Introducing Madagascar

A Computational Platform for Geophysical Data Processing and Reproducible Numerical Experiments

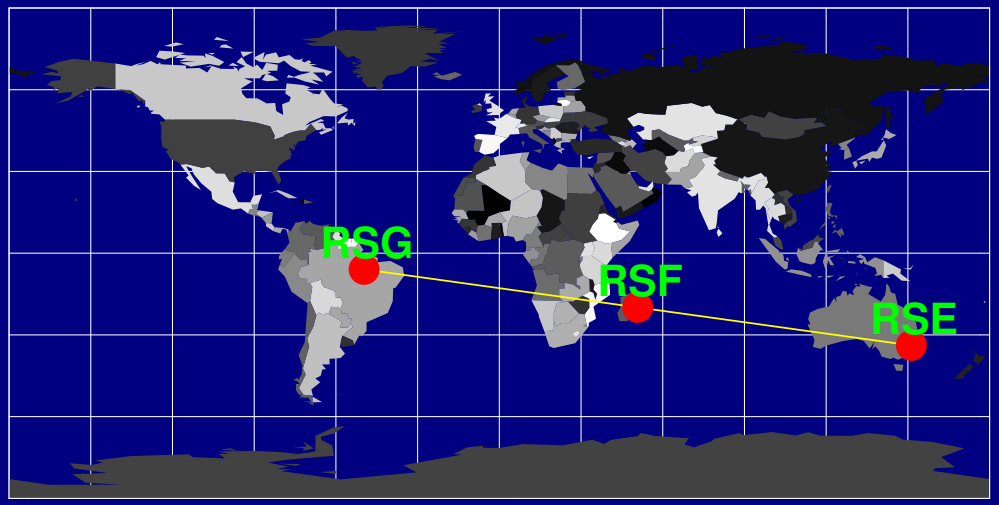
Sergey Fomel[1] Paul Sava[2] Felix Herrmann[3]

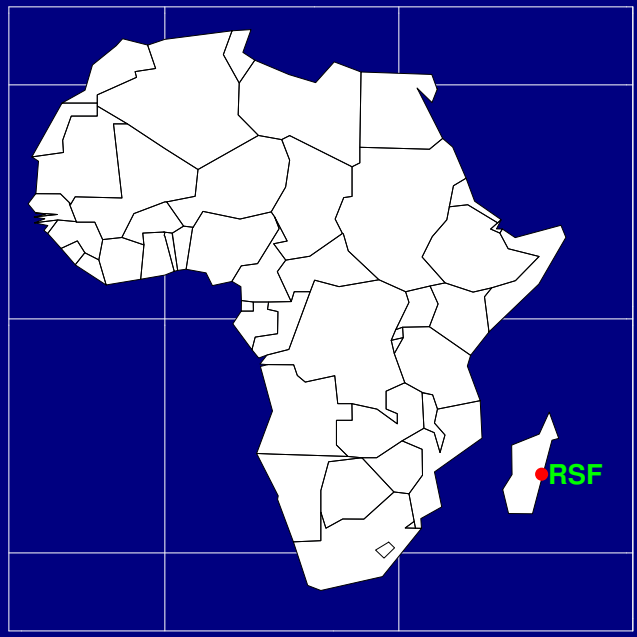
[1]Bureau of Economic Geology
Jackson School of Geosciences
University of Texas at Austin

[2]Colorado School of Mines
[3]University of British Columbia

June 11, 2006

From RSF to Madagascar





Outline

Outline

Outline

Outline

Outline

History

- Started in June 2003
- 6 main contributors
- Version 0.9 released June 2006
- Development version
  - 2,000 revisions
  - 300 main programs (140,000 lines of C)
  - 3,000 tests (40,000 lines of Python)
  - 30 papers (20,000 lines of LaTeX)
- `http://rsf.sf.net/`

Heritage

- SEPlib
  - Rob Clayton, Jon Claerbout, Dave Hale, Stew Levin, Rick Ottolini, Joe Dellinger, Steve Cole, Dave Nichols, Martin Karrenbach, Biondo Biondi, Bob Clapp
- SEP reproducible research system
  - Martin Karrenbach, Matthias Schwab, Joel Schroeder, Sergey Fomel, Bob Clapp
- Seismic Unix
- DDS

Outline

Test Driven Development

- XP, agile software engineering, refactoring
  - Tests drive development
- Computational geophysics
  - Tests are numerical experiments
- Science
  - Reproducible experiments mean progress

RSF: RSF Programs - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://egl.beg.utexas.edu/RSF/

Red Hat, Inc. Red Hat Network Support Shop Products Training

RSF Programs

Add a search feature later.

filt/imag

sfafdm2d	sffinstack	sfmutter	sfsrmod
sfborn2d	sffkamo	sfpreconstkirch	sfsrmva
sfc2r	sffkdmo	sfrays2	sfsrsyn
sfcamig	sfgazdag	sfreg2tri	sfsstep2
sfcell2	sfhalfint	sfremap1	sftrace2
sfcgscan	sfhwt2d	sfricker1	sftrapez
sfconstfdmig2	sfhwtex	sfrweab	sftri2reg
sfcube2list	sfinterp2	sfrwemete2d	sftrirand
sfdistance	sfinterp3	sfrwesrmig	sfunif2
sfdmo	sfinterpt	sfrwezomig	sfunif3
sfdsr	sfkirchinv	sfs2ofz	sfvoft
sfdsr2	sfkirchnew	sfshoot2	sfvofz
sfeikonal	sfkirmod	sfslant	sfzomig
sfeikonalvti	sfkirmod3	sfsrmig	sfzomva
sffincon	sfmig45	sfsrmig2	

filt/main

sfadd	sfdisfil	sfinterleave	sfrotate
sfattr	sfdottest	sfmask	sfrtoc
sfcat	sfget	sfmath	sfscale
sfcconjgrad	sfheaderattr	sfmerge	sfsegyread
sfcdottest	sfheadercut	sfmv	sfsegywrite
sfcmplx	sfheadermath	sfpad	sfspike
sfconjgrad	sfheadersort	sfput	sfspray
sfcp	sfheaderwindow	sfreal	sfstack
sfcut	sfimag	sfreverse	sftransp
sfdd	sfin	sfrm	sfwindow

Done

RSF: sfeikonal - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://egl.beg.utexas.edu/RSF/sfeikonal.html

Red Hat, Inc. Red Hat Network Support Shop Products Training

sfeikonal (Meikonal.c 1507 2005-10-22 04:01:28Z savap)

index

filt/imag/Meikonal.c

Fast marching eikonal solver (3-D).

Synopsis

sfeikonal < vel.rsf > time.rsf shotfile=shots.rsf vel=y order=2 br1=d1 br2=d2 br3=d3 plane1=n plane2=n plane3=n b1=plane[2]? n1: (int) (br1/d1+0.5) b2=plane[1]? n2: (int) (br2/d2+0.5) b3=plane[0]? n3: (int) (br3/d3+0.5) zshot=0. yshot=o2 + 0.5*(n2-1)*d2 xshot=o3 + 0.5*(n3-1)*d3 shotfile=

Parameters

int **b1=plane[2]? n1: (int) (br1/d1+0.5)**

int **b2=plane[1]? n2: (int) (br2/d2+0.5)**

int **b3=plane[0]? n3: (int) (br3/d3+0.5)** — Constant-velocity box around the source (in samples)

float **br1=d1**

float **br2=d2**

float **br3=d3** — Constant-velocity box around the source (in physical dimensions)

int **order=2** [1,2] — Accuracy order

bool **plane1=n** [y/n]

bool **plane2=n** [y/n]

bool **plane3=n** [y/n] — plane-wave source

Done

bool **plane3=n** [y/n] — plane-wave source

string **shotfile=** — File with shot locations (n2=number of shots, n1=3)

bool **vel=y** [y/n] — if y, the input is velocity; n, slowness squared

float **xshot=o3 + 0.5*(n3-1)*d3**

float **yshot=o2 + 0.5*(n2-1)*d2**

float **zshot=0.** — Shot location (used if no shotfile)

Used In

GEO391

- hw3/sigsbee2
- hw3/sigsbee

GTI

- fdmod/ziggyFDM
- multi/tree
- multi/basic
- timec/anal
- wavem/dsr
- wavem/imp

SEP

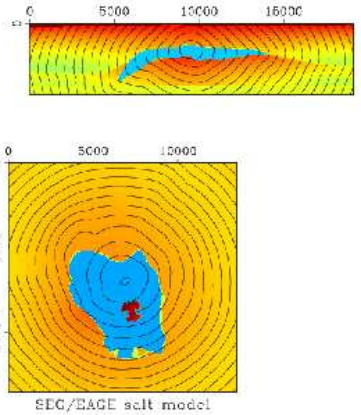
- fmeiko/fmarch
- fmsec/cvel
- fmsec/marm

```
from rsfproj import *
import math

Fetch('salt_slow_desp.HH','segeage')
Flow('salt','salt_slow_desp.HH','dd form=native')

Flow('seg','salt','window j1=2 j2=2 j3=2 | eikonal vel=0 zshot=1500')

def traverse(n2,n3,d2,d3):
    return 'put n2=%d n3=1 d2=%g | window j2=%d' % \
           (n2*n3,math.hypot(d2,d3)/(n2+1),n2+1)

Flow('st','salt',traverse(676,676,20,20))
Flow('sgt','seg',traverse(338,338,10,10))

Plot('sxy','salt',
     '''
     window n1=1 f1=75 |
     grey pclip=100 gpow=1 bias=0.00034
     color=j scalebar=n
     title="SEG/EAGE salt model" screenratio=1 screenht=9.6571
     ''')
Plot('exy','seg',
     '''
     window n1=1 f1=75 |
     contour nc=100 plotcol=7 wantaxis=n wanttitle=n
     screenratio=1 screenht=9.6571
     ''')
Plot('cxy','sxy exy','Overlay')

Plot('sxz','st',
     '''
     grey pclip=100 gpow=1 bias=0.00034
     color=j scalebar=n wanttitle=n screenratio=0.21966 screenht=3
     ''')
Plot('exz','sgt',
     '''
     contour nc=100 plotcol=7 wantaxis=n wanttitle=n
     screenratio=0.21966 screenht=3
     ''')
Plot('cxz','sxz exz','Overlay')

Result('salt','cxz cxy','OverUnderIso')

End()
```

sfdd sfeikonal sfgrey

according to the first-order difference operator (1). As a result, the computational error of this method goes to zero with the decrease in the grid size in a linear fashion. The proof of validity of the method (omitted here) is also analogous to that of Dijkstra's algorithm (Sethian, 1996a; Sethian, 1996b). As in most of the shortest-path implementations, the computational cost of extracting the minimum point at each step of the algorithm is greatly reduced [from $O(N)$ to $O(\log N)$ operations] by maintaining a priority-queue structure (heap) for the *NarrowBand* points (Cormen et al., 1990).

Figure 1 shows an example application of the fast marching eikonal solver on the three-dimensional SEG/EAGE salt model. The computation is stable despite the large velocity contrasts in the model. The current implementation takes about 10 seconds for computing a 100x100x100 grid on one node of SGI Origin 200. Alkhalifah and Fomel (1997) discuss the differences between Cartesian and polar coordinate implementations.



salt

Figure 1. Constant-traveltime contours of the first-arrival traveltime, computed in the SEG/EAGE salt model. A point source is positioned inside the salt body. The top plot is a diagonal slice; the bottom plot, a depth slice.

The difference equation (1) is a finite-difference approximation to the continuous eikonal equation

$$\left(\frac{\partial t}{\partial x}\right)^2 + \left(\frac{\partial t}{\partial y}\right)^2 + \left(\frac{\partial t}{\partial z}\right)^2 = s^2(x,y,z)\ , \tag{2}$$

where x, y, and z represent the spatial Cartesian coordinates. In the next two sections, I show how the updating procedure can be derived without referring to the eikonal equation, but with the direct use of Fermat's principle.

A variational formulation of the fast marching eikonal solver

Next: The theoretic grounds of **Up:** Fomel: Fast marching **Previous:** Introduction

2005-07-30

SourceForge.net SVN Repository - [rsf] View of /trunk/filt/imag/Meikonal.c - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://svn.sourceforge.net/viewcvs.cgi/rsf/trunk/filt/imag/Meikonal.c?view=markup

Red Hat, Inc. Red Hat Network Support Shop Products Training

[rsf] / trunk / filt / imag / Meikonal.c

View of /trunk/filt/imag/Meikonal.c

Parent Directory | Revision Log

Revision 1507 - (download) (annotate)

Sat Oct 22 04:01:28 2005 UTC (8 months ago) by *savap*

File size: 4177 byte(s)

```
/* Fast marching eikonal solver (3-D). */
/*
  Copyright (C) 2004 University of Texas at Austin

  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.

  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
*/

#include <math.h>

#include<rsf.h>

#include "fastmarch.h"

int main (int argc,char* argv[])
{
    int b1, b2, b3, n1, n2, n3, i, nshot, ndim, is,order,n123, *p;
    float br1, br2, br3, o1, o2, o3, d1, d2, d3, slow;
    float **s, *t, *v;
    char *sfile;
    bool isvel, plane[3];
    sf_file vel, time, shots;

    sf_init (argc, argv);
    vel = sf_input("in");
    time = sf_output("out");

    if (SF_FLOAT != sf_gettype(vel))
        sf_error("Need float input");
    if(!sf_histint(vel,"n1",&n1)) sf_error("No n1= in input");
```

Done

Inspirational Quotes

Abandoning the habit of secrecy in favor of process transparency and peer review was the crucial step by which alchemy became chemistry. In the same way, it is beginning to appear that open-source development may signal the long-awaited maturation of software development as a discipline.
Eric S. Raymond, The art of UNIX programming, 2004

Inspirational Quotes

The purpose of reproducible research is to facilitate someone going a step further by changing something. The first step that someone will want to make is to be sure that your work is reproducible before they change and improve upon it.
Jon F. Claerbout, Reproducible research

Inspirational Quotes

An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

Jon B. Buckheit and David L. Donoho, WaveLab and reproducible research, 1995

Inspirational Quotes

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... Where else in science can one get away with publishing observations that are claimed to prove a theory or illustrate the success of a technique without having to give a careful description of the methods used, in sufficient detail that others can attempt to repeat the experiment?

Randall J. LeVeque, Wave propagation software, computational science, and reproducible research, 2006

Reproducible Research Tools

- SEP
  - GNU make rules
  - Perl scripts
  - Shell scripts
  - LaTeX macros
- Madagascar
  - SCons (Python replacement for make)

Outline

Open Source

- Open source means freedom to the user
  - Freedom to use
  - Freedom to study and modify
  - Freedom to redistribute
  - Freedom to improve
- Open source means collaboration and peer review
- Madagascar package is 100% open-source
  - GPL license
  - Hosted by Sourceforge
  - Developed with Subversion

Outline

Universal File Format

- Borrowed from classic SEPlib
- Binary data and test headers
- N-dimensional hypercubes
- Can represent seismic data (regular, irregular, 3-D, 9-D, etc.) as well as any other kind of data

Inspirational Quotes

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.

Eric S. Raymond, The art of UNIX programming, 2004

Summary

- Madagascar is a software package for geophysical data processing and reproducible numerical experiments.
- New, test-driven, open-source, using a universal file format

Please Visit

- `http://rsf.sf.net`
- School and workshop
  - *Reproducible Research in Computational Geophysics*
  - August 30-31, 2006
  - Vancouver BC, Canada