Advanced Madagascar School
Qingdao, China   August 8-9, 2015

Writing programs in C, C++, and F90

Zhiguang Xue
Ph.D. student in The University of Texas at Austin

For me, it's a high-efficiency research tool!

Learn/create new ideas ← Many reproducible papers are hanged on Madagascar website.

↓

Implementation
(programming) ← Madagascar includes rich APIs of common programming languages.

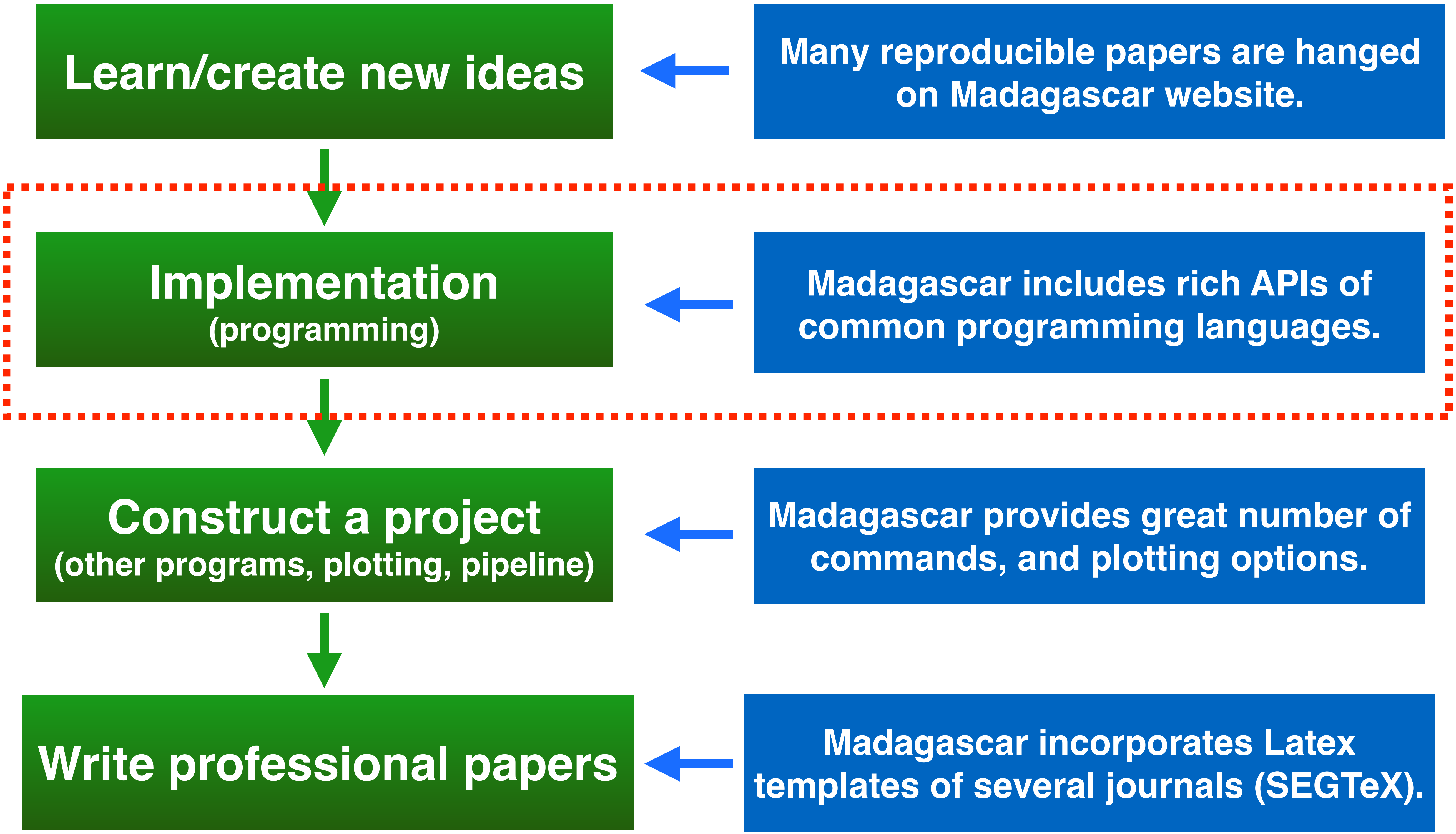
↓

Construct a project
(other programs, plotting, pipeline) ← Madagascar provides great number of commands, and plotting options.

↓

Write professional papers ← Madagascar incorporates Latex templates of several journals (SEGTeX).

For me, it's a high-efficiency research tool!



Madagascar API

- **Application Programming Interface (API)**
  - a set of routines, protocols, and tools for building software applications;
  - often come in the form of a library that includes specifications for routines, data structures, object classes, and variables.

Command: ls $RSFSRC/api/

- C
- C++
- F90
- Matlab
- F77
- Python
- Java
- Octave

```
/* Velocity file */
sf_file Fvel;
/* Velocity variable */
float **vel;
/* Allocate storage */
vel=sf_floatalloc2(n1, n2);
/* Read data from file to variable */
sf_floatread(vel[0], n1*n2, Fvel);
```

A snippet of Madagascar C program

Madagascar API

- **Application Programming Interface (API)**
  - a set of routines, protocols, and tools for building software applications;
  - often come in the form of a library that includes specifications for routines, data structures, object classes, and variables.

Command: ls $RSFSRC/api/

- C
- C++
- F90
- Matlab
- F77
- Python
- Java
- Octave

```c
/* Velocity file */
sf_file Fvel;
/* Velocity variable */
float **vel;
/* Allocate storage */
vel=sf_floatalloc2(n1, n2);
/* Read data from file to variable */
sf_floatread(vel[0], n1*n2, Fvel);
```

A snippet of Madagascar C program

Outline

— A quick tour of APIs

- ✦ Preparations
- ✦ Two examples
- ✦ Madagascar API
  - C
  - C++
  - F90
- ✦ Exercise (Born modeling)

Outline

— A quick tour of APIs

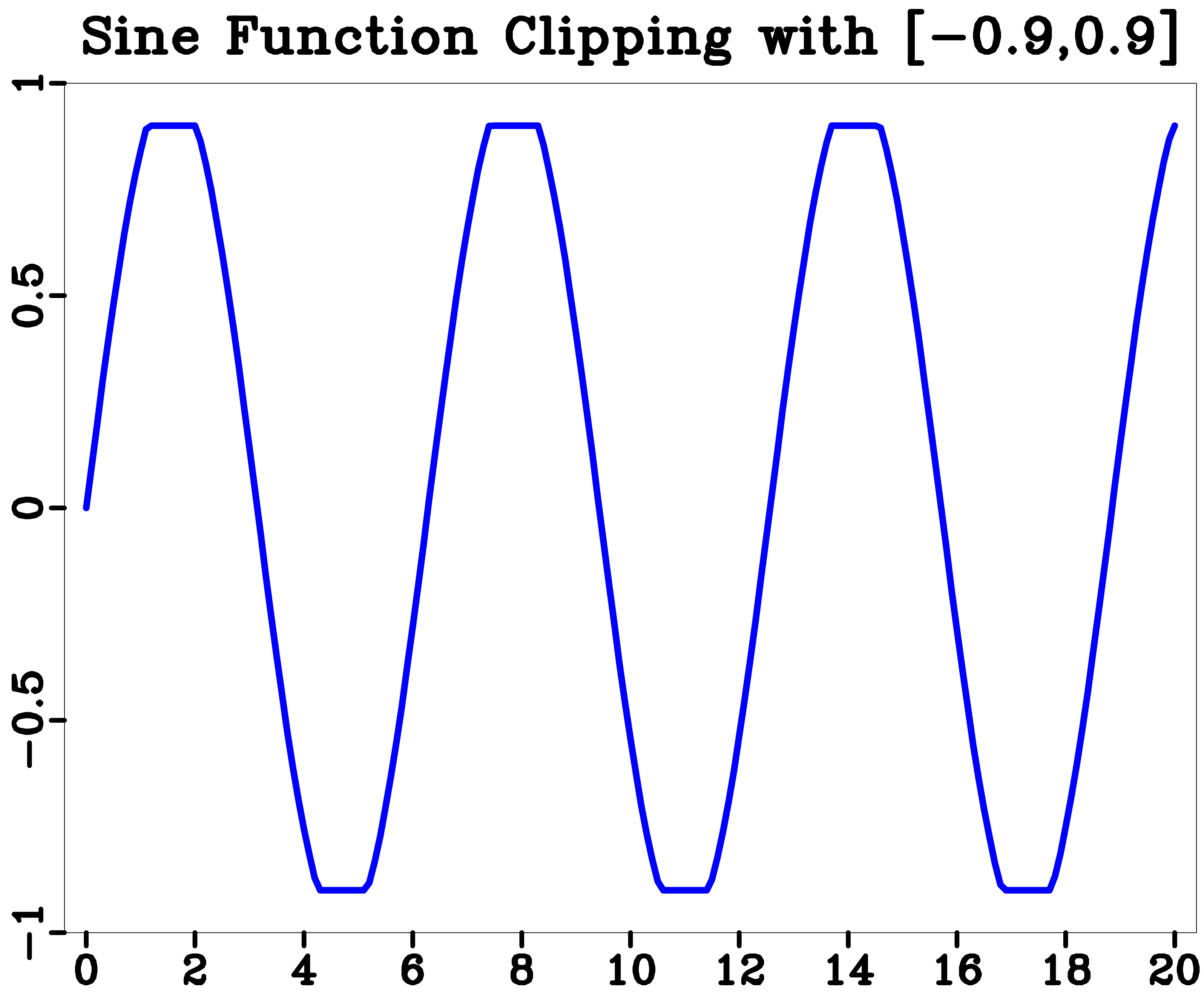
- Preparations
- Two examples
- Madagascar API
  - C
  - C++
  - F90
- Exercise (Born modeling)

1st Thing: Installation check

- C, C++ and F90 compilers (command: *which compiler*)
  - GNU: gcc, g++, and gfortran
  - Intel: icc, icpc, and ifort
- ./configure API = c(default), c++, f90 -- prefix=$RSFROOT
- Check libraries (command: *cd $RSFROOT/lib*)
  - rsf(c), rsf++(c++), and rsff90(f90)
- Test with attached examples
  - C directory, and run *scons clip1.view*
  - C++ directory, and run *scons clip1.view*
  - F90 directory, and run *scons clip1.view*

1st Thing: Installation check

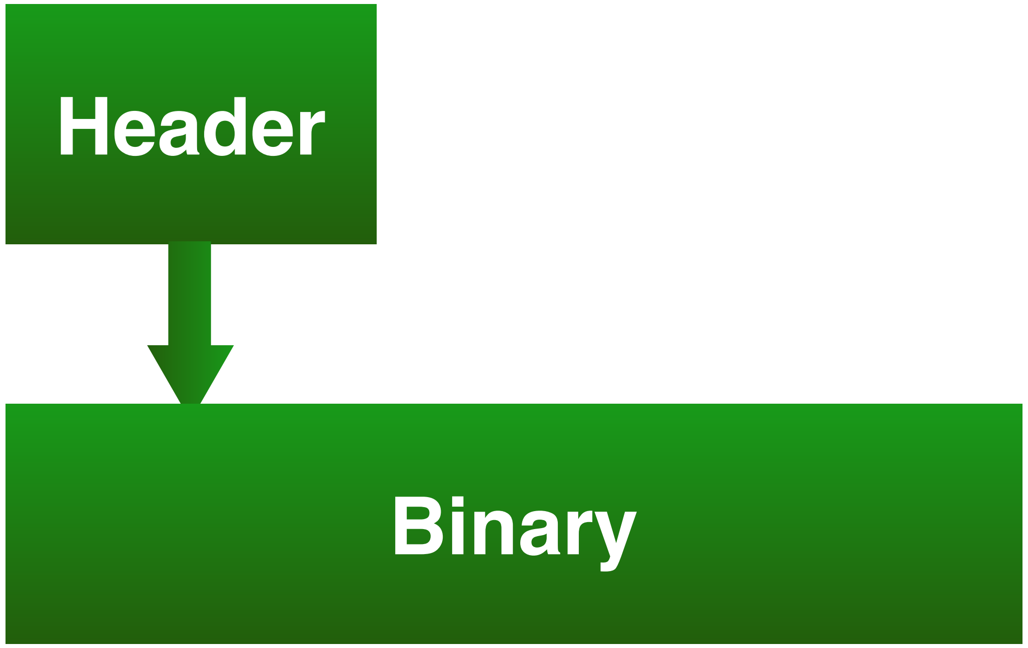
run *scons clip1.view*



If the above figure is obtained, the installation is successful!

2nd Thing: Madagascar data format

- **Regularly Sampled Format (RSF)**
  - **Header (data.rsf)**
  - **Binary (data.rsf@)**



Cartoon of the RSF file format

Example:

```
Zhiguangs-MacBook-Pro:C zhiguang$ sfin vel.rsf
vel.rsf:
    in="/var/tmp/Desktop/mada_school/C/vel.rsf@"
    esize=4 type=float form=native
    n1=201          d1=0.01         o1=0            label1="x1" unit1="km"
    n2=301          d2=0.01         o2=0            label2="x2" unit2="km"
  60501 elements 242004 bytes
```

Make sure the header information is correct before using it in program!

3rd Thing: Compilation option

- **$RSFSRC/user/your-directory**
  - **Your source codes**
  - **SConstruct script**
  - **scons & scons install (Compilation and self-documentation)**

- **Build the executables directly from SConstruct in working directory**

Design:

```
from rsf.proj import *
argv[])
# Program compilation
proj = Project()
exe_clip = proj.Program('clip_c.c')
```

Result:

```
gcc -o clip_c.o -c -O2 -x c -std=gnu99 -Wall -pedantic -
DNO_BLAS -I/Users/zhiguang/RSFROOT/include clip_c.c
gcc -o clip_c.exe -framework Accelerate clip_c.o -L/Users/
zhiguang/RSFROOT/lib -lrsf -lm -lmx
```

For convenience, we use the second option for compilation!

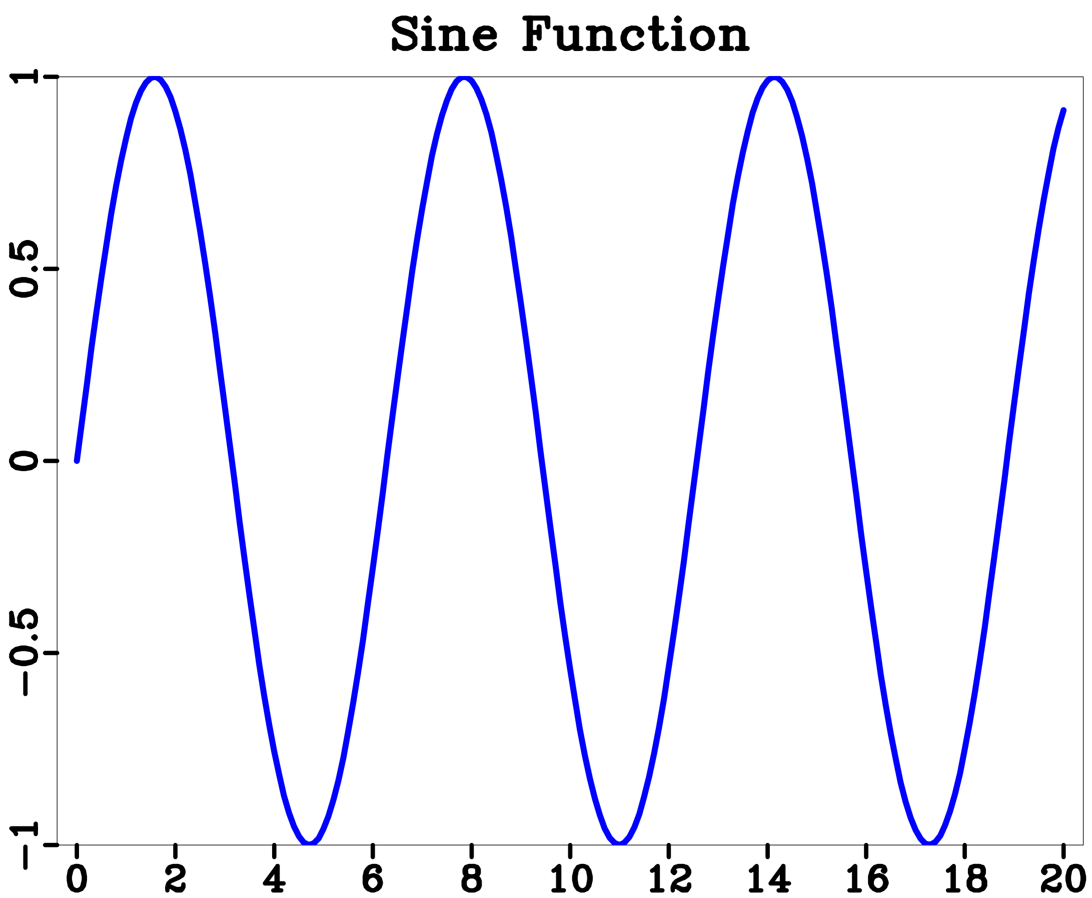
Outline

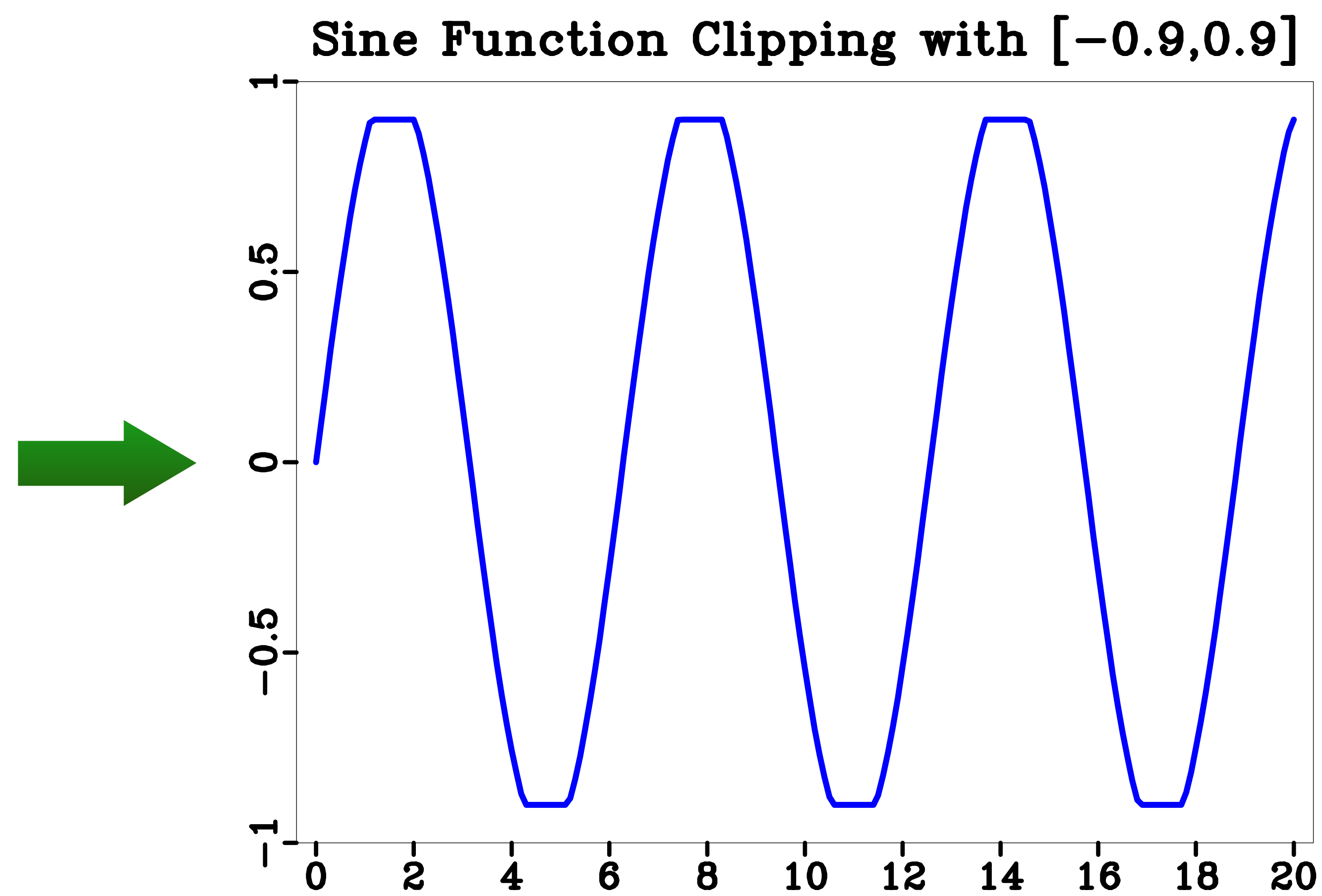
— A quick tour of APIs

- Preparations
- Two examples
- Madagascar API
  - C
  - C++
  - F90
- Exercise (Born modeling)

First example: Data clipping

$$f(x) = \begin{cases} upper, & if\ f(x) > upper \\ unchanged, & others \\ lower, & if\ f(x) < lower \end{cases}$$





Second example: Wave propagation

Acoustic wave equation:

$$\left(\frac{1}{v^2(x)}\frac{\partial^2}{\partial t^2} - \nabla^2\right)u(x,t) = f(x,t)$$

Discretization form:

Scaled Laplacian term

Scaled source term

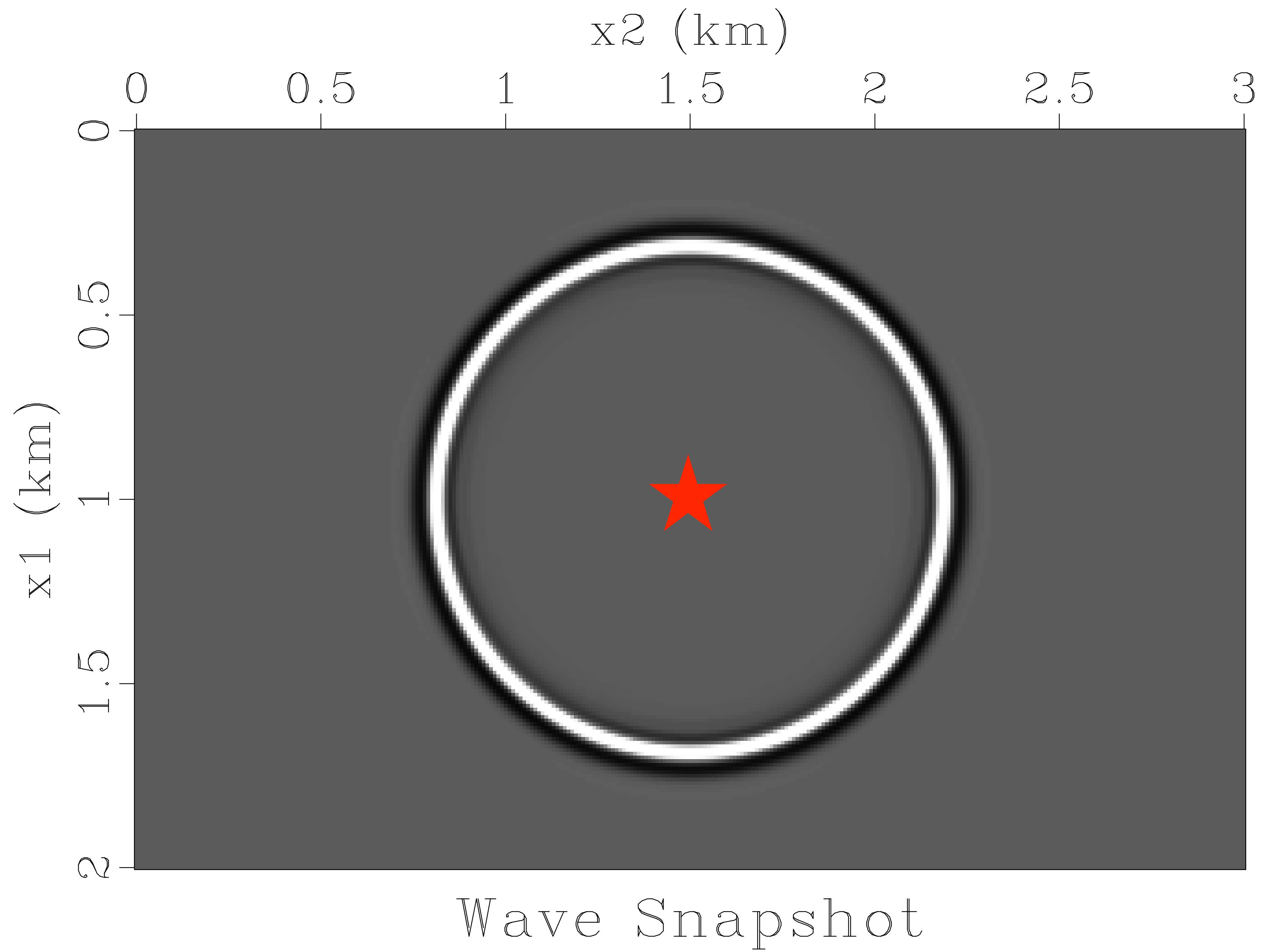
$$u(x,t_{i+1}) = \nabla^2 u(x,t_i)\,\Delta t^2 v^2(x) + f(x,t)\,\Delta t^2 v^2(x) + 2u(x,t_i) - u(x,t_{i-1})$$

Next time step

Current time step

Previous time step

Second example: Wave propagation



Outline

— A quick tour of APIs

- ✦ Preparations
- ✦ Two examples
- ✦ **Madagascar API**
  - C
  - C++
  - F90
- ✦ Exercise (Born modeling)

Commonly used API functions

1. Set up input/output files
2. Read arguments from file header/command-line
3. Set up the axes of output files
4. Read/write data from/into files
5. Other functions
   - Get data type
   - Get file dimension
   - Allocate storage
   - … …

Learning approach

Analyze first example

↓

Extract all API functions and explain more details

↓

Review the API functions through second example

↓

Exercise: Apply the APIs into Born modeling program

Outline

— A quick tour of APIs

- ✦ Preparations
- ✦ Two examples
- ✦ Madagascar API
  - C
  - C++
  - F90
- ✦ Exercise (Born modeling)

C code of first example

example_zhiguang/C/clip_c.c

```c
/* Clip the data. */

#include <rsf.h>
#include <float.h>

int main(int argc, char* argv[])
{
    int n1, n2, i1, i2;
    float upper, lower;
    float *trace;
    /* Input and output files */
    sf_file in, out;

    /* Initialize RSF */
    sf_init(argc,argv);
    /* standard input */
    in  = sf_input("in");
    /* standard output */
    out = sf_output("out");

    /* check that the input is float */
    if (SF_FLOAT != sf_gettype(in))
    sf_error("Need float input");

    /* n1 is the fastest dimension (trace length) */
    if (!sf_histint(in,"n1",&n1))
    sf_error("No n1= in input");
```

sf_input()

sf_output()

sf_gettype()

sf_histint()

C code of first example

```c
    /* leftsize gets n2*n3*n4*... (the number of traces) */
    n2 = sf_leftsize(in,1);

    /* parameter from the command line (i.e. upper=1.) */
    if (!sf_getfloat("upper", &upper)) upper=+FLT_MAX;
    if (!sf_getfloat("lower", &lower)) lower=-FLT_MAX;

    /* allocate floating point array */
    trace = sf_floatalloc (n1);

    /* loop over traces */
    for (i2=0; i2 < n2; i2++) {

    /* read a trace */
    sf_floatread(trace,n1,in);

    /* loop over samples */
    for (i1=0; i1 < n1; i1++) {
        if      (trace[i1] > upper) trace[i1]= upper;
        else if (trace[i1] < lower) trace[i1]= lower;
    }

    /* write a trace */
    sf_floatwrite(trace,n1,out);
    }

    exit(0);
}
```

sf_leftsize()

sf_getfloat()

sf_floatalloc()

sf_floatread()

sf_floatwrite()

Commonly used C API functions

1. Set up input/output files

- sf_file in, vel, out;
- in=sf_input("in");   // standard input
- out=sf_output("out");   // standard output
- vel=sf_input("vel");   // other input file
- < data.rsf sfcommand vel=velocity.rsf > image.rsf

2. Read arguments from file header/command-line

- float upper; int n;   float interval;
- sf_getfloat("upper", &upper);   sf_getint("number", &n);   // command-line
- sf_histfloat(in, "d1", &interval); // parameter from header of input file
- < data.rsf sfcommand upper=1. number=10 > image.rsf

Commonly used C API functions

3. Set up the axes of output files

    - sf_file out;
    - sf_putint(out, “n3”, 100);
    - sf_putfloat(out, “d3”, 1.0);
    - sf_putfloat(out, “o3”, 0.0);
    - sf_putstring(out, “label3”, “Time”);
    - sf_putstring(out, “unit3”, “s”);

4. Read/write data from/into files

    - sf_floatread (array[0], n1*n2, datafile); // float array[n2][n1]
    - sf_intread (index[0][0], n1*n2*n3, indexfile); // int index[n3][n2][n1]
    - sf_floatwrite (array[0], n1*n2, outputfile);
    - sf_intwrite (index[0][0], n1*n2*n3, output_index_file);

Commonly used C API functions

5. Other functions

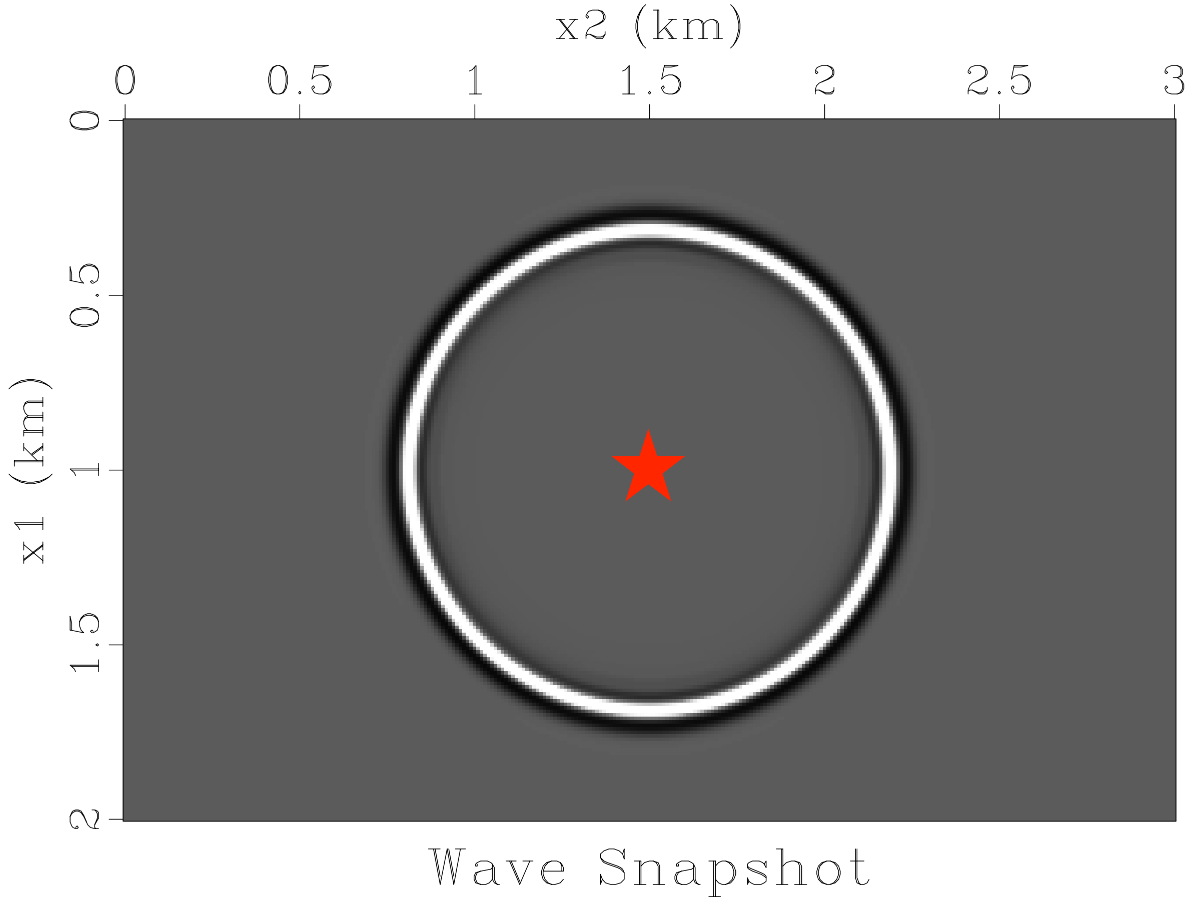
- float **array;
- array=sf_floatalloc2(n1, n2); // 2-D float-type array
- int ***index;
- index=sf_intalloc3(n1, n2, n3); // 3-D integer-type array

- sf_file in;
- if(SF_FLOAT != sf_gettype(in)) sf_error("Need float input");

- sf_file in; // n1*n2*n3*n4
- int number_of_trace;
- number_of_trace=sf_leftsize(in, 1);

C: SConstruct and second example

1. Open [example_zhiguang/C/SConstruct]
2. Run *scons view*
3. Review the API functions via [example_zhiguang/C/wave_c.c]



Outline

— A quick tour of APIs

- ✦ Preparations
- ✦ Two examples
- ✦ Madagascar API
  - C
  - C++
  - F90
- ✦ Exercise (Born modeling)

C++ code of first example

```
// Clip the data.

#include <rsf.hh>
#include <float.h>
#include <valarray>

int main(int argc, char* argv[])
{
    // trace length, number of traces
    int n1, n2;
    float upper, lower;

    // Initialize RSF
    sf_init(argc,argv);

    // input parameter, file
    iRSF par(0), in;
    // output file
    oRSF out;

    // check that the input is float
    if (SF_FLOAT != in.type())
    sf_error("Need float input");

    // n1 is the fastest dimension
    in.get("n1",n1);

    // leftsize gets n2*n3*n4*...
    n2=in.size(1);
```

example_zhiguang/C++/clip_cc.cc

The big difference:

Files are defined as classes, and can be manipulated by member functions!

iRSF par(0), in;

oRSF out;

in.type()

in.get()

in.size()

C++ code of first example

```
    // parameter from the command line
    par.get("upper",upper,+FLT_MAX);
    par.get("lower",lower,-FLT_MAX);

    // allocate floating point array
    std::valarray<float> trace(n1);

    // loop over traces
    for (int i2=0; i2 < n2; i2++) {

    // read a trace (overloading operator)
    in >> trace;

    // loop over samples
    for (int i1=0; i1 < n1; i1++) {
        if      (trace[i1] > upper) trace[i1]=upper;
        else if (trace[i1] < lower) trace[i1]=lower;
    }

    // write a trace
    out << trace;
    }

    exit(0);
}
```

par.get()

in >> array;

out << array;

Commonly used C++ API functions

1. Set up input/output files
   - iRSF in; // standard input
   - iRSF vel(“vel”); // vel=velocity.rsf
   - oRSF out; // standard output
   - oRSF snapshot(“wfld”); // wlfd=snap.rsf
   - < data.rsf sfcommand vel=velocity.rsf wfld=snap.rsf > image.rsf

2. Read arguments from file header/command-line
   - iRSF in; // standard input
   - iRSF par(0); // command-line input
   - int n1; in.get(“n1”, n1); float d1; in.get(“d1”, d1, 0.01);
   - int num; par.get(“number”, num); float clip; par.get(“clip”, clip, 1.);
   - < data.rsf sfcommand number=10 clip=3. > image.rsf

Commonly used C++ API functions

3. Set up the axes of output files

  - oRSF Fout;
  - Fout.put(“n1”, nt);
  - Fout.put(“d1”, dt);
  - Fout.put(“o1”, t0);
  - Fout.put(“label1”, “Time”);
  - Fout.put(“unit1”, “s”);

4. Read/write data from/into files

  - iRSF Fin; oRSF Fout;
  - valarray<float> array(n);
  - Fin >> array;
  - Fout << array;

Commonly used C++ API functions

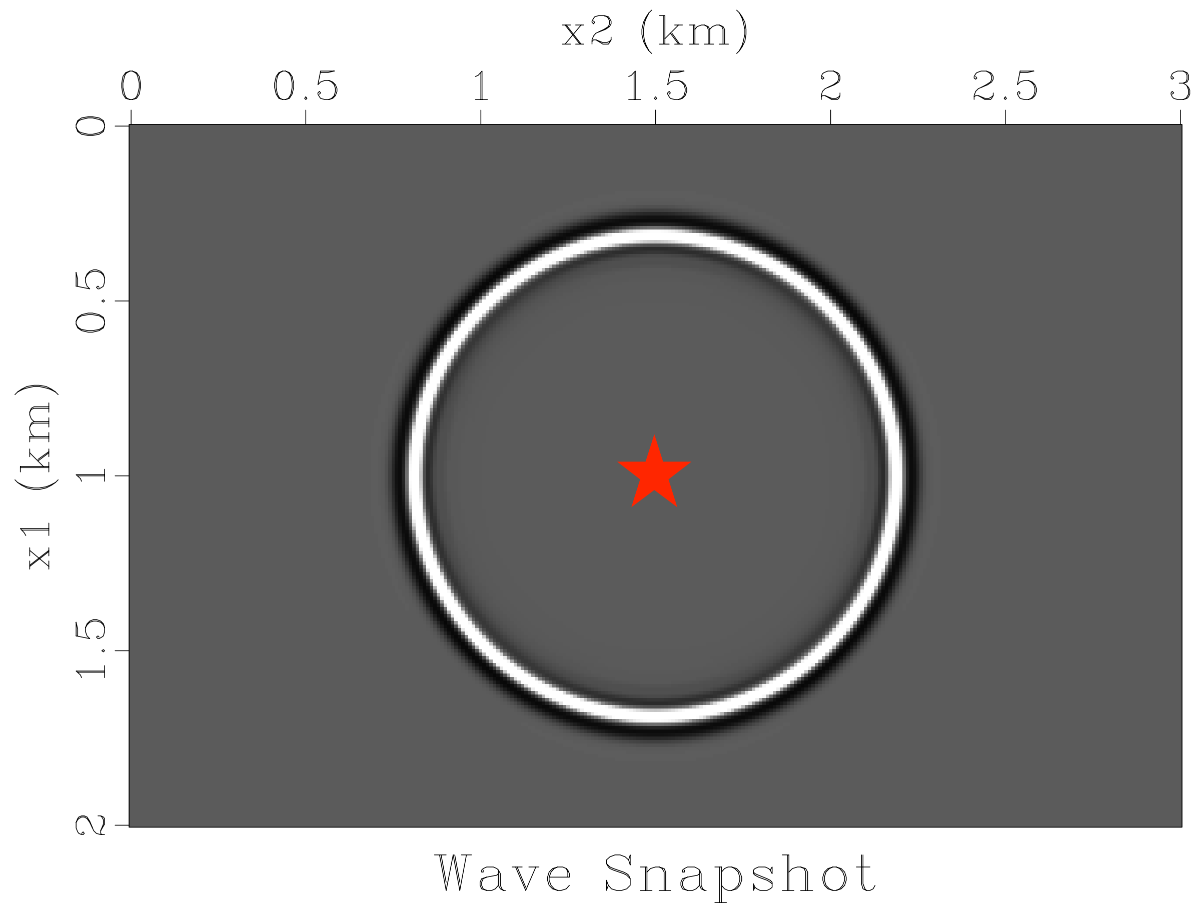
5. Other functions

- iRSF Fin;
- if(SF_FLOAT != Fin.type( )) sf_error(“Need float input”);

- iRSF Fin; // n1*n2*n3*n4
- int number_of_trace;
- number_of_trace=Fin.size(1); // n2*n3*n4

C++: SConstruct and second example

1. Open example_zhiguang/C++/SConstruct
2. Run *scons view*
3. Review the API functions via
   example_zhiguang/C++/wave_cc.cc



Outline

— A quick tour of APIs

- ✦ Preparations
- ✦ Two examples
- ✦ Madagascar API
  - C
  - C++
  - F90
- ✦ Exercise (Born modeling)

F90 code of first example

example_zhiguang/F90/clip_f90.f90

```fortran
! Clip the data.
program Clip
  use rsf ! use module


  implicit none
  type (file)                        :: in, out
  integer                            :: n1, n2, i1, i2
  real                               :: upper, lower
  real, dimension (:), allocatable :: trace

  ! initialize RSF
  call sf_init()
  ! Standard input file (void)
  in = rsf_input()
  ! Standard output file (void)
  out = rsf_output()

  ! check that the input is float
  if (sf_float /= gettype(in)) call sf_error("Need floats")

  ! n1 is the fastest dimension
  call from_par(in,"n1",n1)
```

The big difference: RSF module is used.

rsf_input()

rsf_output()

gettype(in)

from_par

F90 code of first example

```
  ! leftsize gets n2*n3*n4*...
  n2 = filesize(in,1)

  ! parameter form the command line
  call from_par("upper",upper,10000.)
  call from_par("lower",lower,-10000.)

  ! allocate floating point array
  allocate (trace (n1))


  ! loop over traces
  do i2=1, n2

       ! read a trace
      call rsf_read(in,trace)

      ! loop over samples
      where (trace > upper) trace = upper
      where (trace < lower) trace = lower

      ! write a trace
      call rsf_write(out,trace)
  end do
end program Clip
```

filesize()

from_par()

rsf_read()

rsf_write()

Commonly used F90 API functions

1. Set up input/output files

- type(file) :: in, out, vel
- in = rsf_input(“in”) / rsf_input()   // standard input
- vel = rsf_input(“vel”)   // vel=velocity.rsf
- out = rsf_output(“out”) / rsf_output()   // standard output
- < data.rsf sfcommand vel=velocity.rsf > image.rsf

2. Read arguments from file header/command-line

- in = rsf_input( )   // standard input
- call from_par(in, “n1”, n1)   // from input file header
- call from_par(“number”, num, default) // command-line input
- < data.rsf sfcommand number=10 > image.rsf

Commonly used F90 API functions

3. Set up the axes of output files
   - Fout = rsf_output( )
   - call to_par(Fout, “n1”, nt)
   - call to_par(Fout, “d1”, dt)
   - call to_par(Fout, “o1”, t0)
   - call to_par(Fout, “label1”, “Time”)
   - call to_par(Fout, “unit1”, “s”)
4. Read/write data from/into files
   - Fin = rsf_input()
   - Fout=rsf_output()
   - real array(100)
   - call rsf_read(Fin, array)
   - call rsf_write(Fout, array)

Commonly used F90 API functions

5. Other functions

- Fin = rsf_input()

- if(SF_FLOAT /= gettype(Fin))   call sf_error(“Need float input”)

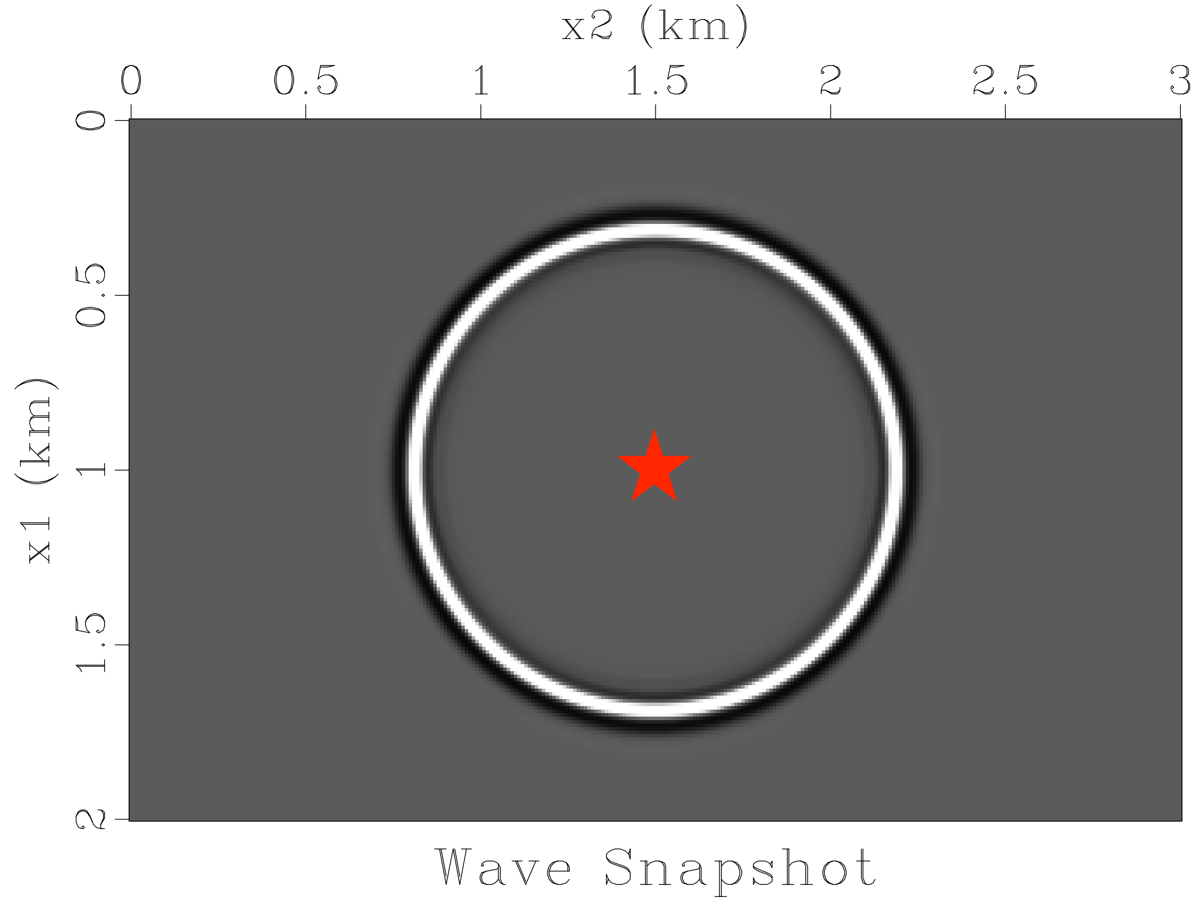
- Fin = rsf_input() // n1*n2*n3*n4

- integer :: number_of_trace

- number_of_trace=filesize(Fin,1) // n2*n3*n4

F90: SConstruct and second example

1. Open [example_zhiguang/F90/SConstruct]
2. Run *scons view*
3. Review the API functions via
   [example_zhiguang/F90/wave_f90.f90]



Outline

— A quick tour of APIs

- Preparations
- Two examples
- Madagascar API
  - C
  - C++
  - F90
- Exercise (Born modeling)

Born approximation

Acoustic wave equation: $$\left(\frac{1}{v^2(x)}\frac{\partial^2}{\partial t^2} - \nabla^2\right)u(x,t) = f(x,t)$$

Scale separation: $$v(x) = v_0(x) + \delta v(x)$$

Correspondingly, $$u(x,t) = u_0(x,t) + \delta u(x,t)$$

Wave equation with background velocity:

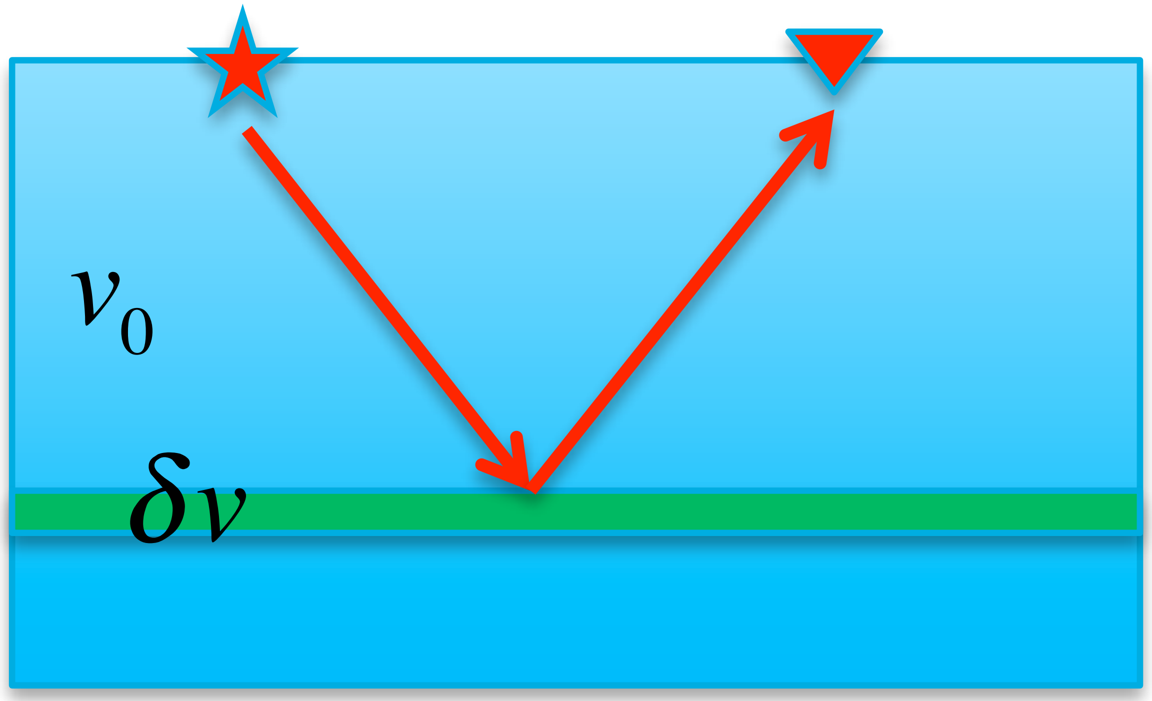
$$\left(\frac{1}{v_0^2(x)}\frac{\partial^2}{\partial t^2} - \nabla^2\right)u_0(x,t) = f(x,t)$$

Born modeling: $$\left(\frac{1}{v_0^2(x)}\frac{\partial^2}{\partial t^2} - \nabla^2\right)\delta u(x,t) = \frac{\partial^2}{\partial t^2}\frac{2\,\delta v\,u_0(x,t)}{v_0^3(x)}$$

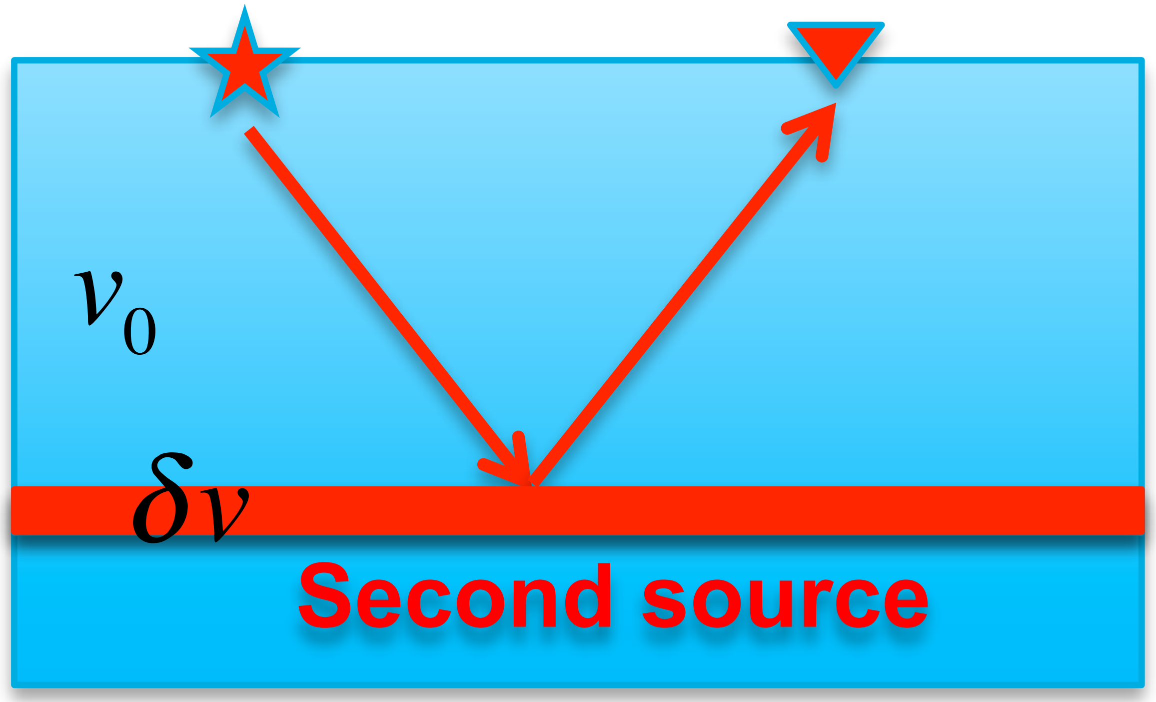
Born approximation

Born modeling:

$$(\frac{1}{v_0^2(x)}\frac{\partial^2}{\partial t^2} - \nabla^2)\delta u(x,t) = \frac{\partial^2}{\partial t^2}\frac{2\,\delta v\, u_0(x,t)}{v_0^3(x)}$$



Forward modeling



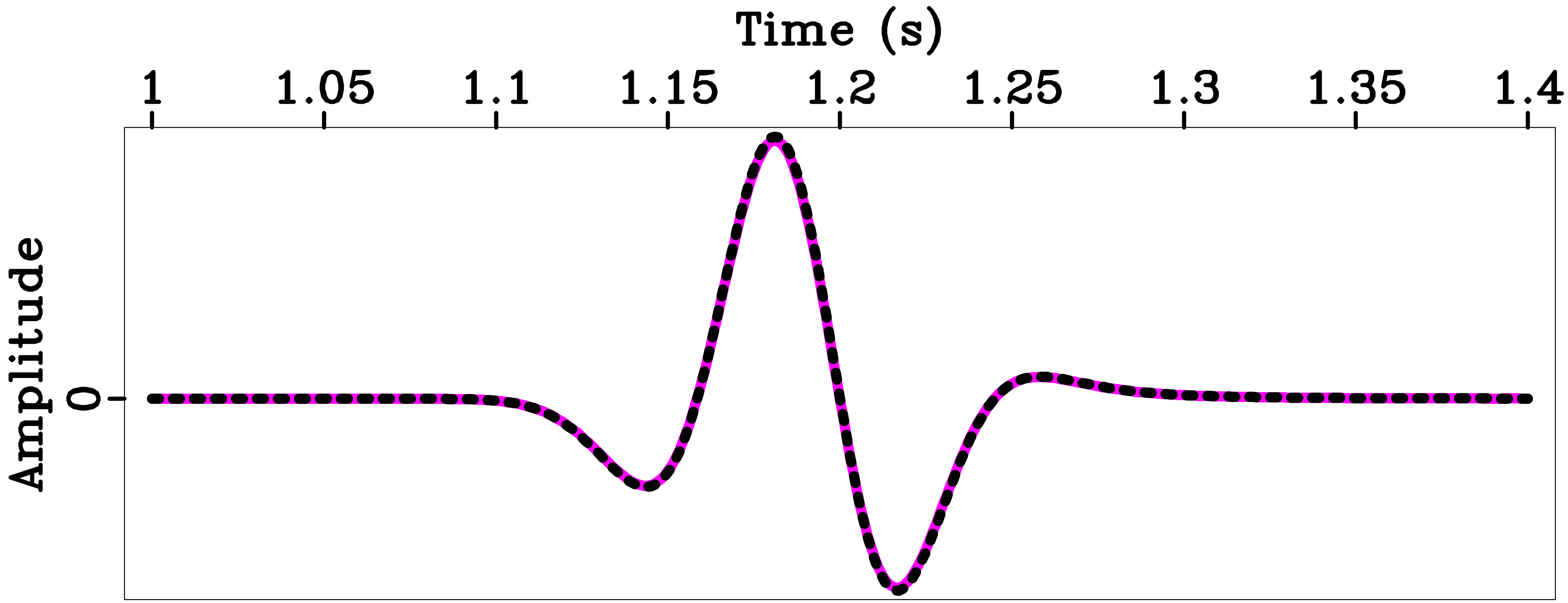
Born modeling

Tasks

1. **Implement Born modeling based on previous wave propagation operators;**

$$\left(\frac{1}{v_0^2(x)}\frac{\partial^2}{\partial t^2}-\nabla^2\right)\delta u(x,t)=\frac{\partial^2}{\partial t^2}\frac{2\,\delta v\,u_0(x,t)}{v_0^3(x)}$$

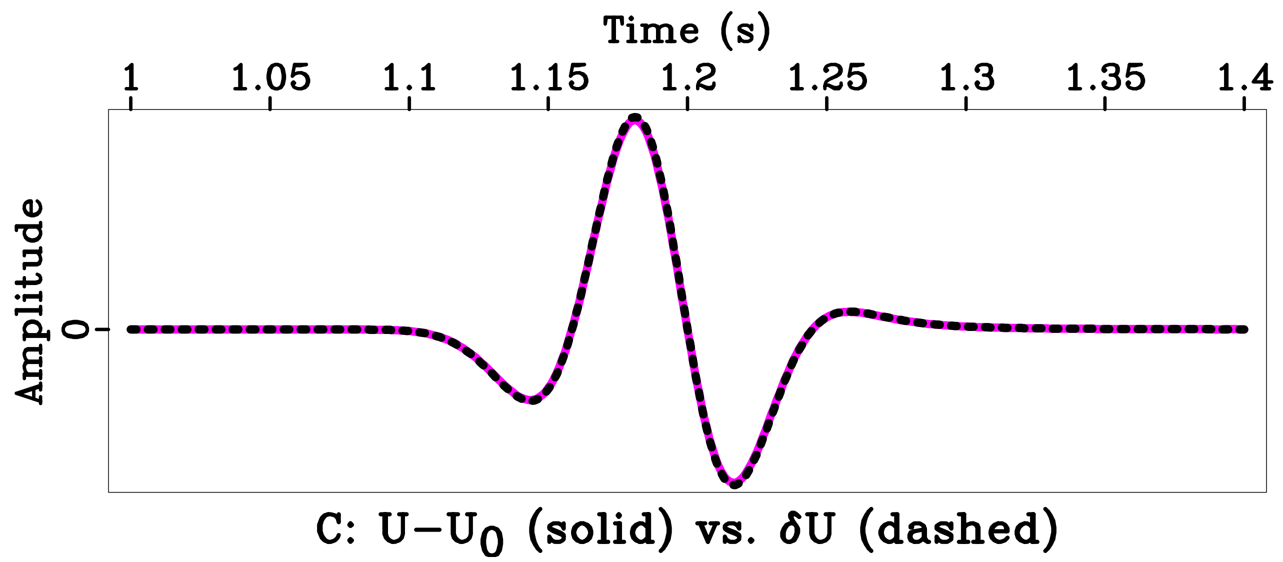
2. **Check the correctness of your Born modeling operator.**



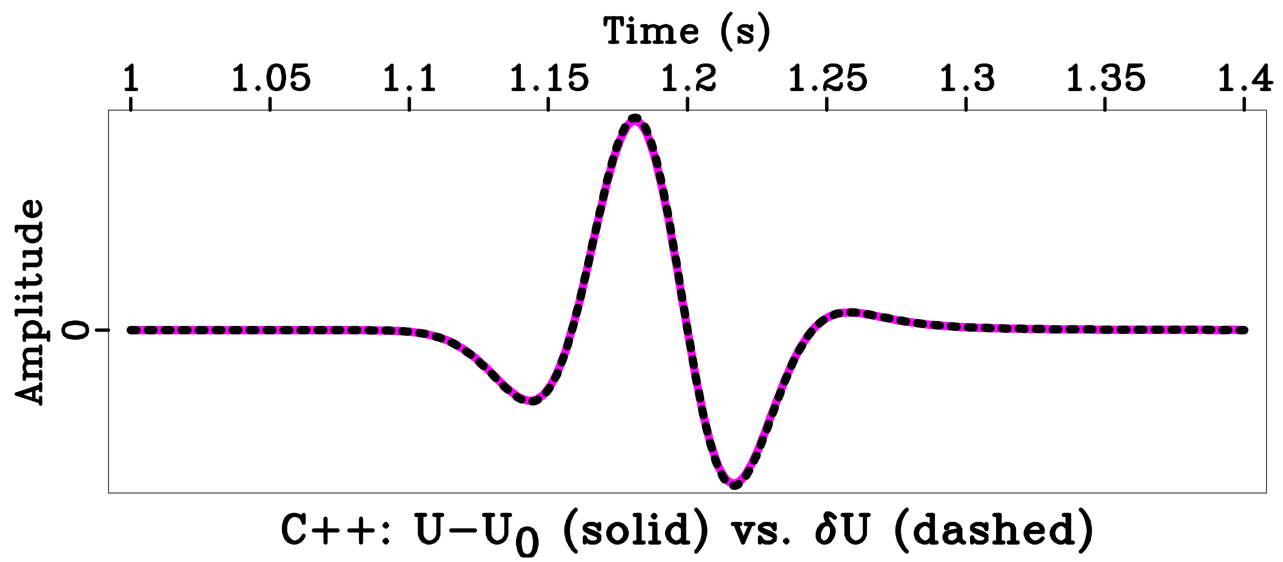
C: $U-U_0$ (solid) vs. δU (dashed)

References (*scons view*)

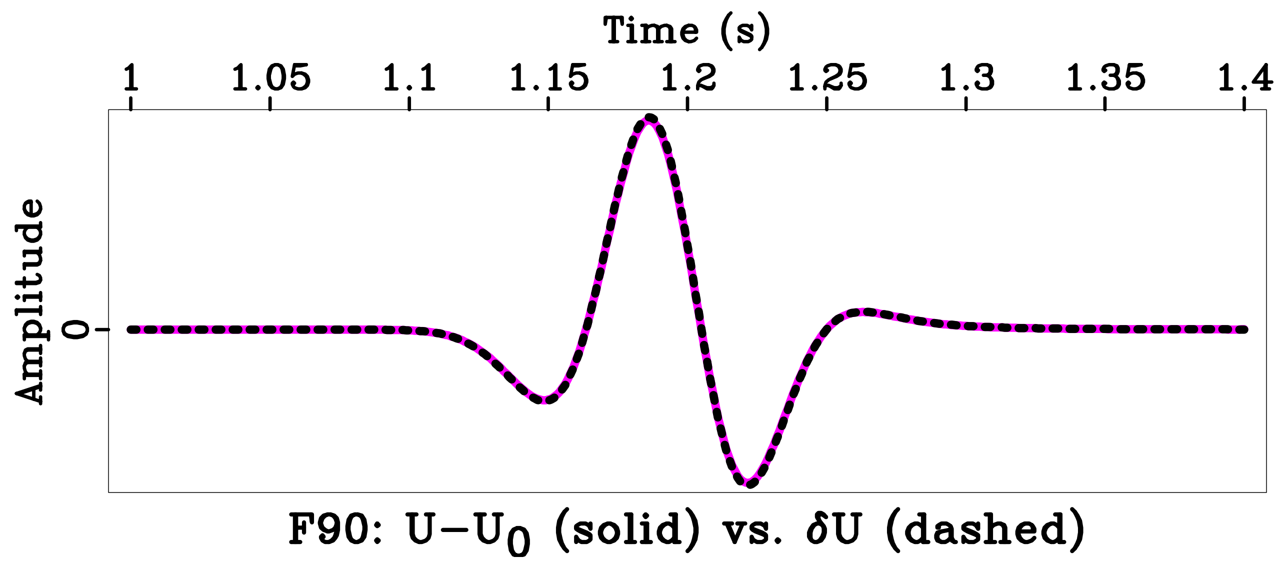
[example_zhiguang/exercise_born/born_c.c]



[example_zhiguang/exercise_born/born_cc.cc]



[example_zhiguang/exercise_born/born_f90.f90]



Thanks for your listening!