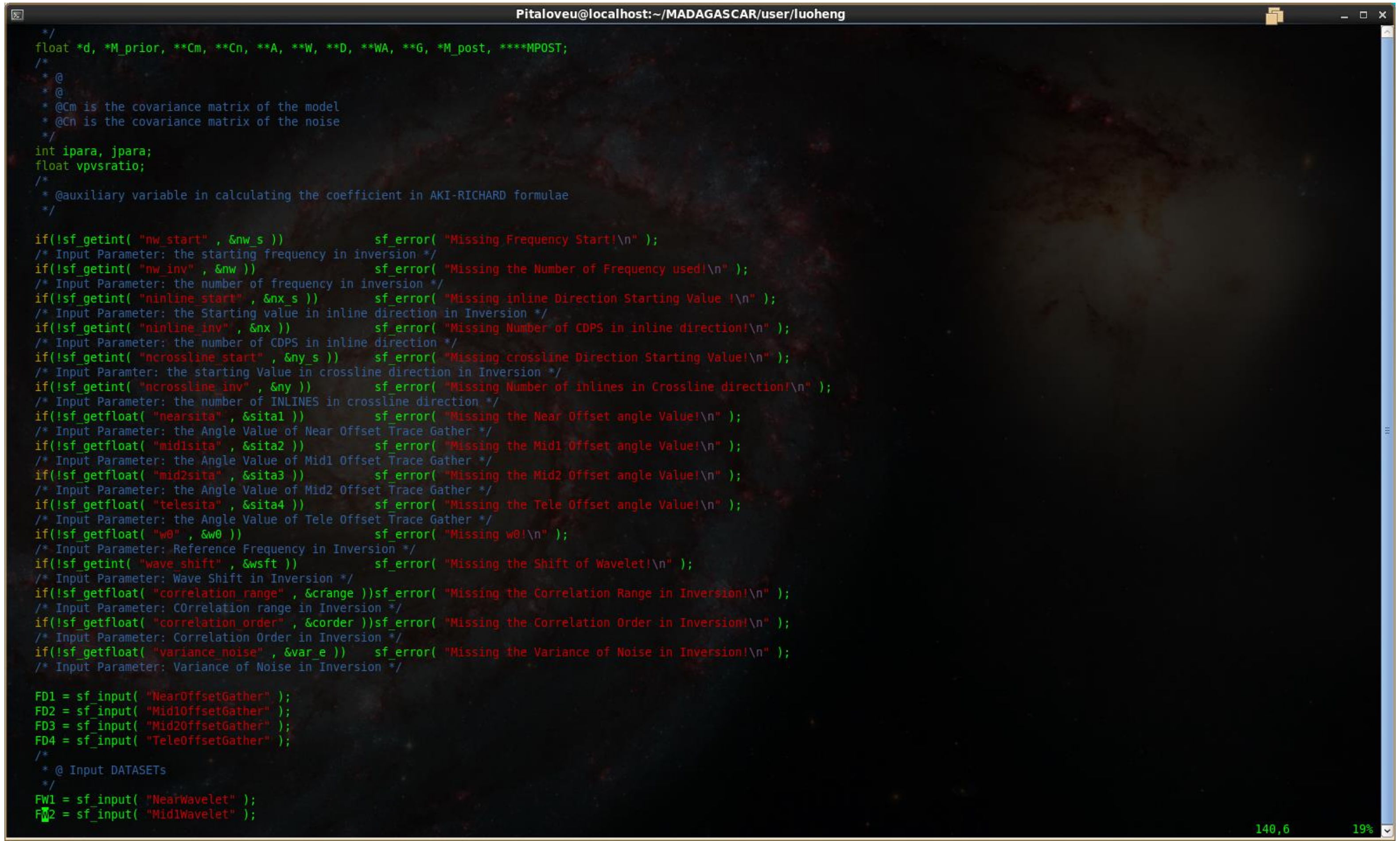
- Zero1float()....zero2float()..

Learning Madagascar



```
Mavf_inversion_case.c
Mavf_inversion_case.o
Mavf_inversion.o
Mbin2rsf.c
Mbin2rsf.o
Mlh_qConvolution.c
Mlh_qConvolution.o
Mlh_randomref.c
Mlh_randomref.o
Mlh_ref2imp.c
Mlh_ref2imp.o
Mlh_ricker.c
Mlh_ricker.o
Mlh_timefreq.c
Mlh_timefreq.o
Mlog2ref.c
Mlog2rvf.c
Mmp_rvf.c
Mtest1.c
Mwinft.c
Mwinft.o
SConstruct
sfava_forward
sfava_inversion
sfava_inversion_case
sfavf_forward
sfavf_inversion
sfavf_inversion_case
sfbin2rsf
sflh_qConvolution
sflh_randomref
sflh_ref2imp
```

- I wrote several programs by using Madagascar,

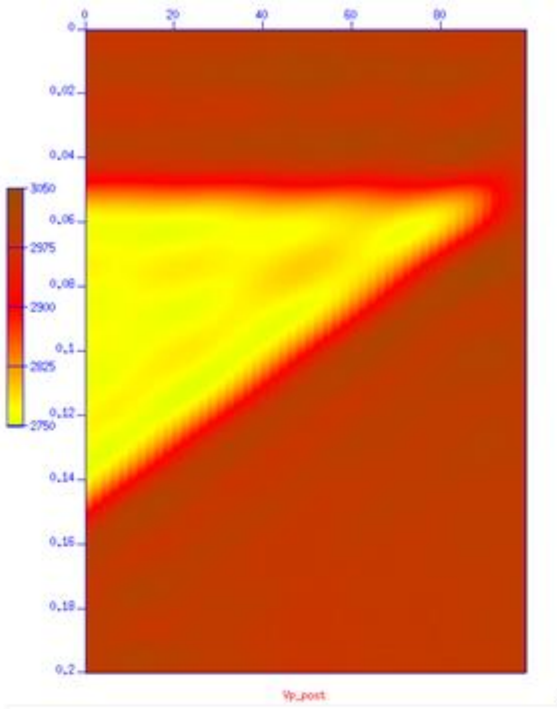


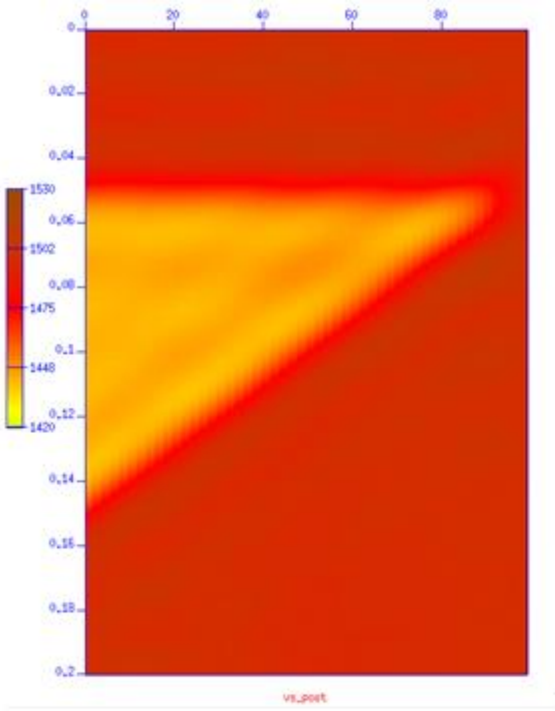
Pitaloveu@localhost:~/MADAGASCAR/user/luoheng

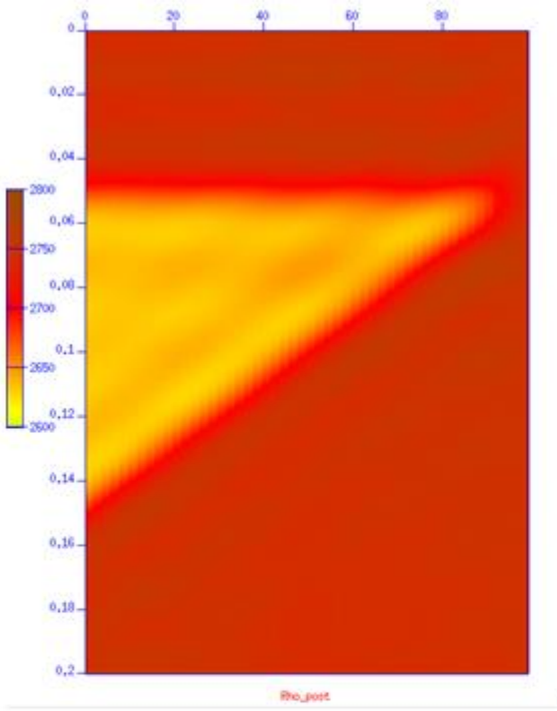
```
 */
float *d, *M_prior, **Cm, **Cn, **A, **W, **D, **WA, **G, *M_post, ****MPOST;
/*
 * @
 * @
 * @Cm is the covariance matrix of the model
 * @Cn is the covariance matrix of the noise
 */
int ipara, jpara;
float vpvsratio;
/*
 * @auxiliary variable in calculating the coefficient in AKI-RICHARD formulae
 */

if(!sf_getint( "nw_start" , &nw_s ))             sf_error( "Missing Frequency Start!\n" );
/* Input Parameter: the starting frequency in inversion */
if(!sf_getint( "nw_inv" , &nw ))                 sf_error( "Missing the Number of Frequency used!\n" );
/* Input Parameter: the number of frequency in inversion */
if(!sf_getint( "ninline_start" , &nx_s ))        sf_error( "Missing inline Direction Starting Value !\n" );
/* Input Parameter: the Starting value in inline direction in Inversion */
if(!sf_getint( "ninline_inv" , &nx ))            sf_error( "Missing Number of CDPS in inline direction!\n" );
/* Input Parameter: the number of CDPS in inline direction */
if(!sf_getint( "ncrossline_start" , &ny_s ))     sf_error( "Missing crossline Direction Starting Value!\n" );
/* Input Paramter: the starting Value in crossline direction in Inversion */
if(!sf_getint( "ncrossline_inv" , &ny ))         sf_error( "Missing Number of inlines in Crossline direction!\n" );
/* Input Parameter: the number of INLINES in crossline direction */
if(!sf_getfloat( "nearsita" , &sita1 ))          sf_error( "Missing the Near Offset angle Value!\n" );
/* Input Parameter: the Angle Value of Near Offset Trace Gather */
if(!sf_getfloat( "mid1sita" , &sita2 ))          sf_error( "Missing the Mid1 Offset angle Value!\n" );
/* Input Parameter: the Angle Value of Mid1 Offset Trace Gather */
if(!sf_getfloat( "mid2sita" , &sita3 ))          sf_error( "Missing the Mid2 Offset angle Value!\n" );
/* Input Parameter: the Angle Value of Mid2 Offset Trace Gather */
if(!sf_getfloat( "telesita" , &sita4 ))          sf_error( "Missing the Tele Offset angle Value!\n" );
/* Input Parameter: the Angle Value of Tele Offset Trace Gather */
if(!sf_getfloat( "w0" , &w0 ))                   sf_error( "Missing w0!\n" );
/* Input Parameter: Reference Frequency in Inversion */
if(!sf_getint( "wave_shift" , &wsft ))           sf_error( "Missing the Shift of Wavelet!\n" );
/* Input Parameter: Wave Shift in Inversion */
if(!sf_getfloat( "correlation_range" , &crange ))sf_error( "Missing the Correlation Range in Inversion!\n" );
/* Input Parameter: COrrelation range in Inversion */
if(!sf_getfloat( "correlation_order" , &corder ))sf_error( "Missing the Correlation Order in Inversion!\n" );
/* Input Parameter: Correlation Order in Inversion */
if(!sf_getfloat( "variance_noise" , &var_e ))    sf_error( "Missing the Variance of Noise in Inversion!\n" );
/* Input Parameter: Variance of Noise in Inversion */

FD1 = sf_input( "NearOffsetGather" );
FD2 = sf_input( "Mid1OffsetGather" );
FD3 = sf_input( "Mid2OffsetGather" );
FD4 = sf_input( "TeleOffsetGather" );
/*
 * @ Input DATASETs
 */
FW1 = sf_input( "NearWavelet" );
FW2 = sf_input( "Mid1Wavelet" );
                                                                                          140,6          19%
```

Learning Madagascar

- We used the Madagascar program to do AVF inversion, and the graphs are ploted by SU. (The plot pros in Madagascar is….)

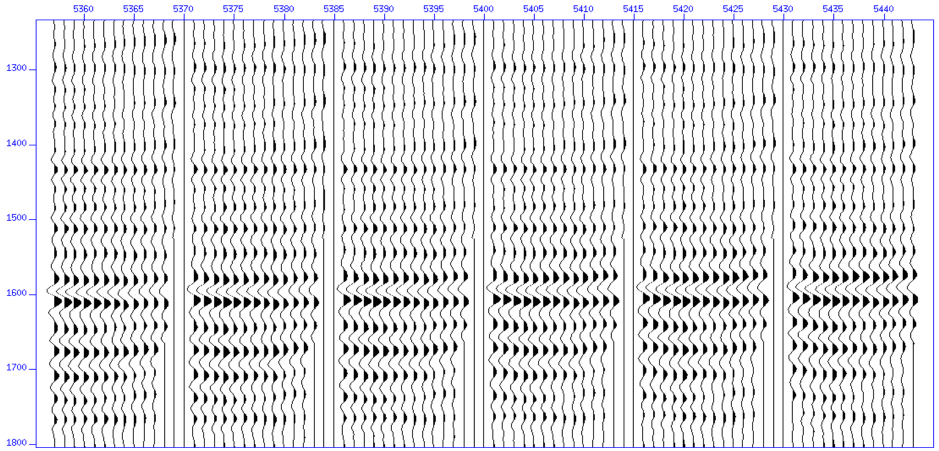


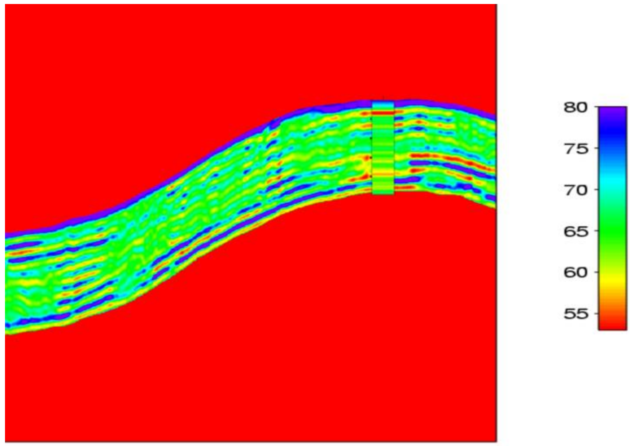




Learning Madagascar

- Dealing with field data.

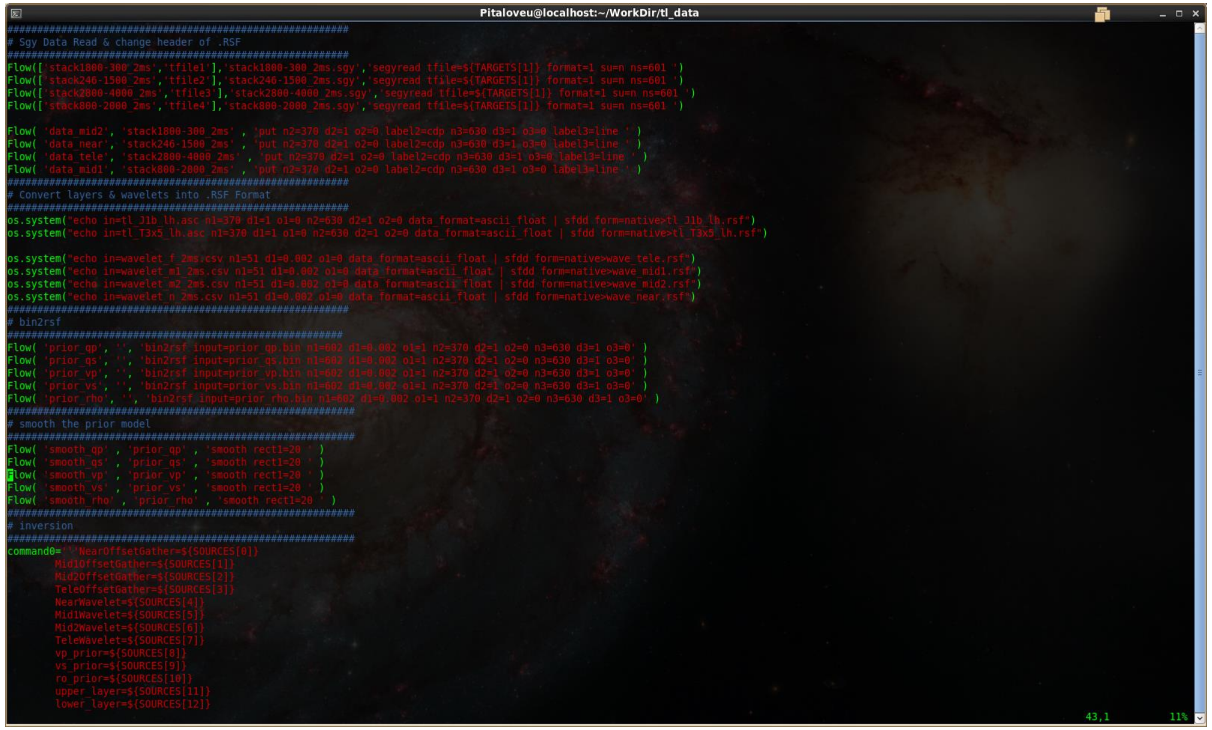




QP_INVERSION

Learning Madagascar

- Dealing with field data:

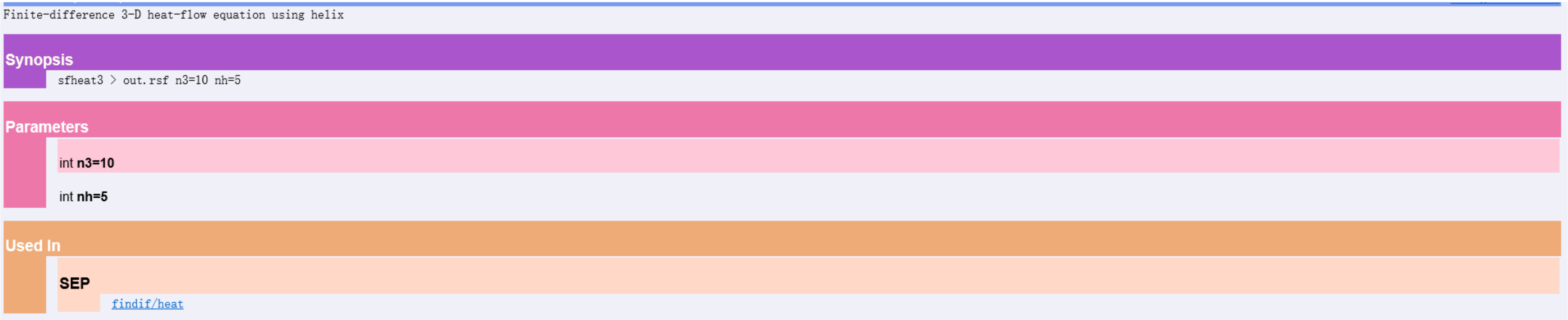


```
##########################################################
# Sgy Data Read & change header of .RSF
##########################################################
Flow(['stack1800-300_2ms','tfile1'],'stack1800-300_2ms.sgy','segyread tfile=${TARGETS[1]} format=1 su=n ns=601 ')
Flow(['stack246-1500_2ms','tfile2'],'stack246-1500_2ms.sgy','segyread tfile=${TARGETS[1]} format=1 su=n ns=601 ')
Flow(['stack2800-4000_2ms','tfile3'],'stack2800-4000_2ms.sgy','segyread tfile=${TARGETS[1]} format=1 su=n ns=601 ')
Flow(['stack800-2000_2ms','tfile4'],'stack800-2000_2ms.sgy','segyread tfile=${TARGETS[1]} format=1 su=n ns=601 ')

Flow( 'data_mid2', 'stack1800-300_2ms' , 'put n2=370 d2=1 o2=0 label2=cdp n3=630 d3=1 o3=0 label3=line ' )
Flow( 'data_near', 'stack246-1500_2ms' , 'put n2=370 d2=1 o2=0 label2=cdp n3=630 d3=1 o3=0 label3=line ' )
Flow( 'data_tele', 'stack2800-4000_2ms' , 'put n2=370 d2=1 o2=0 label2=cdp n3=630 d3=1 o3=0 label3=line ' )
Flow( 'data_mid1', 'stack800-2000_2ms' , 'put n2=370 d2=1 o2=0 label2=cdp n3=630 d3=1 o3=0 label3=line ' )
##########################################################
# Convert layers & wavelets into .RSF Format
##########################################################
os.system("echo in=tl_J1b_lh.asc n1=370 d1=1 o1=0 n2=630 d2=1 o2=0 data_format=ascii_float | sfdd form=native>tl_J1b_lh.rsf")
os.system("echo in=tl_T3x5_lh.asc n1=370 d1=1 o1=0 n2=630 d2=1 o2=0 data_format=ascii_float | sfdd form=native>tl_T3x5_lh.rsf")

os.system("echo in=wavelet_f_2ms.csv n1=51 d1=0.002 o1=0 data_format=ascii_float | sfdd form=native>wave_tele.rsf")
os.system("echo in=wavelet_m1_2ms.csv n1=51 d1=0.002 o1=0 data_format=ascii_float | sfdd form=native>wave_mid1.rsf")
os.system("echo in=wavelet_m2_2ms.csv n1=51 d1=0.002 o1=0 data_format=ascii_float | sfdd form=native>wave_mid2.rsf")
os.system("echo in=wavelet_n_2ms.csv n1=51 d1=0.002 o1=0 data_format=ascii_float | sfdd form=native>wave_near.rsf")
##########################################################
# bin2rsf
#########################################################
Flow( 'prior_qp', '', 'bin2rsf input=prior_qp.bin n1=602 d1=0.002 o1=1 n2=370 d2=1 o2=0 n3=630 d3=1 o3=0' )
Flow( 'prior_qs', '', 'bin2rsf input=prior_qs.bin n1=602 d1=0.002 o1=1 n2=370 d2=1 o2=0 n3=630 d3=1 o3=0' )
Flow( 'prior_vp', '', 'bin2rsf input=prior_vp.bin n1=602 d1=0.002 o1=1 n2=370 d2=1 o2=0 n3=630 d3=1 o3=0' )
Flow( 'prior_vs', '', 'bin2rsf input=prior_vs.bin n1=602 d1=0.002 o1=1 n2=370 d2=1 o2=0 n3=630 d3=1 o3=0' )
Flow( 'prior_rho', '', 'bin2rsf input=prior_rho.bin n1=602 d1=0.002 o1=1 n2=370 d2=1 o2=0 n3=630 d3=1 o3=0' )
###########################################################
# smooth the prior model
###########################################################
Flow( 'smooth_qp' , 'prior_qp' , 'smooth rect1=20 ' )
Flow( 'smooth_qs' , 'prior_qs' , 'smooth rect1=20 ' )
Flow( 'smooth_vp' , 'prior_vp' , 'smooth rect1=20 ' )
Flow( 'smooth_vs' , 'prior_vs' , 'smooth rect1=20 ' )
Flow( 'smooth_rho' , 'prior_rho' , 'smooth rect1=20 ' )
###########################################################
# inversion
###########################################################
command0='''NearOffsetGather=${SOURCES[0]}
        Mid1OffsetGather=${SOURCES[1]}
        Mid2OffsetGather=${SOURCES[2]}
        TeleOffsetGather=${SOURCES[3]}
        NearWavelet=${SOURCES[4]}
        Mid1Wavelet=${SOURCES[5]}
        Mid2Wavelet=${SOURCES[6]}
        TeleWavelet=${SOURCES[7]}
        vp_prior=${SOURCES[8]}
        vs_prior=${SOURCES[9]}
        ro_prior=${SOURCES[10]}
        upper_layer=${SOURCES[11]}
        lower_layer=${SOURCES[12]}
                                                                        43,1          11%
```

- How to use the Madagascar codes directly in a different platform or bring into a software package.

Learning Madagascar

- Using the programs written by developers is too hard. The intros are too vague.



Finite-difference 3-D heat-flow equation using helix

Synopsis

```
sfheat3 > out.rsf n3=10 nh=5
```

Parameters

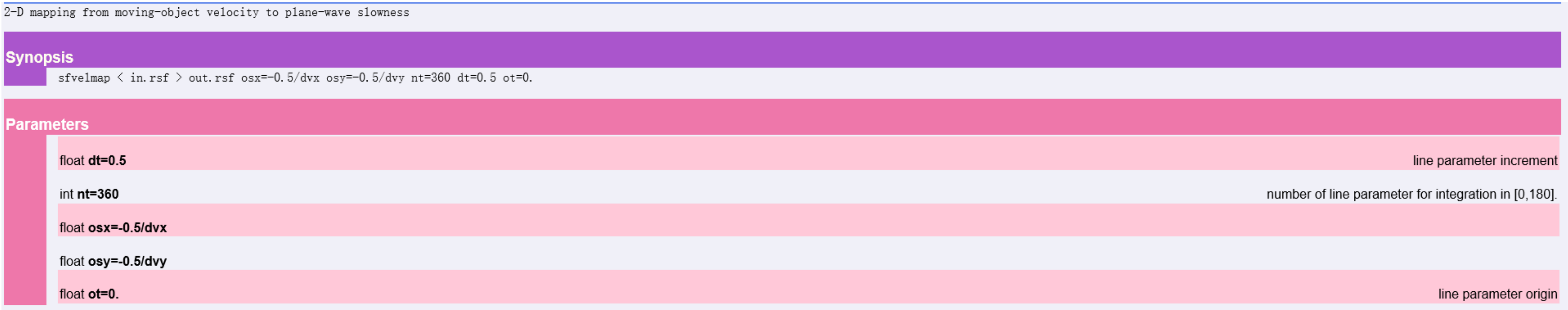
int **n3=10**

int **nh=5**

Used In

SEP

findif/heat



2-D mapping from moving-object velocity to plane-wave slowness

Synopsis

```
sfvelmap < in.rsf > out.rsf osx=-0.5/dvx osy=-0.5/dvy nt=360 dt=0.5 ot=0.
```

Parameters

float **dt=0.5**	line parameter increment
int **nt=360**	number of line parameter for integration in [0,180].
float **osx=-0.5/dvx**	
float **osy=-0.5/dvy**	
float **ot=0.**	line parameter origin

Learning Madagascar

- Converting pure binary data into a .rsf data is unavailable in the original Madagascar.

After using Madagascar

- Use Sconstruct file to rerun a program and change the parameters.
- Trying to use Latex and the given templates to write papers.
- Still using SeismicUnix do the plotting part.
- Use the subroutines provided by Madagascar or some opensource software.
- Share the programs between colleagues.
- The code is much more easier to be managed, I don’t need to edit them any more after I finished them in the first time.

Some suggestions about Madagascar

- Tell more details about every individual programs developed by different authors. More details about the meaning of every parameters used in the command line.
- The plotting programs are not so flexible.
- Categorize the programs by authors, but also by functionality, So we could use them more convenient.
- If it’s possible, a short video about how to use Madagascar will be better. ^=^

Thank you very much!