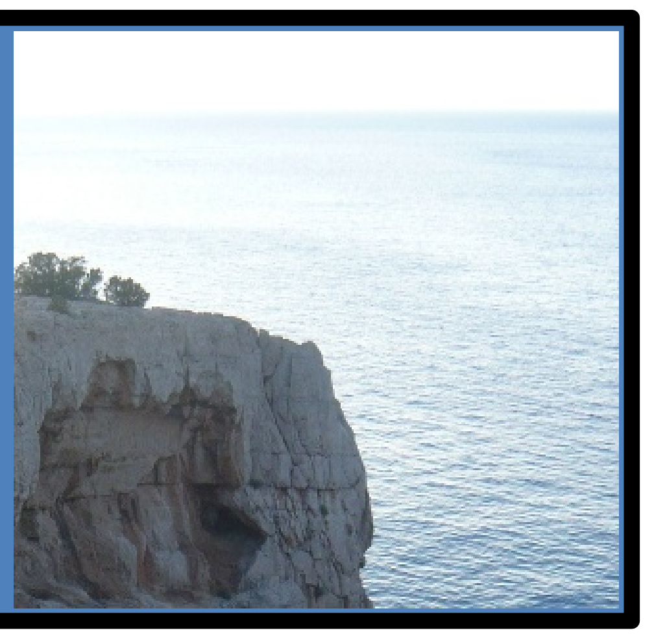
SeaSeis: A simple open-source seismic data processing system

Bjorn Olofsson, SeaBird Exploration



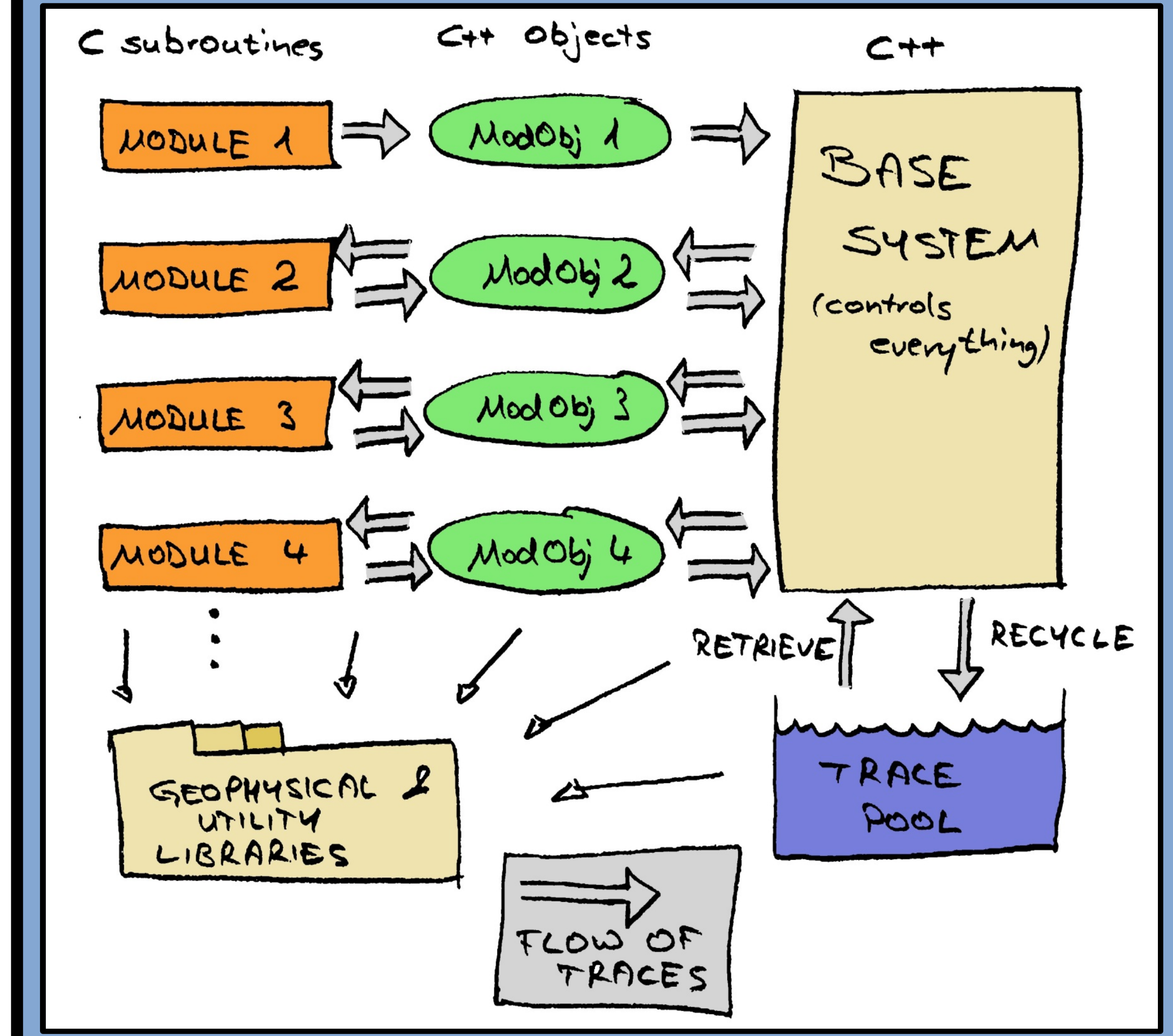
Overview

SeaSeis is a seismic data processing system developed over the past six years, and which has been distributed as open-source software for about three years.

The main objectives when writing the system were:

1) For the normal user: Easy to write processing flows.
2) For the developer: Easy to write new processing modules.
3) For production use: Availability of flow control, logs, and master flow submission.
4) Stability of the base system's source code. This has hopefully been achieved by ample use of the tool *valgrind*.
5) Platform independence: Currently compiles on most Linux/UNIX and Windows platforms.

System design



The base system…:

- Manages tedious tasks, e.g. memory allocation
- Monitors trace flow
- Writes log files = history
- Checks for errors
- Controls versions

To add a new module, only the C subroutine MODULE 1 needs to be written.
The corresponding C++ object ModObj 1 is auto-generated.

Modules

SeaSeis currently has around 80 processing modules, most of them dealing with the logistics of the trace flow and trace header manipulations.
Most are well implemented, but some are mere placeholders with rudimentary functionality only.

Module list

ATTRIBUTE	Extract attribute from data
BEAM_FORMING	Generate beams from receiver array
BIN	Perform binning
CCP	Perform ccp binning
CMP	Perform CMP binning
CONCATENATE	Concatenate adjacent traces
CORRELATION	Cross-correlation between adjacent traces, or auto-correlation of same
trace	
DEBIAS	De-bias input data
DESIGNATURE	Designature filter operation
DESPIKE	Spike/noise burst removal
ELSE	Else statement
ELSEIF	Elseif statement
ENDIF	Endif statement
ENDSPLIT	Endsplit statement
ENS_DEFINE	Define ensemble trace headers
FFT	FFT transform
FFT_2D	2D FFT
FILTER	Frequency filter
GAIN	Apply gain function to trace samples
GEOTOOLS	Various geophysical tools
HDR_DEL	Delete trace headers
HDR_MATH	Trace header computation
HDR_MATH_ENS	Multi-trace header computation
HDR_PRINT	Print trace header values
HDR_SET	Set trace header from table
HISTOGRAM	Histogram
HODOGRAM	Hodogram analysis
IF	If statement
IMAGE	Create image file from seismic display
INPUT	Input SeaSeis data
INPUT_ASCII	Input ASCII file
INPUT_CREATE	Create traces
INPUT_MSEED	Input Mini SEED file
INPUT_SEGD	Input SEGD data
INPUT_SEGY	Input SEGY/SU data
INPUT_SINEWAVE	Create traces with sine waves
KILL	Kill traces
KILL_ENS	Kill ensembles
LMO	Linear moveout correction
MIRROR	Perform mirror image binning
MUTE	Mute trace data
NMO	Perform normal moveout correction
OFF2ANGLE	Offset to angle transform
ORIENT	Solve sensor orientation
ORIENT_CONVERT	Convert sensor orientation parameters
OUTPUT	Output SeaSeis data
OUTPUT_SEGY	Output SEGY/SU data
OVERLAP	Create data overlap between adjacent traces
PICKING	Pick first breaks or other event
POSCALC	Compute receiver position from time picks
PZ_SUM	PZ combination
RAY2D	2D isotropic ray tracer
READ_ASCII	Read trace header values from ASCII file
REPEAT	Repeat/duplicate traces
RESAMPLE	Resample trace to different sample interval
RESEQUENCE	Resequence trace header
RMS	Compute RMS value in given time window
ROTATE	Perform 2D/3D rotation to input traces
SCALING	Scale trace data with linear function
SELECT	Select traces
SELECT_TIME	Select time of interest
SEMBLANCE	Semblance panel generation
SORT	Sort traces
SPLIT	Split/branch trace flow
SPLITTING	Shear-wave splitting analysis
STACK	Ensemble stack
STATICS	Apply trace statics
SUMODULE	Generic wrapper for Seismic Unix (SU) module
TEST	Demonstration single trace module
TEST_MULTI_ENSEMBLE	Test module – multi-trace, ensemble module
TEST_MULTI_FIXED	Test module – multi-trace, fixed number of input traces
TIME_SLICE	Extract time slice(s) from input data, write to trace header field or output data
TIME_STRETCH	Stretch/squeeze data trace
TRC_ADD_ENS	Ensemble trace adding
TRC_INTERPOL	Interpolate traces
TRC_MATH	Trace sample computation
TRC_MATH_ENS	Multi-trace sample computation
TRC_PRINT	Print trace samples
TRC_SPLIT	Trace split
XSCRATCH	Test module

2D Seismic viewer

The ***SeaView*** 2D seismic viewer is a prototype illustrating the underlying generic seismic display engine CSeisLib, written in Java.

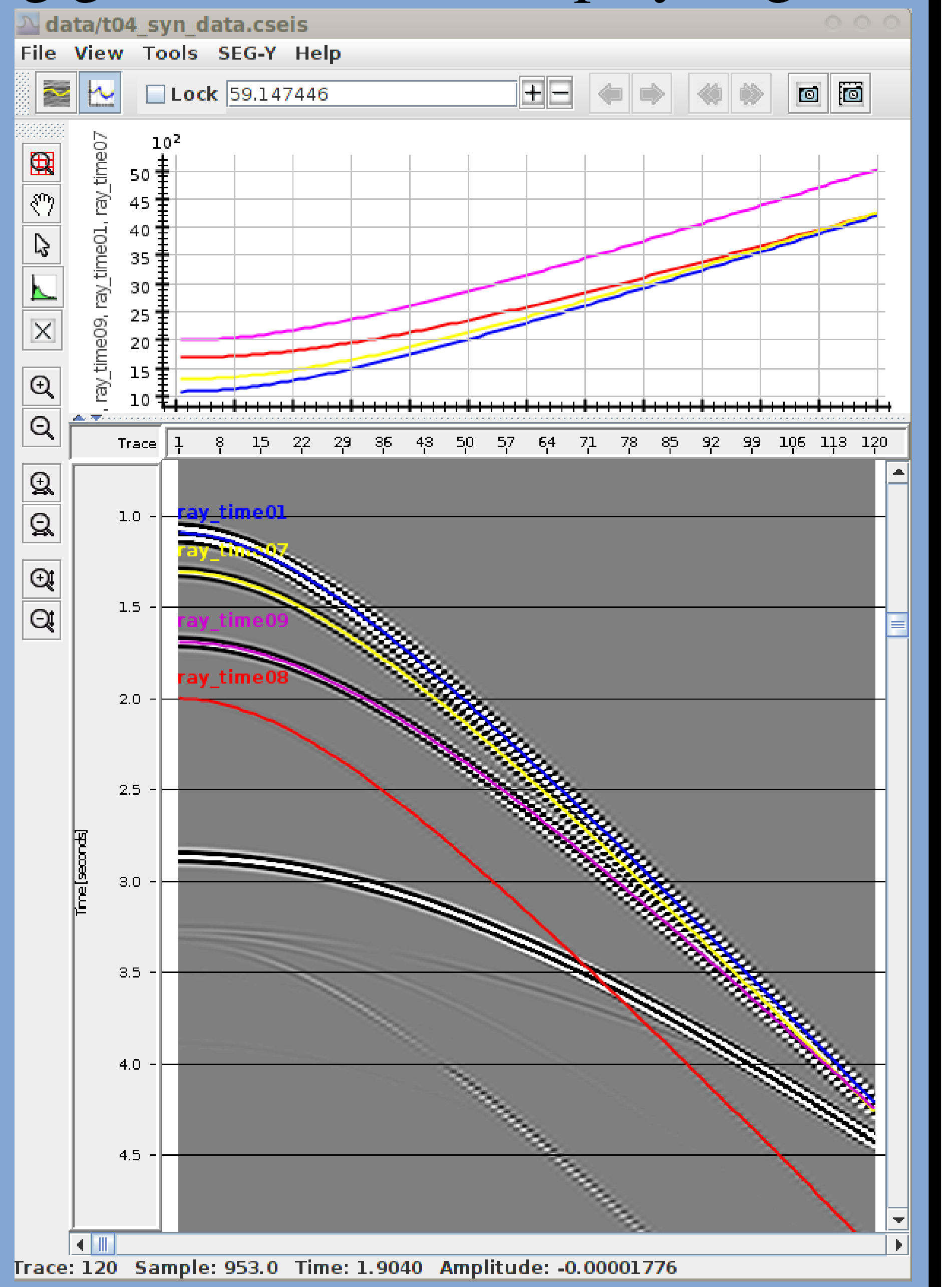
Formats:

- CSEIS
- SEGY *(hdr map!)*
- SEGD *(nsamp!)*
- SU *(endian!)*
- ASCII file format

Input data size: Any
Display size: Any *(until out of memory)*

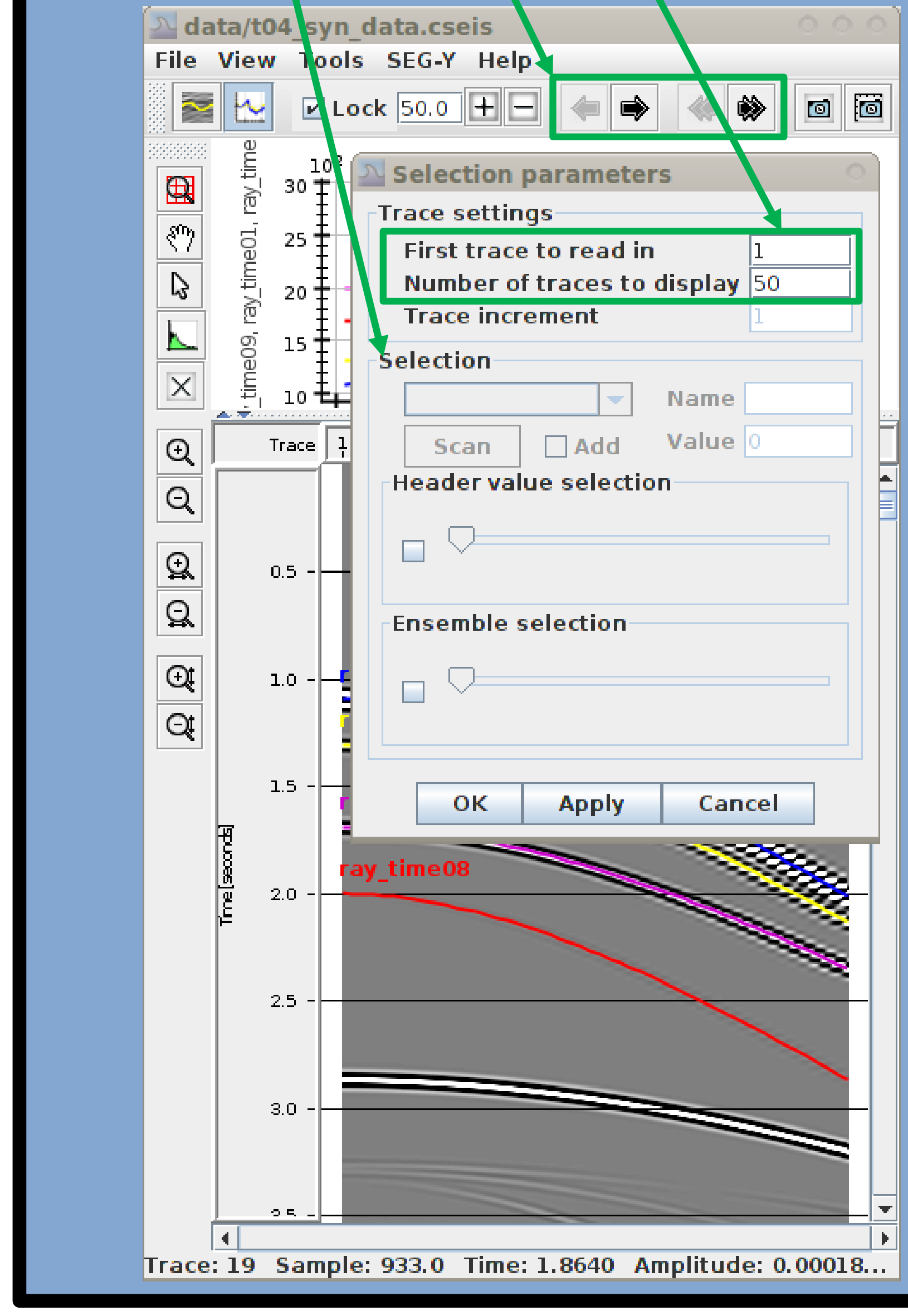
Navigation:

- Click forward/ backward buttons
- Specify trace #
- …other



Lock 59.147446

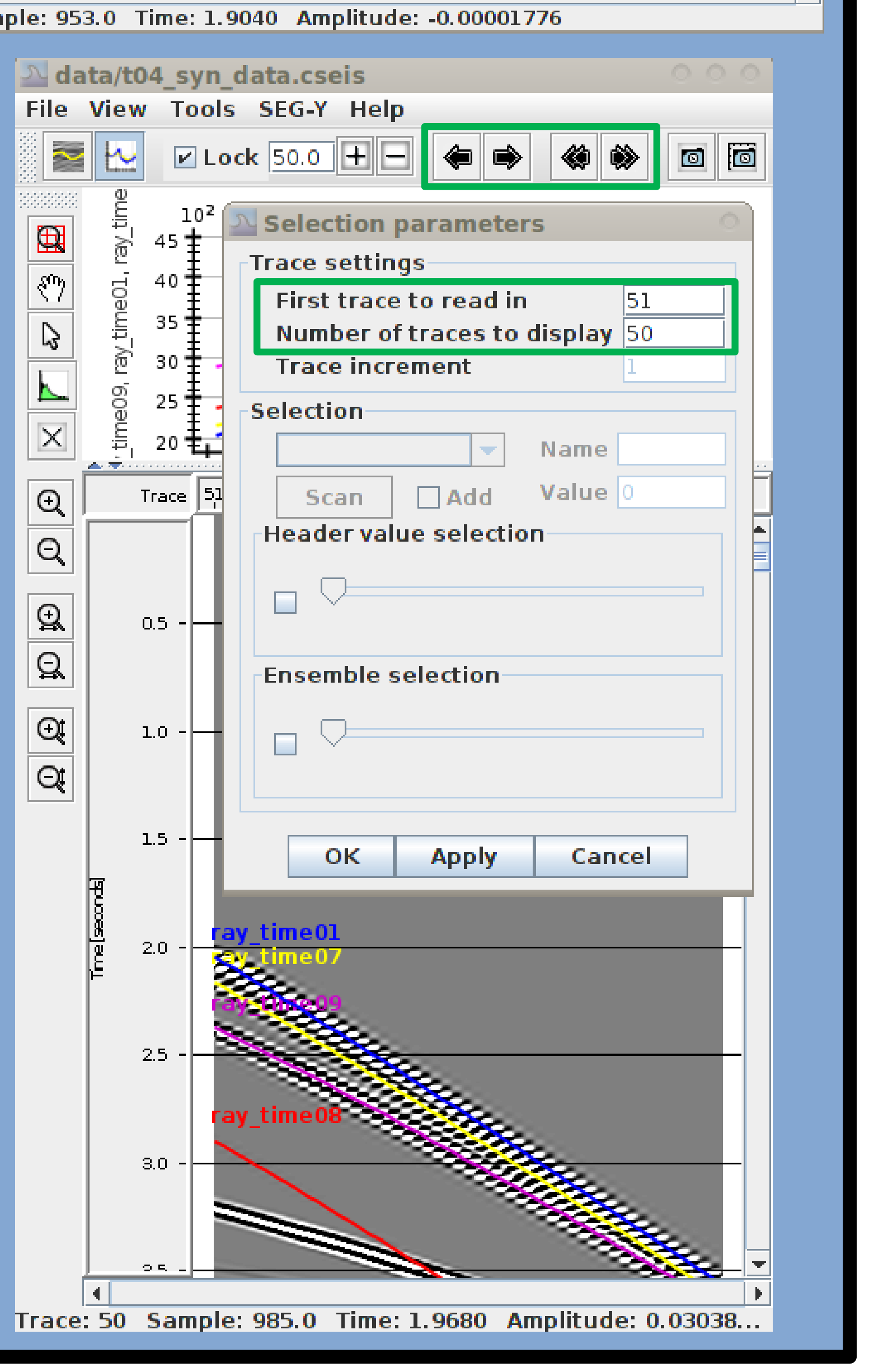
Trace: 120 Sample: 953.0 Time: 1.9040 Amplitude: -0.00001776



☑ Lock 50.0

First trace to read in 1
Number of traces to display 50
Trace increment 1

Trace: 19 Sample: 933.0 Time: 1.8640 Amplitude: 0.00018…



☑ Lock 50.0

First trace to read in 51
Number of traces to display 50
Trace increment 1

Trace: 50 Sample: 985.0 Time: 1.9680 Amplitude: 0.03038…

Example flow

```
####################################################
# Demo job flow

&define year 2002
&define seq  834
&define datadir  /disk/projX/data/

$INPUT_SEGD
 filename   &datadir&/raw_data_seq&seq&.segd

# Kill auxiliary channels
$KILL
 header     trc_type
 select     !1

# Read in navigation headers
$READ_ASCII
 filename    &datadir&/source_seq&seq&.S
 method      positions
 key_sps_time time_samp1 72 9
 time_year   &year&
 header      sail_line   6 2
 header      seq         9 3
 header      source     22 4
 header      sou_z      33 4
 header      sou_x      48 8
 header      sou_y      57 9

# Remove trace mean
$DEBIAS
 mode trace

# Compute source-receiver offset, set cable number
$HDR_MATH
 new       cblno  int
 equation  offset "sqrt( pow(sou_x-rec_x,2) + pow(sou_y-rec_y,2) )"
 equation  cblno  "int( (chan-1)/120 ) + 1"

# Resequence trace number
$ENS_DEFINE
 header    cblno

$RESEQUENCE
 header    trcno
 set       1  1  0
 mode      ensemble

# Output near traces
$IF
 header   chan
 select   1,121,241,361

 $OUTPUT
  filename  &datadir&/near_traces_seq&seq&.cseis

$ENDIF

$RMS
 start     0
 end       500
 hdr_rms   rms1

$HDR_PRINT
 filename  &datadir&/hdr_dump.seq&seq&
 header    rcv chan seq source rec_x  rec_y  sou_x  sou_y  offset rms1
 format    %10d %5d %5d %5d    %11.2f %11.2f %11.2f %11.2f %8.2f  %13.5e

# Output to SEGY file
$OUTPUT_SEGY
  filename    &datadir&/data_seq&seq&.segy
  charhdr     "C 1 SEQUENCE: &seq&"
  charhdr     "C 2 ..."
```

Single trace module

```
#include "cseis_includes.h"

using namespace cseis_system;
using namespace std;

namespace mod_test4 {
  struct VariableStruct {
    int hdrId;
  };
}
using mod_test4::VariableStruct;

//**********************************************************************
// Init phase
//
void init_mod_test4_( csParamManager* param,
                      csInitPhaseEnv* env, csLogWriter* log )
{
  csExecPhaseDef*   edef = env->execPhaseDef;
  VariableStruct* vars   = new VariableStruct();
  edef->setVariables( vars );
  edef->setExecType( EXEC_TYPE_SINGLETRACE );

  string text;
  param->getString("header", &text);
  if( !env->headerDef->headerExists(text) ) {
    log->error("Trace header does not exist: %s", text.c_str());
  }
  vars->hdrId = env->headerDef->headerIndex(text);
}

//**********************************************************************
// Exec phase
//
bool exec_mod_test4_( csTrace* trace, int* port,
                      csExecPhaseEnv* env, csLogWriter* log )
{
  VariableStruct* vars =
    reinterpret_cast<VariableStruct*>(env->execPhaseDef->variables());

  if( env->execPhaseDef->isCleanup()){
    delete vars; vars = NULL;
    return true;
  }

  float value = trace->getTraceHeader()->floatValue( vars->hdrId );
  float* samples = trace->getTraceSamples();
  for( int isamp = 0; isamp < env->superHeader->numSamples; isamp++ ) {
    samples[isamp] *= value;
  }

  return true;
}

//**********************************************************************
// Parameter definition
//
void params_mod_test4_( csParamDef* pdef ) {

  pdef->setModule( "TEST4", "Scale data by trace header" );

  pdef->addParam( "header", "Trace header", NUM_VALUES_FIXED );
  pdef->addValue( "", VALTYPE_STRING, "Trace header name" );

}
```

Single trace

Module-specific code

Multi trace module

```
#include "cseis_includes.h"

using namespace cseis_system;
using namespace std;

namespace mod_test5 {
  struct VariableStruct {
    int hdrId;
  };
}
using mod_test5::VariableStruct;

//**********************************************************************
// Init phase
//
void init_mod_test5_( csParamManager* param,
                      csInitPhaseEnv* env, csLogWriter* log )
{
  csExecPhaseDef*   edef = env->execPhaseDef;
  VariableStruct* vars   = new VariableStruct();
  edef->setVariables( vars );
  edef->setExecType( EXEC_TYPE_MULTITRACE );
  edef->setTraceSelectionMode( TRCMODE_ENSEMBLE );

  string text;
  param->getString("header", &text);
  if( !env->headerDef->headerExists(text) ) {
    log->error("Trace header does not exist: %s", text.c_str());
  }
  vars->hdrId = env->headerDef->headerIndex(text);

}

//**********************************************************************
// Exec phase
//
void exec_mod_test5_( csTraceGather* traceGather, int* port,
   int* numTrcToKeep, csExecPhaseEnv* env, csLogWriter* log )
{
  VariableStruct* vars =
    reinterpret_cast<VariableStruct*>(env->execPhaseDef->variables());

  if( env->execPhaseDef->isCleanup()){
    delete vars; vars = NULL;
    return true;
  }
  for( int itrc = 0; itrc < traceGather->numTraces(); itrc++ ) {
    csTrace* trace = traceGather->trace(itrc);
    trace->getTraceHeader()->setIntValue( vars->hdrId, itrc+1 );
  }
  // Remove traces: traceGather->freeTraces(a,b);
  // Add traces:    traceGather->createTraces(a,n,hdef,ns);
}

//**********************************************************************
// Parameter definition
//
void params_mod_test5_( csParamDef* pdef ) {

  pdef->setModule( "TEST5", "Resequence trace header within gather" );

  pdef->addParam( "header", "Trace header", NUM_VALUES_FIXED );
  pdef->addValue( "", VALTYPE_STRING, "Trace header name" );

}
```

Ensemble