Tutorial to Jumpstart SAGA

Dave Ensberg
Yong Han Goh
Marine Physical Laboratory
Scripps Institution of Oceanography
University of California, San Diego
Email : yhgoh@ucsd.edu

Table of contents

1. Introduction

This tutorial gives a quick introduction to SAGA to jumpstart the new user on SAGA, but is not meant to replace the SAGA user manual. This tutorial is broken up into three parts. The first part provides an understanding of the SAGA parameter file. The second part discusses how to input receiver information (array shape) into the forward model SNAP (or SNAP3D) and a quick summary of the options available to do this. The third part explains the actual running of SAGA and how to plot results.

The annex lists 7 jumpstart examples which can be downloaded and run with SAGA. The first two examples are single frequency examples for a simple and fast introduction into running SAGA. The next three examples demonstrate MFP using 13 frequencies with a vertical line array (VLA). The last 2 examples demonstrate MFP with a horizontal line array (HLA).

Single frequency examples:

- SAGA.ex1.VLA.sim - invert for tilt and range using 388 Hz (simulation)

- SAGA.ex2.VLA - invert for tilt and range using 388 Hz (real data from SWellEx96 Event S5)

Multi-frequency examples:

- SAGA.ex3.VLA.sim - invert for source range and depth (MFP) (simulation)

- SAGA.ex4.VLA.kraken - invert for source range and depth (MFP) (simulation) data generated by Kraken, field, and krakp and are imported to SAGA for processing

- SAGA.ex5.VLA.real - invert for source range and depth (MFP) real data from SWellEx96 Event S5.

- SAGA.ex6.HLA.sim- invert for source range, depth, array bearing and array bow (MFP) (simulation)

- SAGA.ex7.HLA.real- invert for source range, depth, array bearing and array bow (MFP) (real data from SWellEx96 Event S5)

2. Setup

1. Download the saga_jumpstart.tar and saga_jumpstart.pdf files to your SAGA directory (e.g. /home/saga53).

2. Extract the saga_jumpstart.tar file
   > tar –xvf saga_jumpstart.tar

3. Go through the tutorial and try the examples!

3. SAGA jumpstart Part I – setting up the parameter file

Refer also to SAGA manual Section 6.

There are two parts to running SAGA. Part 1 is setting up the parameter file which is the input file for SAGA. Part 2 is the actual running of SAGA.

In the SAGA parameter file there are 5 sections, blocks 1 - 3 and 5 are SAGA specific commands. The 4th block of commands is specific to the forward model that will be used in the SAGA run.

A sample SAGA parameter file:

```
01)  SAGA jump start example, data based on physical array (opt X)
02)  c b W                  ! options
03)
04)   2000  64  20          ! niter, q, npop
05)   0.8 0.5 0.05  0       ! px,pu,pm
06)  t                      ! snap options
07)   2 100                 ! freq  max_no_modes
08)  49 388
09)   213.,0.,0.,0          ! water depth scatt(1),scatt(2),beta(0)
10)       0.       1521.94
11)       10.1176   1518.77
12)       19.7780   1499.80
13)       30.2703   1495.00
14)       60.5963   1490.78
15)      213.0000   1488.26
16)   30 1.76 0.2           ! sediment thickness, r1, beta(1)
17)   0.0 1572.
18)   30  1593.
19)   2.1, 0.06, 1880.       ! bottom r2 beta(2),c2
20)   0.,0.                 ! bottom shear beta(3) c2s
21)   60                    ! source depth
22)  0 0 -21 1.0            ! rec depth
23)     212.25 206.62 200.99 195.38 189.76 184.12 178.49
24)     172.88 167.26 161.62 155.99 150.38 144.74 139.12
25)     127.88 122.25 116.62 111.00 105.38 99.755 94.125
26)
27)   1                     ! number of ranges
28)   2140                  ! receiver range
29)
```

```
30)  3                  ! nparm
31)  9  1 1000 5000 128    ! source range
32)  19 1  -10   10 128    ! array shape tilt
33)  8  1   10   90 128    ! source depth
```

The first column of numbers is a line number. The line numbers are normally not present in the SAGA parameter file they are only here for reference so that it is clear which line is being discussed.

The SAGA input parameter file is divided into 5 sections called blocks. The first 3 blocks, and the 5th block are for SAGA. The 4th block contains the forward model parameters.

Block I -- Title

```
01)  SAGA jump start example, data based on physical array (opt X)
```

Line 01) is the title for this file. It is not put on output plots but is put in the output file 'results'. If you keep the title current with the changes made in the file, the output file 'results' will list the updated title with the current results so you can keep track of the changes that were made and their effects. This line is referred to as Block I in the SAGA manual in Chapter 6.

Block II -- OPTIONS

```
02)  c b W              ! options
```

Line 02) are the options to use in this SAGA run. SAGA has more than 50 options available, only a few are discussed here. See the SAGA manual for a complete listing of all the options.

Two options should always be specified in SAGA. The first is the Observed Data Format. SAGA needs to understand the format of the input data. The 'c' option says the input data will be a covariance matrix. See Sect 7.1 for details of this format.

The second option is the Objective Function. Only certain objective functions work with each data type. In this case the default option for the covariance matrix is used. When a default option is used there is no need to specify it on the parameter line. The default option for the objective function is the Bartlett power for each frequency is summed.

The 'b' option specifies that the covariance matrix is normalized by dividing it with the sum of the diagonal. The maximum obtainable Bartlett power is slightly less than one (depending on the noise in the data).

The 'W' option indicates to write the calculated data (using the baseline model) to the *.obs file. The format the data is written out in will be determined by the input data format parameter. In this example it will be written out as a covariance matrix. This line is referred to as Block II in the SAGA manual in Chapter 6.

Block III -- Genetic Algorithm Parameters (2 lines)

```
04)   2000  64  20        ! niter, q, npop
05)   0.8 0.5 0.05  0      ! px,pu,pm
```

Line 04 specifies the genetic algorithm population parameters niter, q, and npop.

niter = 2000 = is the number of forward modeling runs for one population. In this example each population will be run 2000 times.
q = 64 = is the Population size in bits. In this example 64 bits are used.
npop = 20 = is the number of parallel populations to be run. In this example 20 populations will be run.

It is best to only change the niter and npop parameters. The number of forward models run will be equal to niter multiplied by npop. In this example there are 2000 * 20 = 40,000 forward models.

Line 05 Specifies px, pu, and pm.

px = 0.8 = Is the crossover rate.
pu = 0.5 = Is the update rate.
pm = 0.05 = Is the mutation rate.

It is best to leave the parameters fixed at this time.

Block IV -- Forward Model Parameters (forward model SNAP used in this example)

```
06)  t                    ! snap options
07)   2 100                ! freq  max_no_modes
08)  49 388
09)   213.,0.,0.,0          ! water depth scatt(1),scatt(2),beta(0)
10)       0.      1521.94
11)      10.1176   1518.77
12)      19.7780   1499.80
13)      30.2703   1495.00
14)      60.5963   1490.78
15)     213.0000   1488.26
16)   30 1.76 0.2           ! sediment thickness, r1, beta(1)
17)   0.0 1572.
18)   30  1593.
19)   2.1, 0.06, 1880.       ! bottom r2 beta(2),c2
20)   0.,0.                ! bottom shear beta(3) c2s
21)   60                   ! source depth
22)  0 0 -21 1.0            ! rec depth: first last Nrec tilt
23)    212.25 206.62 200.99 195.38 189.76 184.12 178.49
24)    172.88 167.26 161.62 155.99 150.38 144.74 139.12
25)    127.88 122.25 116.62 111.00 105.38 99.755 94.125
26)
27)   1                    ! number of ranges
28)   2140                 ! receiver range
```

These parameters specify all the information needed to run the forward model.

Line 06 indicates the options to pass on to snap in this example the 't' option is used. This option allows tilt of the receiver array. Tilt is measured as the horizontal deviation at the last receiver (in meters).

Line 07 indicates the number of frequencies and the maximum number of modes for each frequency.

Line 08 this line lists the frequencies (in Hz) that will be used in this forward model. Be sure that the number of frequencies listed matches the previous lines number of frequencies.

Line 09 lists the water depth as well as various scatter parameters. It is important that the water depth value is also the last depth in the sound speed profile or you will get an error.

Line 10 - 15 contain the sound speed profile. The first point must be at zero meters depth, the last point must match the water depth value on line 9.

Line 16 defines the sediment thickness, r(1), and beta(1) parameters. Sediment thickness is in meters.

Line 17 - 18 contains the sound speed profile for the sediment. The first point must be at zero meters depth, the last point must match the sediment depth value on line 16.

Line 19 defines bottom parameters r2, beta(2), and c2

Line 20 defines bottom shear parameters beta(3) and c2s

Line 21 defines the source depth. If searching for source depth parameter this value must be in the range of the search as it is used to define an initial guess.

Line 22 specifies the receivers. Normally the parameters RDmin RDmax and Nrd are all that is needed:

RDmin - is the minimum receiver depth to be used
RDmax - is the maximum receiver depth to be used
Nrd - is the number of receivers to be equally spaced on the Z-axis.with the first element at RDmin and last element at RDmax.

If Nrd is negative, as in this example, then | Nrd | receiverdepths are read individually in the next line. But with the tilt 't' option for snap a fourth parameter is added which is array tilt in meters. The array tilt is measured as the horizontal deviation at the last receiver measured in meters. See the bottom of this section for more information on receiver options.

Line 23 - 25 are the specified receiver depths for the Nrd elements; in this example there are 21 elements.

Line 27 defines the number of ranges to run SAGA at (meters).

Line 28 defines the range for the first run. Range is defined as the distance between the source and element #1 of the array.

Block V -- Optimization Parameters
```
  30)  2                    ! nparm
  31)  9 1  1000 5000 128    ! source range
  32)  19 1  -10   10 128    ! array shape tilt
  33)  8  1   10   90 128    ! source depth
```

Line 30 defines the number of parameters that SAGA will optimize.

Each forward model in SAGA has its own mapping between the optimization variable and the environmental parameters to be optimized. Each optimization parameter has the following form: param index Fmin Fmax Ndiscr where

- param - is a pointer used to map the variable to the environmental parameter. A list of the pointers available in SNAP is in section 12.4.3
- index - is a subpointer. In many case there are no subclasses so this value will be set to one. Some parameters have multiple options like surface roughness (param = 5). If index=1 then ocean surface roughness is optimized. If index=2 then bottom roughness is optimized
- Fmin - Lower bound for the search parameter
- Fmax - Upper bound for the search parameter
- Ndiscr- The number of discrete values that the range defined by Fmin and Fmax will be split into. Ndiscr should be between 2 and 1200 (a max set at compile time). When using Genetic Algorithms powers of 2 are most efficient. But for simulated annealing, contour plots, exhaustive sampling, and Gibbs it is not important it be a power of 2.

Line 31 shows param=9, index=1 which maps to an optimization parameter for source range. The search bounds are Fmin=1000, Fmax=5000. Ndiscr=128 so the search range will be split in 128 evenly spaced discrete values.

Line 32 shows param=19, index=1 which maps to an optimization parameter for array shape tilt. The search bounds are Fmin=-10, Fmax=10. Ndiscr=128 so the search range will be split in 128 evenly spaced discrete values.

Line 33 shows param=8, index=1 which maps to an optimization parameter for source depth. The search bounds are Fmin=10, Fmax=90. Ndiscr=128 so the search range will be split in 128 evenly spaced discrete values.

4. SAGA jumpstart PartII - how to input array shapes in SAGA for the forward model SNAP and SNAP3D

There are many ways to input receiver information (array shape) to the forward model SNAP (or SNAP3D). A quick summary of the options available with the forward model SNAP and SNAP3d are listed here. This information is from the SAGA manual chapters 12 and 13 in the sections about SNAP and SNAP3D. The section 12.4.2 lists the snap options, and the section 13.3 list the additional snap3d options.

SNAP works in the XZ plane. The Y axis is pointing out of the plane. The array is located on the Z-axis, and the deviation in the x-axis. Z is zero at the surface and points downward, X is zero at element #1 location and points away from the source. In SNAP, the array is never moved out of the XZ plane.

To define the receiver locations in snap there are two main methods. Without any options for snap the receiver locations are specified by line 22 parameters RDmin, RDmax, and Nrd. If you want to specify the receiver locations exactly make Nrd negative, and on the next line list all of the receiver depths. Make sure to have exactly Nrd values or you will get an error.

The 'd' option allows each receiver to have its own range offset (X-axis). If this option is used a line should be inserted between Line 22 and Line 23 where the range offset for each element is specified in meters. Positive X values are farther from the source. Once again make sure to have exactly Nrd values or you will get unexpected results.

There are a few options available in SAGA for SNAP that are not listed in chapter 12. The options 'o', 'x', 'x2', 'x3' are part of the SNAP3D set of options found in chapter 13. SNAP, as explained above, only works in the XZ plane. When using SNAP3D options the receivers are defined in the XYZ plane, but it is the projection of the receivers into the XZ plane that is used by SNAP.

When using the SNAP3D options there are two additional lines that are added to the SAGA parameter file. Between the receiver depth line and the number of source ranges line the following lines are added:

100 15 ! array param: length[only with option 'o'], bow [not with x2 or x3]
90 90 0 ! angles: rotation 1 (Z-axis), rotation 2 (Y-axis), rotation 3 (Z-axis)

When using the SNAP3D parameters the array is approximated as a parabola. The length and bow parameters of the first line describe the parabola shape. The angles of rotation describe the parabola's orientation in the XYZ plane, which are then projected into the XZ plane for running SNAP.

The 'array parameter' line has two values: length and bow. Length is the length (meters) of the array. In the above line the length is set to 100m. The receivers will be equally spaced along the length of the array starting at the RDmin depth for the first element (RDmax is not used when the length parameter is used). The length parameter is only used with the 'o' option.

The bow parameter is the maximum deflection from the center of the array to the Z-axis in meters. It is only used with the o and x option. The x2 and x3 options define the array shape in more detail.

The angles of rotation are Eulerian angles. There is a discussion of this along with figures in section 13.1.2 of the SAGA manual.

Rotation-1 (Z-axis) - rotate the array shape clockwise around the Z-axis out of the XZ plane. The end points of a parabola shaped array would not change position. If the array were a rod, no effect would visible from this rotation.

Rotation-2 (Y-axis) - rotate the array shape around the Y-axis. This rotation adds tilt to the array shape. A positive value for rotation 2 will move the tail of the array away from the source toward increasing x values. If the array were a rod, and rotation 2 was 90 degrees this rotation would produce a horizontal array. The array would end up perpendicular to Z-axis, parallel to the X-axis. with element 1 at the x=0 and the last element at x=array length.

Rotation-3 (Z-axis) - rotate the array shape clockwise around the Z-axis.
This rotation foreshortens the array n the XZ plane. If rotation 2 is non-zero this foreshortens the array. If rotation 2 is zero, rotation 3 should be zero as it is just the same rotation a rotation 1.

It is difficult to understand the rotations in 3 dimensions. In SAGA's matlab directory is a M-file called rotation3d.m which will allow you to experiment in Matlab with these rotations in a visual way.

The SNAP3D options are more fully explained in the SAGA manual under section 13.3 and 13.4.

Option 'o' simulates the array as a parabola with equidistant receivers. RDmin (line 22) is used as the initial Z-axis coordinate for element 1.The length of the array is from the SNAP3D 'length' parameter, Nrd-1 elements are equally spaced along the array in the Z-axis. TheSNAP3D 'bow' parameter is the maximum deflection of the array (on the X-axis). The array shape is rotated using the Eulerian angles.

Option 'x' is like the 'o' option except that instead of equidistant receivers, the Z coordinate of each receiver is specified in the parameter file. The first element must be at the origin (0,0). All of these values will have the receiver depth value RDmin added to them. The array shape is then rotated by the three Euler angles.

Option 'x2' With this option the z and x coordinates for each receiver are specified in the parameter file. As with all the X options the first receiver must be at the origin (0,0). All of these values will have the receiver depth value RDmin added to them. The array shape is then rotated by the three Euler angles.

Option 'x3' with this option the x,y,and z coordinates for each receiver are specified in the parameter file. The first receiver must be at the origin (0,0).

5. SAGA jumpstart part III - running SAGA

The actual running of SAGA is straight forward:

```
saga paramFile forwardModel
```

For all of these examples we will use SNAP (normal modes) as the forward model. The argument paramFile is a prefix that SAGA will use to name input and output files. For example:

```
saga vert snap
```

will tell SAGA to run the forward model SNAP and to get the input parameters from the file "vert.dat". SAGA will use this prefix for all files associated with this parameter set.

Here is a brief explanation of a few files that the beginning SAGA user should be familiar with:

input files needed to run SAGA:
- *.dat - input parameter file
- *.in - input datafile for Saga

output files produced by running SAGA:
- results - results from each run are appended to this file.
  most recent results are at end of the file
- snap.dat - same as *.dat except that the inversion results have
  been inserted as the input parameters.

*.obs - synthetic data file generated by SAGA
(only present when "W" option used)

file produced by SAGA that you only need to work with if something goes wrong.

*.out - output file from SAGA, can help debug runtime problems

5.1 Producing plots from SAGA output:

Most of the plots are produced by SAGA directly. But the 'a posteriori' distribution plot requires the use of the post-processor program POST to analyze the results and compute the best, most likely, and mean parameter vectors. The program POST is run with the same arguments as when running SAGA:

```
post paramFile forwardModel
```

for example:

```
post vert snap
```

will generate the following files that the MATLAB scripts will use to plot:

*.plt - data values for producing a posteriori distribution plot
(inversion plots)
*.plp - supplemental info for plot such as axis ranges , axis labels,
title info, ...
*.cdr - this file is not used when plotting from MATLAB

*.bdr - data values for producing the contour plot

in MATLAB use the following commands to produce the desired plots:

```
% for a posteriori distribution plot (inversion plot)
% after running the post processor program POST
>> plotsaga
paramFile
print('-dpsc','SAGAinversion.ps')


% for scatter plot of objective function use
>> sagascat
paramFile
print('-dpsc','SAGAscatter.ps')

% for contour plot  use
% note that the contour plot only plots the first parameter(X-axis)
% against the second parameter(Y-axis). It ignores the rest of the
% parameters as far as the contour plot is concerned.
>> contsaga
paramFile
```

print('-dpsc','SAGAcontour.ps')

to generate synthetic data

- remove the current data " rm *.in"
- add the "W" option in the paramFile
- run SAGA (first time generate synthetic data)
- rename the generated synthetic from *.obs to *.in
- run SAGA (second time compute inversion)

The synthetic data is generated each time SAGA is run when the W option is selected. If there is no *.in file SAGA will end after generating the data complaining that the input data file is not found. Then just rename the *.obs file to *.in file. Now run SAGA with the simulated data just generated.

Annex A : Detailed description of examples

Example 1 : SAGA.ex1.VLA.sim

```
   This is a simple example of how to run SAGA with a vertical array on
a single frequency ( 388 Hz ).  The script runSAGA.scr will first run
SAGA to generate the simulated data file. Then copy the input file from
a *.obs extension to a *.in extension.
   Now that the simulated data is set as the input file, run SAGA a
second time to produce the 'a posteriori' distribution data.  The
post-processor POST is run on the SAGA output to compute the 'a posteriori'
distribution best, most-likely, and mean fit results.  POST also appends a
text copy of the results to the file results.
   To produce the contour plot, repeat these steps with a different input
parameter file for SAGA (vertC.dat).
   To generate the two postscript plots use Matlab.  Using the built in
M-files in the SAGA matlab directory generate the inversion plot and the
contour plot.

   The inversion plot shows a panel for each parameter searched. The plot
shows that in each panel (search parameter) the result seems to be well
determined.
   The contour plot shows the first two optimization parameters plotted
as a contour plot with the first search parameter as the X-axis and the
second search parameter as the Y-axis.  Note that only one frequency is
used as the highest frequency available will be the one that is most
sensitive to array tilt.  388 Hz was chosen as it was the highest frequency
available in the real data.

   The results file looks like:
>>m results
SAGA jumpstart Example,  search for source range & depth + array tilt
 ***************
  best obtainable theoretical fit -1.0414954E-08
  best fit (best, ppd, mean)  0.2469351      0.2494895      0.3463788

                                best-of all  most likely     mean   std-dev
  1 Source range (m)          1  2133.858     2133.858  2134.427     0.083
  2 Array tilt (m)            1    -1.181       -1.024    -0.745     0.073
  3 Source depth (m)          1    59.764       59.764    58.565     0.082

note that the range parameter is off from the true range(2120m) by 13.9 m.
If we determine the interval between discrete samples for the source range
parameter we see that (5000m - 1000m+1)/128 intervals = 31.257 m between
discrete samples.  This is greater than the range parameter is off by, so
SAGA found the closest bin.

Now double the number of discrete intervals from 128 to 256 for the range
parameter and rerun the SAGA example(takes a little bit longer).  The results
improve to:

SAGA jumpstart Example,  search for src range & depth + tilt  (range Nincr=256)
 ***************
  best obtainable theoretical fit -1.0414954E-08
  best fit (best, ppd, mean)  6.9761269E-02  6.9761269E-02  6.2417135E-02

                                best-of all  most likely     mean   std-dev
  1 Source range (m)          1  2113.726     2113.726  2114.106     0.001
  2 Array tilt (m)            1    -0.824       -0.824    -0.824     0.012
  3 Source depth (m)          1    60.196       60.196    60.131     0.005
```

Now the range parameter is off from the true range by 6.3 m, roughly cutting the difference in half.

Pay attention to the size of each discrete interval for each search parameter to make sure that its limit will be accurate enough for the work being done.

summary of what is in the directory

scripts:
runSAGA.scr - script to run SAGA and produce 'a posteriori' distribution plot

input parameter files:
vert.dat - SAGA input parameter file (options set to compute inversion)
vertC.dat - SAGA input parameter file (options set to produce contour plot)

results output by SAGA:
results - results of both runs in test file format
SAGAcontour.ps - contour postscript plot
SAGAscatter.ps - 'a posteriori' distribution plot (inversion plot)

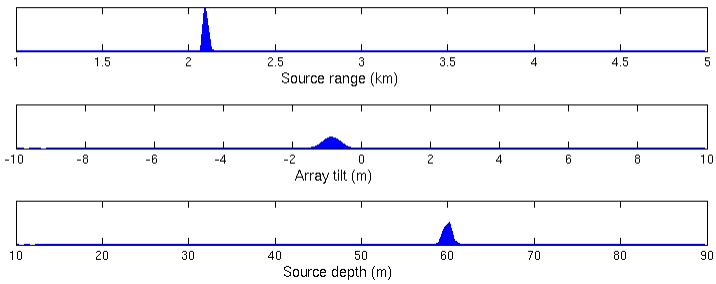

Figure 1 : 'A posteriori' distribution plot for example 1

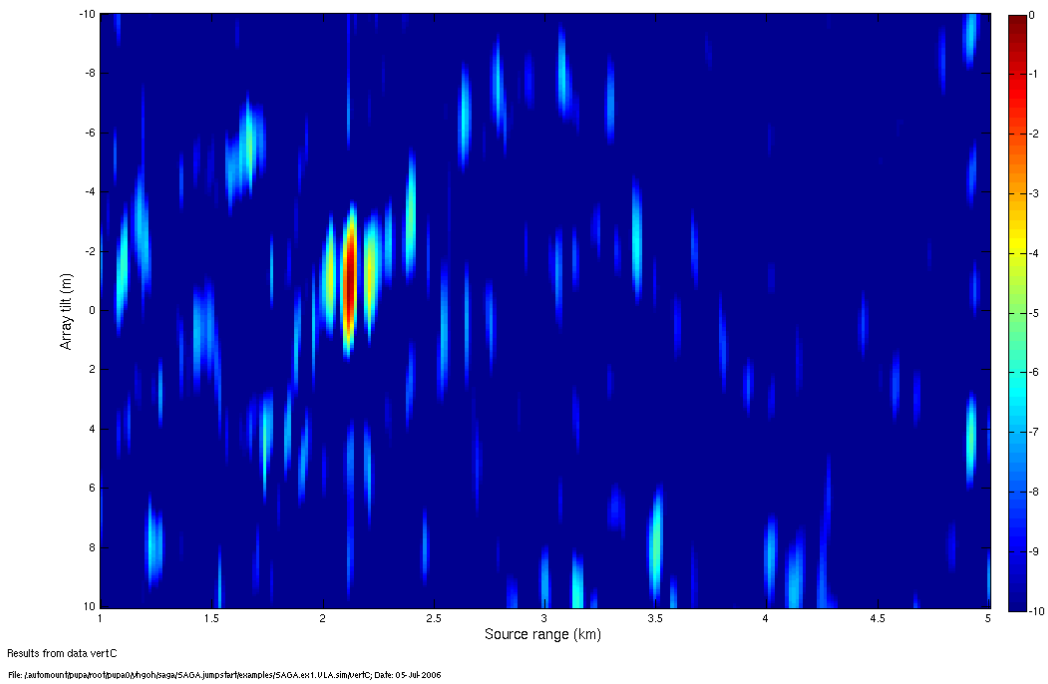

Figure 2 : Contour plot for example 1

Example 2 : SAGA.ex2.VLA

```
   This is a simple example of how to run SAGA with a vertical array on
a single frequency ( 388 Hz ).  This example uses real data from the SWellEx-96
experiment.  The script runSAGA.scr will first obtain the input data in a form
that SAGA likes. Also the data file needs to be copied from the mpl website
before this section is run.  Read the file 'getData.txt' in the subdirectory
fftData for details on how to do this.
   In this example SAGA is using the 't' option for the forward model snap.
This option measures the tilt as the horizontal deviation at the last receiver
measured in meters. In SwellEx-96 the VLA was moored to the bottom. But the last
element is also the bottom element.  This would mean that when computing the
tilt the bottom of the array would move instead of the top.  To better
replicate the real world case need to have the last element be at the top
of the array so SAGA will move the top of the array when computing tilt.
   To make this change the receiver locations need to be reversed in the
files 'vert.dat' and 'vertC.dat' (input parameter files for SAGA). The receiver
positions also need to be reversed in the 'convert_to_saga.m' M-file that makes
the input data file for SAGA.  Lastly the order of the channels needs to be
reversed in order to keep the receiver depths matched up to the correct data
channels.
   The script runcsdm.scr first converts the CSDM data file in fftData
to a SAGA formatted data file. This is done in Matlab with a SAGA provided
M-file.  This M-file expects to have a single input file for each frequency
being processed. This M-file produces an output file called 'cov.in'
This is the input data file for SAGA.
   Now that the data is set as the input file, the script runSAGA.scr runs SAGA
to produce the inversion data using the input parameter file 'vert.dat'.  The
post-processor POST is run to compute the 'a posteriori' distribution.
POST also appends a text copy of the results to the file 'results'.
   The script runs SAGA a second time using the input parameter file
'vertC.dat' to produce a contour plot showing source range verse array tilt.
   Matlab is run to generate the two postscript plots with plotSAGA.scr
using the provided M-files in the SAGA matlab directory to generate the
inversion plot and the
contour plot.
```

The inversion plot shows a panel for each parameter searched. The plot shows that source range and depth are well determined. The tilt panel shows a less peaked but reasonably well defined result for the tilt. The peak for the tilt is at -1.811 m. The contour plot shows the source range verses array tilt since these are the first two search parameters in the input parameter file.

The contour plot always shows the first two optimization parameters plotted with the first search parameter as the X-axis and the second search parameter as the Y-axis. Note that only one frequency is used as the highest frequency available will be the one that is most sensitive to array tilt. 388 Hz was chosen as it was the highest frequency available in the real data.

The results file looks like:

```
SAGA jumpstart Example 2,  VLA processing (inversion) with Real Data
 ***************
  best obtainable theoretical fit  0.3884194
  best fit (best, ppd, mean)  0.6709696      0.6709696      0.6648355

                                best-of all  most likely     mean   std-dev
  1 Source range (m)           1  3204.724     3204.724  3211.038     0.009
  2 Array tilt (m)             1    -1.811       -1.811    -1.762     0.018
  3 Source depth (m)           1    74.882       74.882    74.689     0.020


The tilt of -1.811 meters is equivalent to  -0.865 degrees

      theta = arcsin(array_tilt/array_length)
            = arcsin( 1.811 m / 120 m )
            = arcsin( 0.015092 )
            = 0.865 degrees
```

This matches well to the expected tilt of between 0.5 and 1.0 degrees for this time period. It is important to keep in mind that tilt in SAGA is the horizontal deviation at the last reciever measured in meters.

For this data set the source range is 3 km and the source depth is 60 m. The source range and depth are longer and deeper than expected. This is due to the mirage effect caused by different bathymetries. In this case the source is in 200m of water and the array is in 216.5 meters of water. A difference of ~7.7 % thus the values need to be reduced by this amount giving a range of 3204.724 / 1.077 = 2975.6 m and for the source depth of 74.882 / 1.077 = 69.528 m. The source depth is still a bit deep (expecting 60 m) but this is only one frequency. See examples 3, 4, and 5 for demonstrations of SAGA with multiple frequencies.

To learn more about the mirage effect see the section on the mirage effect in the covert calibration example notebook. Also the paper 'Mirages in shallow water matched field processing' by Gerald D'Spain, JASA June 1999.

summary of what is in the directory

scripts (to run in order):
runcsdm.scr - script to generate covariance matrix and write it to cov.in using real data
runSAGA.scr - script to run SAGA and produce data for 'a posteriori' distribution plot
and contour plot.
plotSAGA.scr - script to produce the 'a posteriori' distribution plot and contour plot
using plotsaga and contsaga in matlab

input files:
vert.dat - SAGA input parameter file (options set to compute inversion)
vertC.dat - SAGA input parameter file (options set to produce contour plot)
cov.in - input data for SAGA

results output by SAGA:
results - results of both runs in test file format
SAGAcontour.ps - contour postscript plot

```
SAGAscatter.ps - 'a posteriori' distribution plot (inversion plot)

directory:
fftData - subdirectory that shows how to obtain the original time series
          data. Contains the matlab file compute_csdm.m to compute the
covariance matrix.

M-files:
compute_csdm.m - Matlab script to compute the covariance matrix.
write_covmat.m - Matlab script used by convert_to_saga.m to write out the
covariance matrix.
```

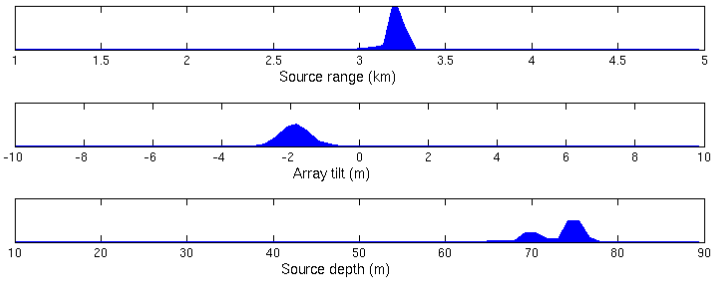

Figure 3 : 'A posteriori' distribution plot for example 2

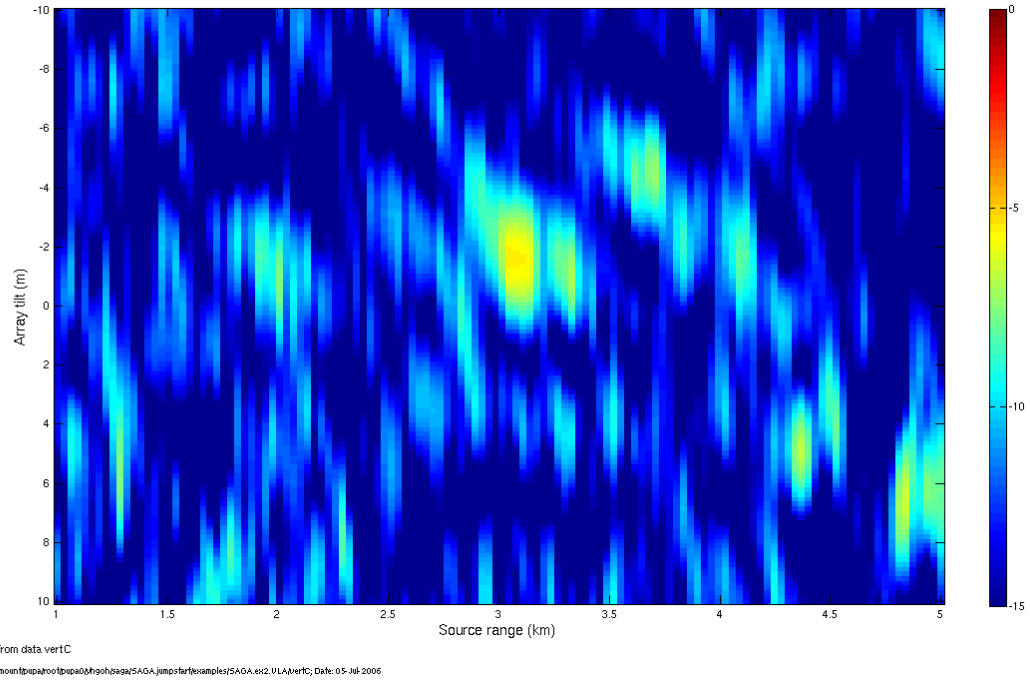

Figure 4 : Contour plot of array tilt vs source range for example 2

Example 3 : SAGA.ex3.VLA.sim

```
   This is an example of how to run SAGA to do Match Field Processing with
multiple frequencies using a vertical array.  This simulated example matches
the SWellEx96 Event S5 data set at J131 23:55.  The source is at a range
```

of 3 km and depth of 60m. The array tilt is between 0.5 and 1.0 degrees. For the simulation the tilt was set at -2 meters (or 0.95 degrees).

The script runSAGA.scr will first run SAGA to generate the simulated data file. Then copy the input file from a *.obs extension to a *.in extension.

Now that the simulated data is set as the input file, run SAGA a second time to produce the 'a posteriori' distribution data. The post-processor POST is run on the SAGA output to compute the 'a posteriori' distribution best, most-likely, and mean fit results. POST also appends a text copy of the results to the file results.

To produce the contour plot, repeat these steps with a different input parameter file for SAGA (vertC.dat). Note that MFP is done with only the first two parameters when generating the contour plot. Any additional parameters are not searched, they are set to the initial value in the *.dat file. This may not be the optimal value found from the inversion.

To generate the two postscript plots use Matlab. Using the built in M-files in the SAGA matlab directory generate the inversion plot and the contour plot. In this example the SAGA provided M-file to produce the contour plot has been modified to allow for choice of color bar range. In the script runSAGA.scr the matlab plotting portion now has a cmin and cmax parameter. When the M-file 'contsaga.m' is called it uses the values for cmin and cmax as the ranges in dB for the colorbar.

The inversion plot shows a panel for each parameter searched. The plot shows that in each panel (search parameter) the result seems to be well determined. The addition of 12 more frequencies does not improve the tilt result as most of this information comes from the highest frequency signal. But the source depth and range is better determined with multiple frequecies instead of just one.

The results file looks like:

```
SAGA jumpstart Example,  search for src range & depth + tilt  range Nincr=128
 ***************
  best obtainable theoretical fit  2.0062746E-08
  best fit (best, ppd, mean)  0.1480726      0.1480726      0.1325386

                                 best-of all  most likely     mean   std-dev
  1 Source range (m)           1  2984.252     2984.252  2984.954     0.007
  2 Array tilt (m)             1    -2.283       -2.283    -2.273     0.036
  3 Source depth (m)           1    60.866       60.866    60.245     0.009
```

So with simulated results the match is very close to the true location of the source range, source depth, and array tilt.

In this example two contour plots are generated, 'source range verses array tilt' is the first plot generated. The range of the source is the x-axis and the y-axis is the array tilt in meters.

The second contour plot displays the 'source range verse source depth'. This shows a very good match for the source. A simple experiment to try is to reduce the number of frequencies and see how the results degrade.

summary of what is in the directory

scripts:
runSAGA.scr - script to run SAGA and produce 'a posteriori' distribution plot
and contour plot.

input files:
vert.dat - SAGA input parameter file (options set to compute inversion)
vertC.dat - SAGA input parameter file (produce range verses tilt plot)
vertC2.dat - SAGA input parameter file (produce range verses depth plot)
cov.in - input data for SAGA

results output by SAGA:
results - results of both runs in test file format
SAGAscatter.ps - 'a posteriori' distribution plot (inversion plot)
SAGAcontour.ps - contour plot (range verse tilt)
SAGAcontour2.ps - contour plot (range verse depth)

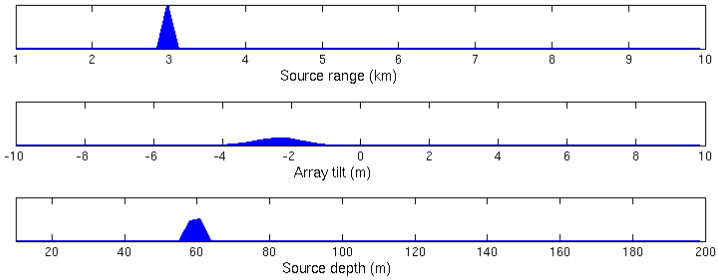

Figure 5 : 'A posteriori' distribution plot for example 3

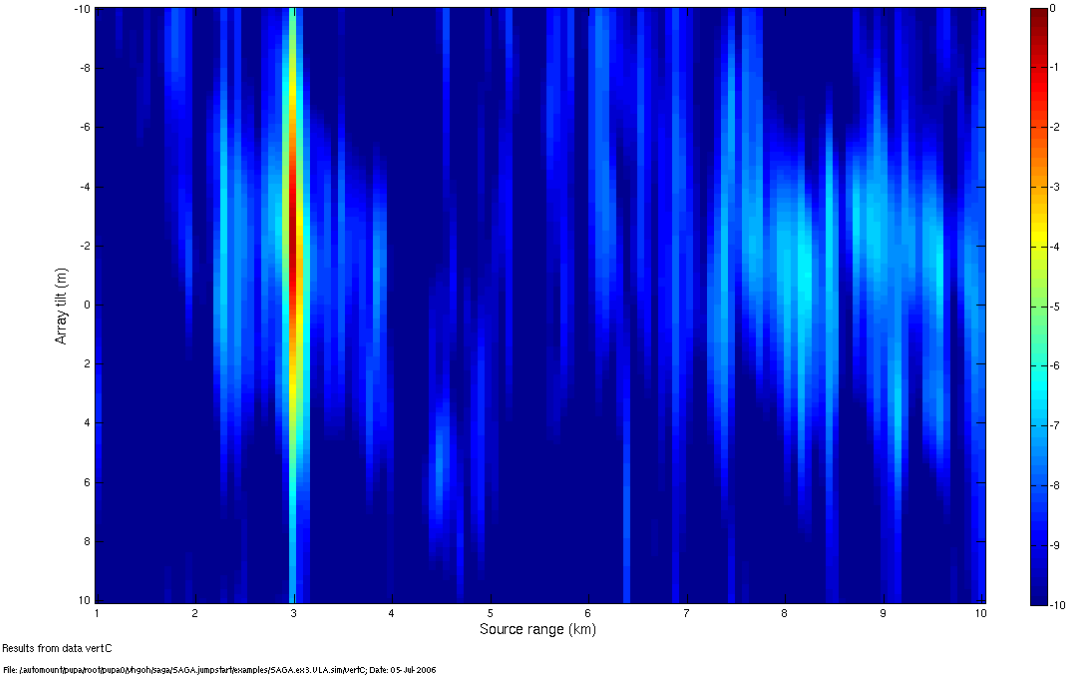

Figure 6 : Contour plot of array tilt vs source range for example 3

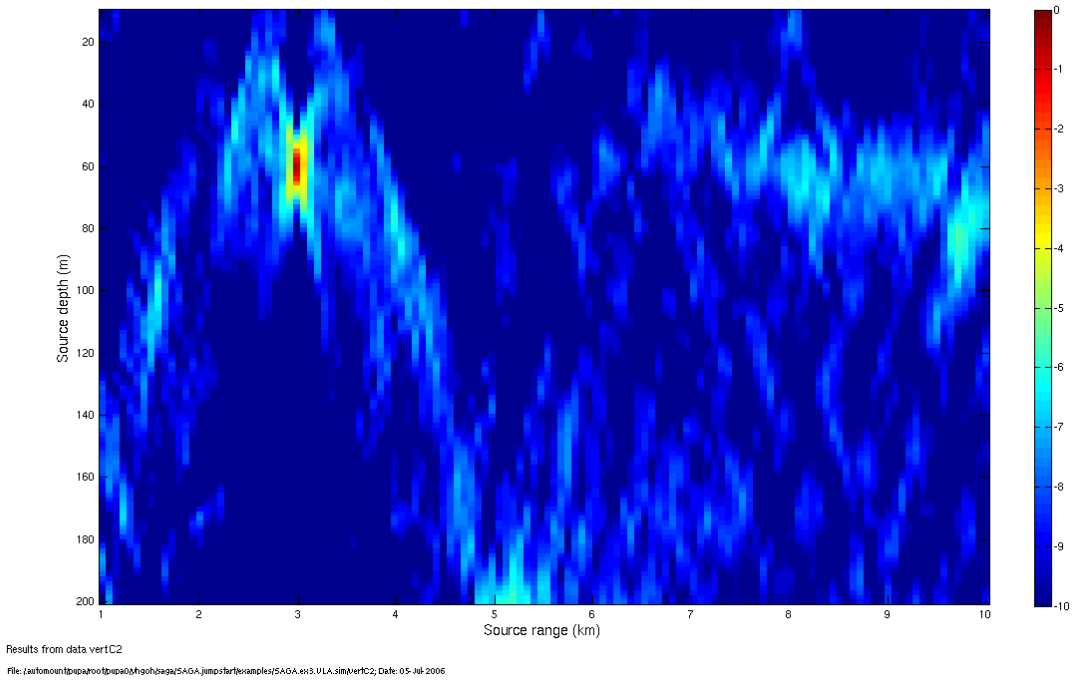

Figure 7 : Contour plot of source depth vs range for example 3

Example 4 : SAGA.ex4.VLA.kraken

```
   This is an example of how to run SAGA to do Match Field Processing with
multiple frequencies using a vertical array.  The input (simulated) data for
this example is generated by Kraken and imported into SAGA.
   This simulated example matches the SWellEx96 Event S5 data set at
J131 23:55.  The source is at a range of 3 km and depth of 60m.  The array
tilt is between 0.5 and 1.0 degrees.  For the simulation the tilt was set
at -2 meters ( or 0.95 degrees ).
   The script runSAGA.scr will first run SAGA to generate the simulated data
file. Then copy the input file from a *.obs extension to a *.in extension.
   Now that the simulated data is set as the input file, run SAGA a
second time to produce the 'a posteriori' distribution data.  The
post-processor POST is run on the SAGA output to compute the 'a posteriori'
distribution best, most-likely, and mean fit results.  POST also appends a
text copy of the results to the file results.
   To produce the contour plot, repeat these steps with a different input
parameter file for SAGA (vertC.dat). Note that MFP is done with only the
first two parameters when generating the contour plot.  Any additional
parameters are not searched, they are set to the initial value in the *.dat
file.  This may not be the optimal value found from the inversion.
   To generate the two postscript plots use Matlab.  Using the built in
M-files in the SAGA matlab directory generate the inversion plot and the
contour plot.  In this example the SAGA provided M-file to produce the
contour plot has been modified to allow for choice of color bar range.
In the script runSAGA.scr the matlab plotting portion now has a cmin
and cmax parameter.  When the M-file 'contsaga.m' is called it uses the
values for cmin and cmax as the ranges in dB for the colorbar.

     The inversion plot shows a panel for each parameter searched. The plot
shows that in each panel (search parameter) the result seems to be well
determined. The addition of 12 more frequencies does not improve the tilt
result as most of this information comes from the highest frequency signal.
But the source depth and range is better determined with multiple frequecies
instead of just one.
```

The results file looks like:

```
SAGA jumpstart Example,  search for src range, depth, tilt  range Nincr=128
 ***************
  best obtainable theoretical fit -4.1751056E-09
  best fit (best, ppd, mean)  0.2592583      0.2607544      0.1180783

                                best-of all  most likely     mean   std-dev
  1 Source range (m)          1  3055.118     3055.118  3026.758     0.004
  2 Array tilt (m)            1    -2.441       -2.283    -2.071     0.037
  3 Source depth (m)          1    59.370       59.370    59.846     0.006
```

So with Kraken generated data the match is very close to the true location of the source range, source depth, and array tilt. With a perfect match the peak level would be 0 dB. This result has a peak value of -1.36 dB.

In this example two contour plots are generated, 'source range verses array tilt' is the first plot generated. The range of the source is the x-axis and the y-axis is the array tilt in meters.

The second contour plot displays the 'source range verse source depth'. This shows a very good match for the source. A simple experiment to try is to reduce the number of frequencies and see how the results degrade.

summary of what is in the directory

scripts:
runSAGA.scr - script to run SAGA and produce 'a posteriori' distribution plot
and contour plot.

input files:
vert.dat - SAGA input parameter file (options set to compute inversion)
vertC.dat - SAGA input parameter file (produce range verses tilt plot)
vertC2.dat - SAGA input parameter file (produce range verses depth plot)
cov.in - input data for SAGA

results output by SAGA:
results - results of both runs in test file format
SAGAscatter.ps - 'a posteriori' distribution plot (inversion plot)
SAGAcontour.ps - contour plot (range verse tilt)
SAGAcontour2.ps - contour plot (range verse depth)

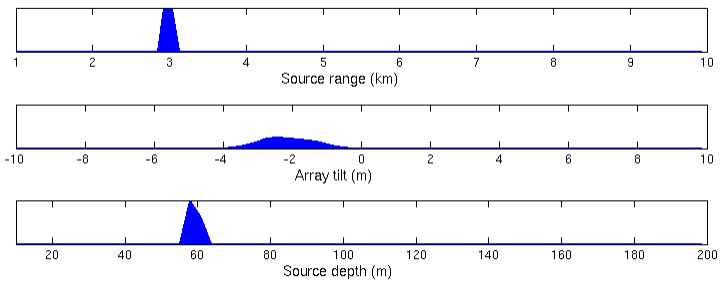

Figure 8 : 'A posteriori' distribution plot for example 4

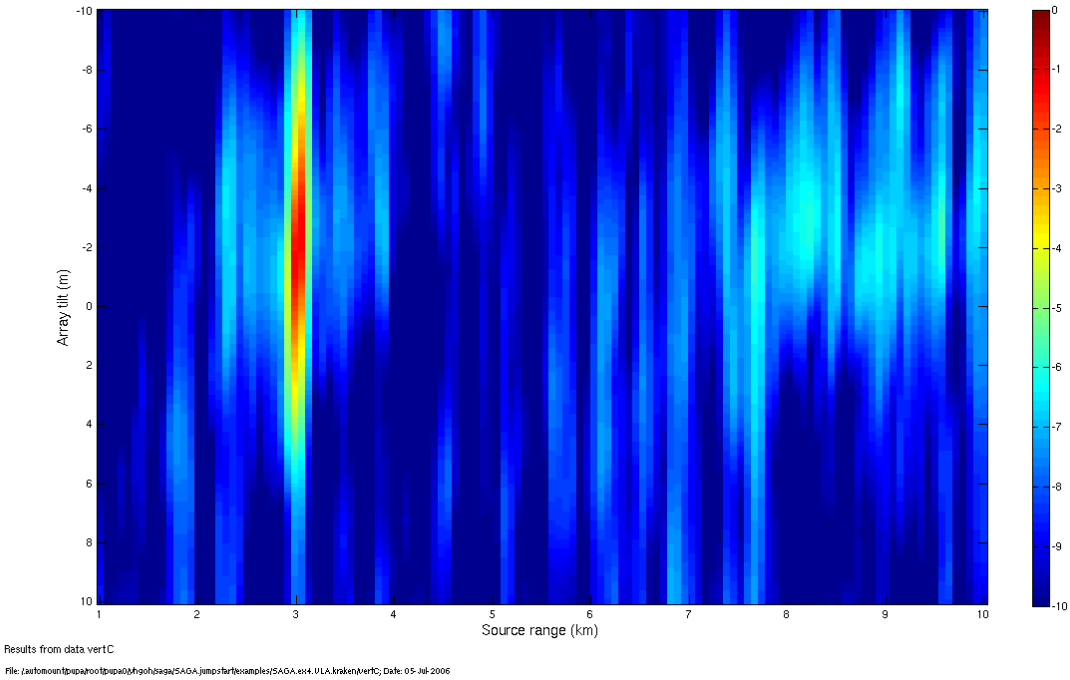

Figure 9 : Contour plot of array tilt vs source range for example 4

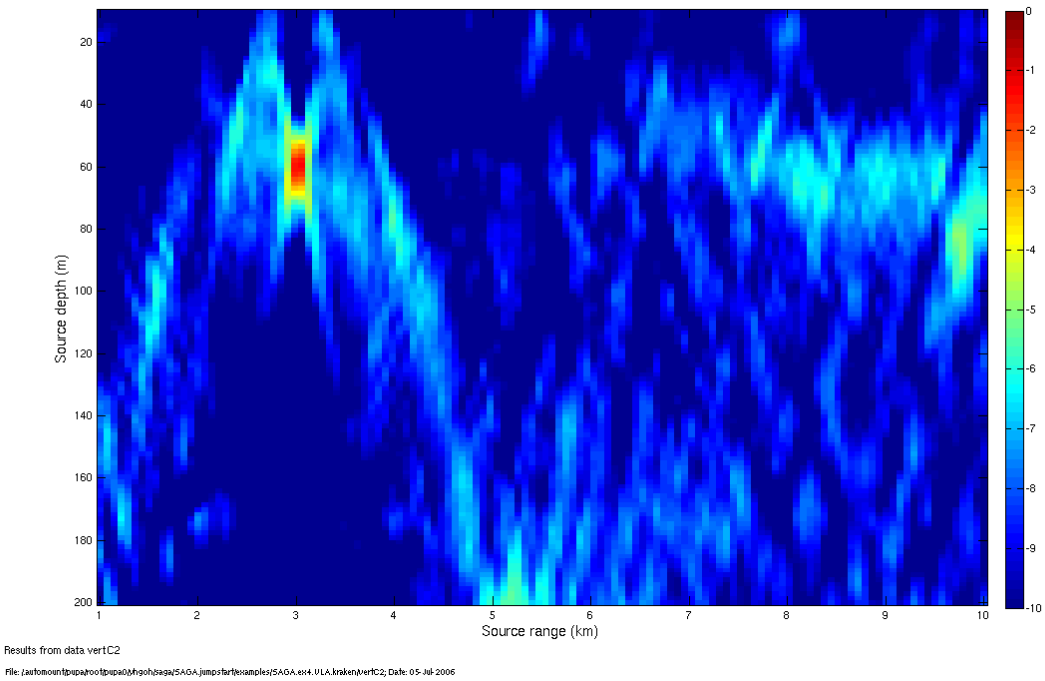

Figure 10 : Contour plot of source depth vs range for example 4

Example 5: SAGA.ex5.VLA.real

This is a simple example of how to run SAGA with a vertical array on a multiple frequencies (49 Hz - 388 Hz) The script runSAGA.scr will first obtain the input data in a form that SAGA likes. See the ReadMe in the subdirectory 'fftData' for details on this process. This section is currently commented out to increase the speed of the example. Also the data file needs to be copied from the mpl website before this section is run. Read the file 'getData.txt' in the subdirectory 'fftData' for details of how to do this.

The script runcsdm.scr first converts the CSDM data file in fftData to a SAGA formatted data file. This is done in Matlab with a SAGA provided M-file. This M-file expects to have a single input file for each frequency being processed. This M-file produces an output file called 'cov.in' This is the input data file for SAGA.

Now that the data is set as the input file, run SAGA to produce the inversion data using the input parameter file 'vert.dat'. The post-processor POST is run to compute the 'a posteriori' distribution. POST also appends a text copy of the results to the file 'results'.

The script runs SAGA a second time using the input parameter file 'vertC.dat' to produce a contour plot of source range verses array tilt. The third run of SAGA produces the source range verses source depth using the parameter file vertC2.dat. The ambiguity surfaces plotted are generated with MFP processing using only the first two parameters in the input file. Any additional search parmeters are not search, but default to their inital value. For the optimal MFP search results it is important to have the initial values in the search parameter files for the contour plots be set to best fit results from the 'a posteriori' distribution.

Matlab is run to generate the three postscript plots. Using the provided M-files in the SAGA matlab directory generate the inversion plot and the contour plots.

The inversion plot shows a panel for each parameter searched. The plot shows that source range and depth are well determined. The tilt panel shows a less peaked but reasonably well defined result for the tilt. The peak for the tilt is at -1.429 m. This peak does show clearly on the contour plot. The second contour plot shows the source depth verses range. It has a defined peak at a slightly longer range between 60 and 70 meters depth. The value that SAGA chose is 64.286 m deep for the source depth. This depth is a bit deeper than expected

The contour plot always shows the first two optimization parameters plotted with the first search parameter as the X-axis and the second search parameter as the Y-axis. For optimal MFP results remaining parameters should have their inital values set to the results of the 'a posteriori' best fit.

The results file looks like:

```
>>m results
SAGA jumpstart Example 5,  VLA processing (inversion and MFP) with Real Data
 ***************
  best obtainable theoretical fit  0.2508154
  best fit (best, ppd, mean)  0.4566793      0.4566793      0.5489922

                                best-of all  most likely     mean   std-dev
  1 Source range (m)           1  3142.857     3142.857  3184.718     0.053
  2 Source depth (m)           1    64.286       64.286    66.125     0.040
  3 Array tilt (m)             1    -1.429       -1.429    -1.474     0.057


The tilt of -1.429 meters is equivalent to  -0.682 degrees

      theta = arcsin(array_tilt/array_length)
            = arcsin( -1.429 m / 120 m )
            = arcsin( -0.01191 )
            = -0.682 degrees
```

This matches well to the expected tilt of between 0.5 and 1.0 degrees for this time period. It is important to keep in mind that tilt in SAGA is the horizontal deviation at the last reciever measured in meters.

```
For this data set the source range is 3 km and the source depth is 60 m.
The source range and depth are longer and deeper than expected.  This
is due to the mirage effect caused by different bathymetries. In this
case the source is in 200m of water and the array is in 216.5 meters
of water.  A difference of ~7.7 %  thus the values need to be reduced
by this amount giving a range of 3142.857 / 1.077 = 2918.2 m and
for the source depth of 64.286 / 1.077 = 59.690 m.

In example 2 ( single frequency ) after correcting for the mirage effect
the range was very close but the depth was off by almost 10 meters. With
13 frequencies the depth matches and the range is still within one range
cell.

To learn more about the mirage effect see the section on the mirage
effect in the covert calibration example notebook. Also the paper
'Mirages in shallow water matched field processing' by Gerald D'Spain,
JASA June 1999.

scripts (to run in order):
runcsdm.scr  - script to generate covariance matrix and write it to cov.in using
real data
runSAGA.scr - script to run SAGA and produce data for 'a posteriori'
distribution plot
              and contour plot.
plotSAGA.scr - script to produce the 'a posteriori' distribution plot and
contour plot
              using plotsaga and contsaga in matlab.

input files:
vert.dat  - SAGA input parameter file ( options set to compute inversion )
vertC.dat - SAGA input parameter file ( options set to produce contour plot )
cov.in - input data for SAGA

results output by SAGA:
results - results of both runs in test file format
SAGAcontour.ps - contour postscript plot
SAGAscatter.ps - 'a posteriori' distribution plot (inversion plot)


directory:
fftData - subdirectory that contains the original time series
          data. Contains the matlab file compute_csdm.m to compute the
covariance matrix.

M-files:
compute_csdm.m - Matlab script to compute the covariance matrix.
write_covmat.m - Matlab script used by convert_to_saga.m to write out the
                 covariance matrix.
```

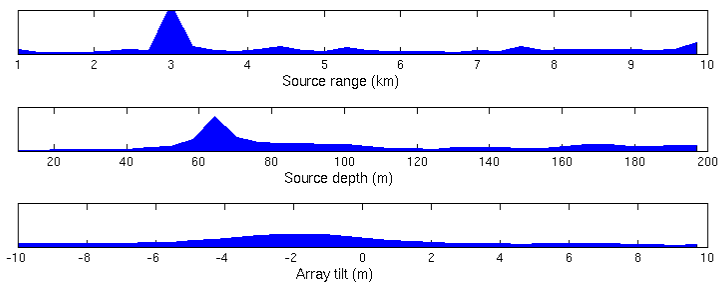

Figure 11 : 'A posteriori' distribution plot for example 5

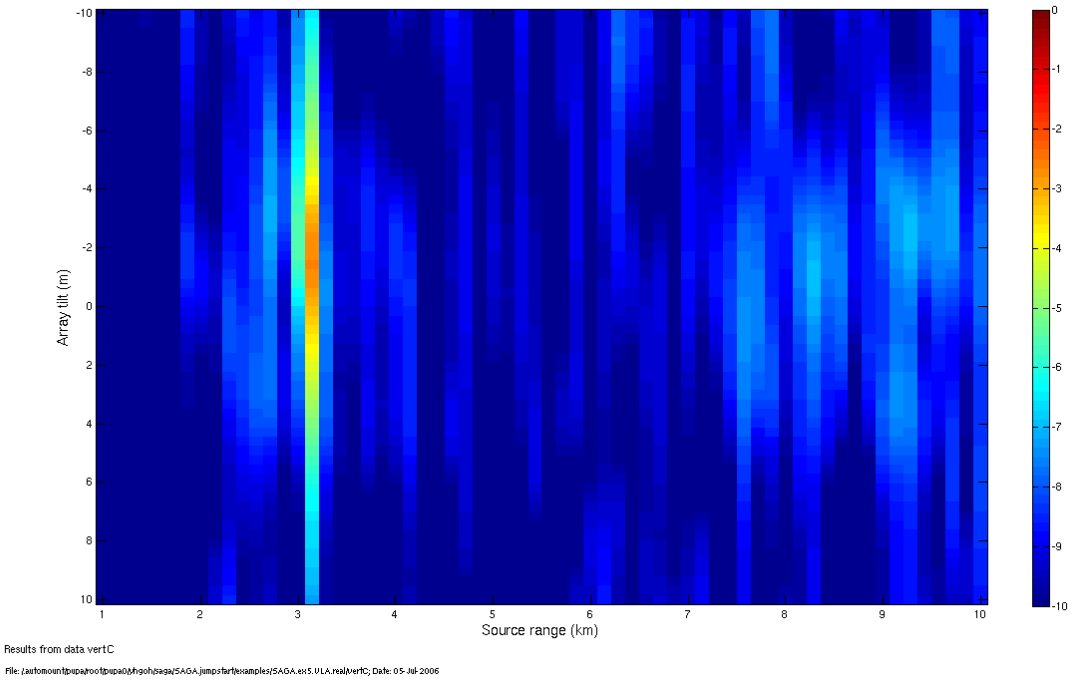

Figure 12 : Contour plot of array tilt vs source range for example 5

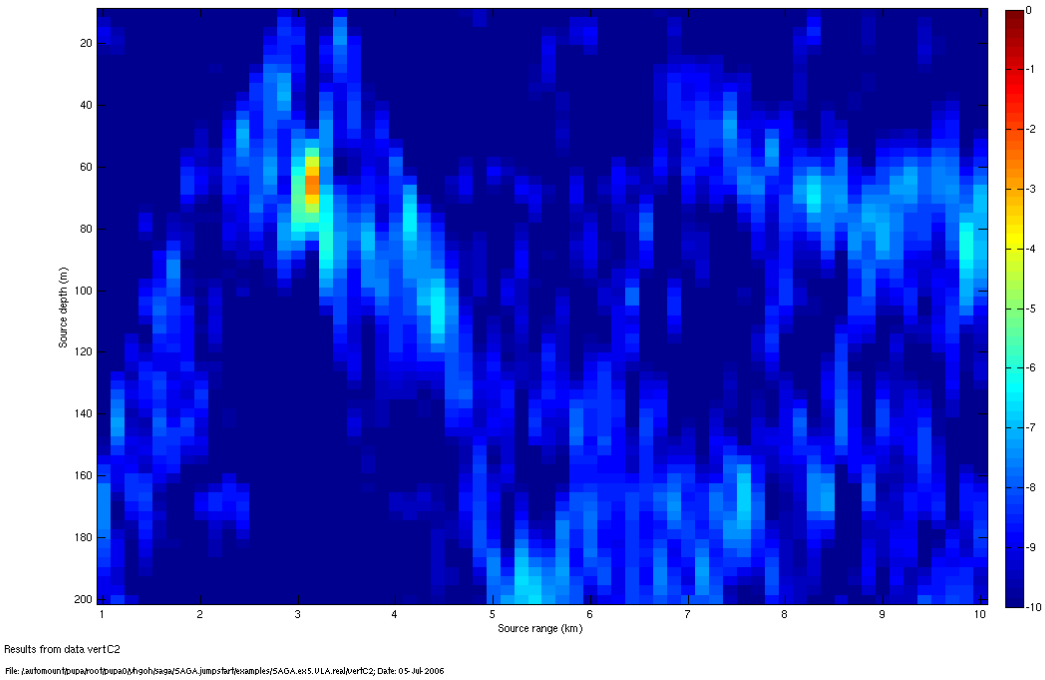

Figure 13 : Contour plot of source depth vs range example 5

Example 6: SAGA.ex6.HLA.sim

This is an example of how to do Match Field Processing using multiple frequencies with a Horizontal Line Array (simulated). This simulation matches the SWellEx96 Event S5 data set at J131 23:46. The source is at a range of 2080 km and depth of 60m. The HLA was deployed with a line of bearing of 34.5 degrees with a bow of -15 meters. The four parameters, source range, source depth, array bearing, and array bow will be searched for in this example.

Line of Bearing is calculated from Element #1 to the last element. For the HLA array Element #1 is the southernmost element.

There are a number of ways to input an array shape into SAGA, this example uses the 'x' option. So the interelement spacing is defined in the input parameter file. The array will be approximated by a parabola with the specified
interelement spacing so that bow values can be searched.

The script runSAGA.scr will first run SAGA to generate the simulated data file. Then copy the input file from a *.obs extension to a *.in extension. This is done for each input parameter file.

Now that the data is set as the input file, run SAGA to produce the inversion data using the input parameter file 'horiz.dat'. The post-processor POST is run to compute the 'a posteriori' distribution. POST also appends a text copy of the results to the file 'results'.

The script runs SAGA a second set of times using the input parameter file 'horizC.dat' to produce a contour plot of source range verses array tilt. The third set of runs of SAGA produces the source range verses source depth using the parameter file horizC2.dat. The ambiguity surfaces plotted are generated with MFP processing using only the first two parameters in the input file. Any additional search parmeters are not searched, but default to their inital value. SAGA outputs an input parameter file with these best fit values set for the initial values. This script demonstrats how to use this file to always produce ambiguity surfaces using the best fit solution results.

Matlab is run to generate the three postscript plots. Using the provided M-files in the SAGA matlab directory generate the inversion plot and the contour plots.

The inversion plot shows a panel for each of the four parameters searched.
The plot shows that source range and source depth well determined. The array azimuth and array bow both have a bimodal distribution. There is not enough information to solve for the left-right ambiguity.

The results file looks like:

```
>>m results
SAGA example 6 , HLA simulation
 ***************
  best obtainable theoretical fit  2.6959244E-08
  best fit (best, ppd, mean)  6.3673586E-02  6.3677058E-02  0.9627493

                                best-of all  most likely     mean   std-dev
  1 Source range (m)           1  2066.142     2066.142  2066.142     0.000
  2 Source depth (m)           1    60.157       60.157    60.143     0.012
  3 Array out of plane param   5   320.315       39.685   179.774     0.390
  4 Array out of plane param   2    14.961      -14.961    -0.023     0.375
```

The best fit result from SAGA does pick close to the correct azimuth and bow for the array, expecting 39.5 degrees with a -15 m bow. SAGA picked an azimuth of 320.3 degrees with a -15.0. SAGA picked the negative symetry solution over the desired solution.

The contour plots always shows the first two optimization parameters plotted with the first search parameter as the X-axis and the second search parameter as the Y-axis.

The first plot shows the source range verses source depth. Normally for a HLA to have the same accuracy as a VLA you need to have the HLA be 4x

longer then the VLA. The VLA from the previous examples is 120m long, so the HLA would need to be 480 m long to have the same quality of results. The HLA deployed was only 240 meters long. This difference is represented in the ambiguity surface in a larger target location and a peak level to background level difference that is much smaller (~2.5 dB for HLA verses closer to 5 dB for the VLA) compared to the VLA results.

The second ambiguity plot show the array azimuth verses array bow. This plot shows the results to be well determined except that SAGA does not have enough information to clearly differentiate between the true array shape and the negative symetry solution.

Summary of this directory:

scripts:
runSAGA.scr - script to run SAGA and produce 'a posteriori' distribution plot and contour plot.

input files:
horiz.dat - SAGA input parameter file (options set to compute inversion)
horizC.dat - SAGA input parameter file (generated from results of horiz.dat run)
horizC2.dat - SAGA input parameter file (generated from results of horiz.dat run)

results output by SAGA:
results - results of both runs in test file format
SAGAscatter.ps - 'a posteriori' distribution plot (inversion plot)
SAGAcontour.ps - plot of source range verses source depth
SAGAcontour2.ps - plot of array azimuth verses array bow

M-files:
write_covmat.m - Matlab script used by convert_to_saga.m to write out the covariance matrix.

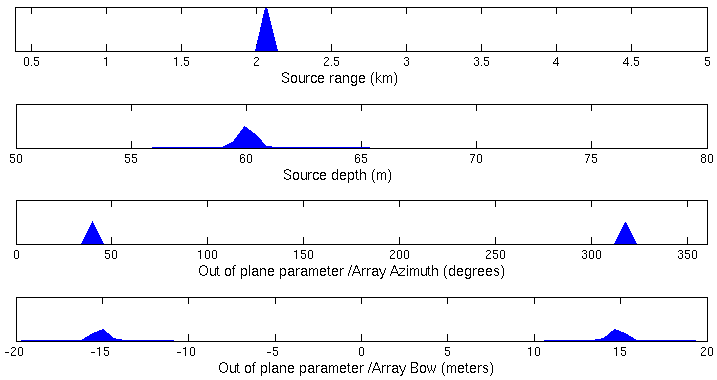

Figure 14 : 'A posteriori' distribution plot for example 6

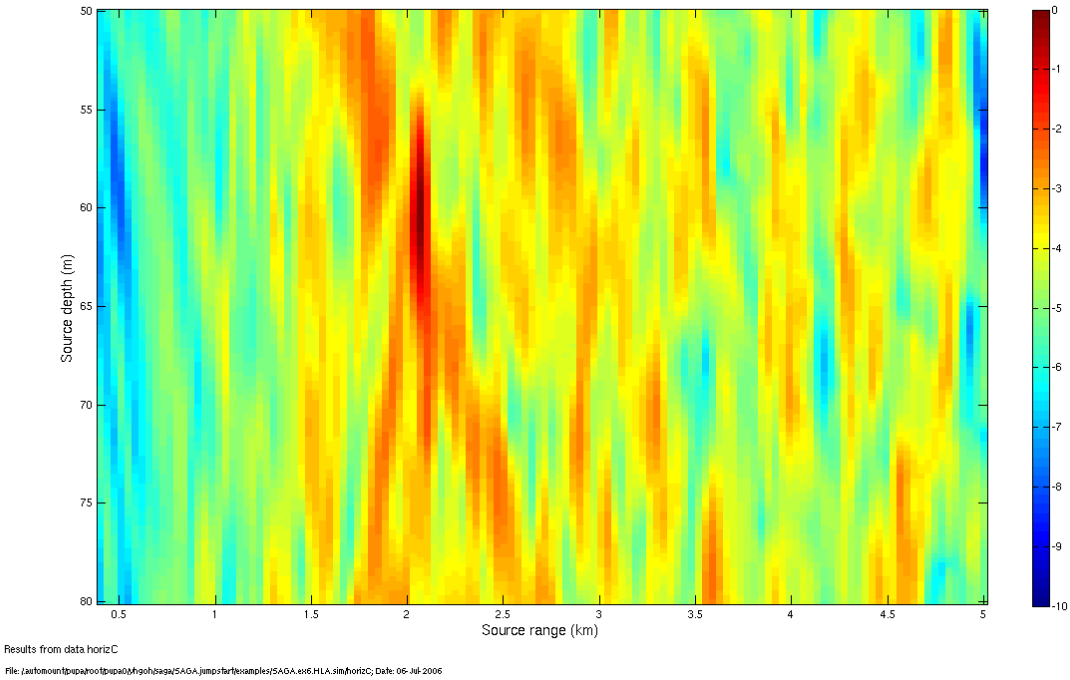

Figure 15 : Contour plot of source depth vs range for example 6

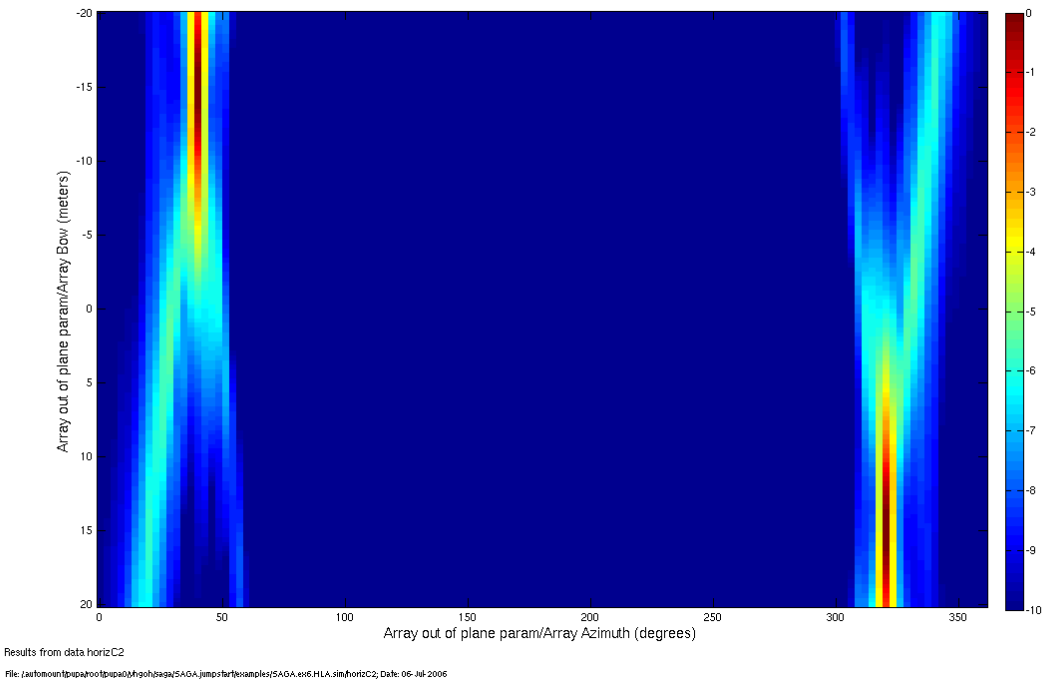

Figure 16 : Contour plot of array bow vs array azimuth for example 6

Example 7 : SAGA.ex7.HLA.real

This is an example of how to do Match Field Processing using multiple frequencies with a Horizontal Line Array using real data. The data is from SWellEx-96 J131 23:46:00. At this time the source was at a range of 2080 m and a depth of 60m. The HLA was deployed with a line of bearing of 34.5 degrees with a bow of -15 meters. These four parameters, source range, source depth, array bearing, and array bow will be searched for in this example.

Line of Bearing is calculated from Element #1 to the last element. For the HLA array Element #1 is the southernmost element. SAGA fixes the source location along the negative Y axis. Only when the source is at this location will you get the exact line of bearing. When the source is not on the -Y axis the offset between the source and the -Y axis will be represented in the the array bearing computed by SAGA. In this example the line of bearing from the source to Element 1 of the HLA is 355 degrees. This will be reflected as a 5 degree increase (to 39.5 degrees) in the array bearing computed by SAGA.

There are a number of ways to input an array shape into SAGA, this example uses the 'x' option. So the interelement spacing is defined in the input parameter file. The array will be approximated by a parabola so that bow values can be searched.

The script runSAGA.scr will first obtain the input data in a form that SAGA likes. See the ReadMe in the subdirectory 'fftData' for details on this process. Also the data file needs to be copied from the mpl website before this section is run. Read the files 'getData.txt' and 'ReadMe' in the subdirectory 'fftData' for details of how to do this.

The script runSAGA.scr first converts the CSDM data file in fftData to a SAGA formatted data file. This is done in Matlab with a SAGA provided M-file. This M-file expects to have a single input file for each frequency being processed. This M-file produces an output file called 'cov_dpss.in' The runSAGA.scr script then renames this file to be 'cov.in'. This is the input data file for SAGA. Edit the M-file 'convert_to_saga.m' to modify the list of frequencies that this M-file will process.

Now that the data is set as the input file, run SAGA to produce the inversion data using the input parameter file 'horiz.dat'. The post-processor POST is run to compute the 'a posteriori' distribution. POST also appends a text copy of the results to the file 'results'.

The script runs SAGA a second time using the input parameter file 'horizC.dat' to produce a contour plot of source range verses array tilt. The third run of SAGA produces the source range verses source depth using the parameter file horizC2.dat. The ambiguity surfaces plotted are generated with MFP processing using only the first two parameters in the input file. Any additional search parmeters are not searched, but default to their inital value. SAGA outputs an input parameter file with these best fit values set for the initial values. This script demonstrats how to use this file to always produce ambiguity surfaces using the best fit solution results.

Matlab is run to generate the three postscript plots. Using the provided M-files in the SAGA matlab directory generate the inversion plot and the contour plots.

The inversion plot shows a panel for each of the four parameters searched. The plot shows that source range well determined, and the source depth results are closely grouped between 63-65 meters . The array azimuth and array bow both have a bimodal distribution. There is not enough information to solve for the left-right ambiguity.

The results file looks like:

```
>>m results
SAGA example 7 , HLA real data example
 ***************
  best obtainable theoretical fit  4.4412155E-02
  best fit (best, ppd, mean)  0.3956003      0.4015373      0.9725764

                                best-of all  most likely     mean   std-dev
  1 Source range (m)           1  2211.024     2211.024  2232.839     0.047
  2 Source depth (m)           1    63.701       62.992    63.783     0.054
  3 Array out of plane param   5    42.520       42.520   148.695     0.372
  4 Array out of plane param   2   -13.386      -13.701    -3.188     0.320
```

The best fit result from SAGA does pick close to the correct azimuth and

bow for the array, expecting 39.5 degrees with a -15 m bow. SAGA picked an azimuth of 42.5 degrees with a -13.4. The range and source depth are again affected by the mirage effect caused by the differences in bathymetry. The water depth at the HLA is 213 m , the water depth under the source is 198 m. A difference of 7.13 %. Thus the values need to be reduced by this amount giving a range of 2211.024 / 1.0713 = 2082.7 m and for the source depth of 63.701 / 1.0713 = 59.5 m.

The contour plots always shows the first two optimization parameters plotted with the first search parameter as the X-axis and the second search parameter as the Y-axis.

The first plot shows the source range verses source depth. Normally for a HLA to have the same accuracy as a VLA you need to have the HLA be 4x longer then the VLA. The VLA from the previous examples is 120m long, so the HLA would need to be 480 m long to have the same quality of results. The HLA deployed was only 240 meters long. This difference is represented in the ambiguity surface in a larger target location and a peak level to background level difference that is much smaller (~1.5 dB for HLA verses closer to 5 dB for the VLA).

The second ambiguity plot show the array azimuth verses array bow. This plot shows the results to be well determined except that SAGA does not have enough information to clearly differentiate between the true array shape and the negative symetry solution.

Summary of this directory:

scripts:
runSAGA.scr - script to run SAGA and produce 'a posteriori' distribution plot and contour plot.

input files:
horiz.dat - SAGA input parameter file (options set to compute inversion)
horizC.dat - SAGA input parameter file (generated from results of horiz.dat run)
horizC2.dat - SAGA input parameter file (generated from results of horiz.dat run)
cov.in - input data for SAGA

results output by SAGA:
results - results of both runs in test file format
SAGAscatter.ps - 'a posteriori' distribution plot (inversion plot)
SAGAcontour.ps - contour postscript plot
SAGAcontour2.ps - contour postscript plot

directory:
csdm - subdirecotry that contains the CSDMs computed for each frequency. these files are generated from the data in the directory fftData by the script mkCSDM.scr
fftData - subdirectory containing the original time series data. Information: GetData.txt and ReadMe

M-files:
convert_to_saga.m - Matlab script to convert SIO data file format CSDM file to the format that SAGA can read.
write_covmat.m - Matlab script used by convert_to_saga.m to write out the covariance matrix.

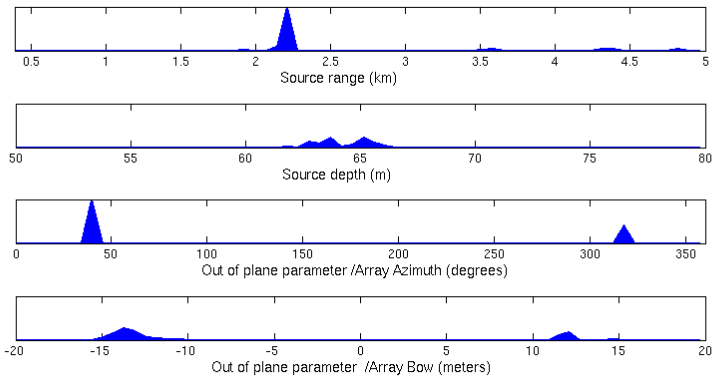

Figure 17 : 'A posteriori' distribution plot for example 7

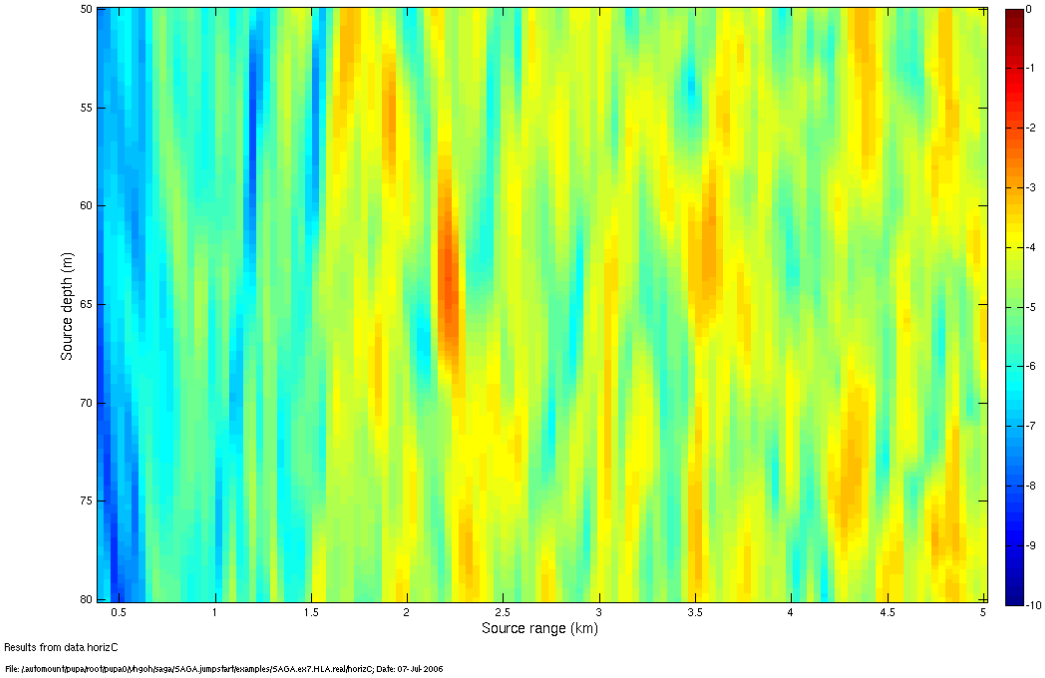

Figure 18 : Contour plot of source depth vs range for example 7

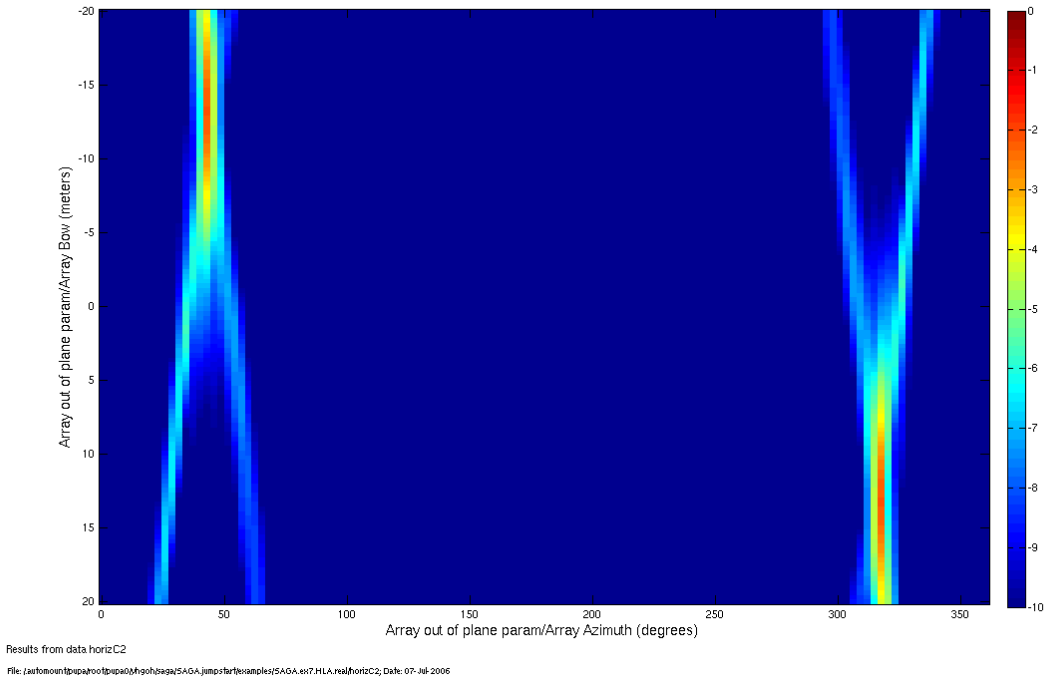

Figure 19 : Contour plot of array bow vs array azimuth for example 7