CGFD3D-elastic
User Manual

February 5, 2022

Contents

Chapter 1

Grid

run_test.sh

```
"#grid_generation_method" : {
    "#import" : "$GRID_DIR",
    "#cartesian" : {
      "origin"  : [0.0, 0.0, -5900.0 ],
      "inteval" : [ 100.0, 100.0, 100.0 ]
    },
    "layer_interp" : {
      "in_grid_layer_file" : "$EXEC_DIR/test/test_grid.gdlay",
      "refine_factor" : [ 1, 1, 1 ],
      "horizontal_start_index" : [ 3, 3 ],
      "vertical_ToFreeSurf_resample_index" : 0
    }
},
"is_export_grid" : 1,
"grid_export_dir"   : "$GRID_DIR",
```

In the configuration of grid, we can choose different grid generation methods.

(1) if `import` is selected, we can directly import the grid file from the outside. `$GRID_DIR` is the file path and name.

(2) if `cartesian` is selected, we can get the grid through interpolation with several given control interfaces. we can generate grids in Cartesian coordinates. Two parameters are required here, `origin` is the starting coordinate of the grid and `inteval` is the grid spacing.

(3) if `layer_interp` is selected, we can get the grid through interpolation with several given control interfaces. there need to be a .gdlay file, and the file format is shown in Section 1.1.
`refine_factor` is resampling control parameters for interface grids. This parameter must be an integer and cannot be zero. eg: If the parameter is 1, it means no resampling; If this parameter is `-2` or `-3` , it means two or three times downsampling; If this parameter is `2` or `3` , it means two or three times interpolating. Its value is greater than or equal to the number of ghost points
`horizontal_start_index` and `vertical_ToFreeSurf_resample_index` intercept sub-models from the interface control model.
`horizontal_start_index` is x and y start index of interface before resampling. Note: ensure that after resampling, three points must be set aside as ghost points outside the sub-model. eg: If you do not resample, the minimum value of this parameter is 3; If you do two or three times downsampling, the minimum value of this parameter is 6 or 10; If you do two or three times interpolating, the minimum value of this parameter is 2 or 1;

`vertical_ToFreeSurf_resample_index` is index to free surface. If this parameter is `0`, it means selecting the submodel that contains the free surface; If this parameter is `40`, it means selecting the submodel that intercepted at 40 nodes from the free surface in the resampled interface control model.

1.1 Layer Interp

The **layer_interp** function interpolates the spatial grid of the model through the given interface.

1.1.1 File Format (.gdlay)

The following description ignores comment lines and blank lines.

- The first line is the number of interface.
- The second line is the cells of each layer.
- The third line is isequal of each layer.
- The fourth line is number of interface grids in X and Y directions.
- Each subsequent line is the X, Y and Z coordinates of the grids.

1.1.2 Example

A model with a horizontal interface can be given as:

test.gdlay

```
  4
  20    10    20
  1       0     1
 100  100
    100.0    100.0  -5000.0
    200.0    100.0  -5000.0
    300.0    100.0  -5000.0
    400.0    100.0  -5000.0
     ...      ...      ...
```

The given interface model have 4 interfaces.
There are 20 cells between the first interface and the second interface; there are 10 cells between the second interface and the third interface; there are 20 cells between the third interface and the fourth interface. The first interface is the deepest underground interface, and the last is the free surface interface.
The cell spacing between the first interface and the second interface are equal; the cell spacing between the third interface and the fourth interface are equal; the cell spacing between the second interface and the third interface is gradual.
The number of interface grids in X is 100 and the number of interface grids in Y is 100. The number of interface grids in Z is 51 (4+(20-1)+(10-1)+(20-1)).

The given interface model size is `100*100*51` . In fact, the largest model area we can calculate is `(100-2` × `fdx_nghosts)*(100-2fdy_nghosts)*(51-fdz_nghosts)`. (In this program, these parameters `fdx_nghosts, fdy_nghosts, fdz_nghosts` are colonization 3). So the largest model area we can calculate is `94*94*48`.

(1) Example of the maximum area we can calculate without resampling:

```
number_of_total_grid_points_x = 94,
number_of_total_grid_points_y = 94,
number_of_total_grid_points_z = 48,
```

run_test.sh

```
"layer_interp" : {
  "in_grid_layer_file" : "$EXEC_DIR/test/test_grid.gdlay",
  "refine_factor" : [ 1, 1, 1 ],
  "horizontal_start_index" : [ 3, 3 ],
  "vertical_ToFreeSurf_resample_index" : 0
}
```

the calculation area index range of x is 3->96,
the calculation area index range of y is 3->96,
the calculation area index range of z is 3->50.
In the x direction, ghosts points is 0->2, and 97->99,
in the y direction, ghosts points is 0->2, and 97->99,
In the z direction, ghosts points is 0->2.

(2) Example of the partial area we can calculate without resampling:
number_of_total_grid_points_x = 50,
number_of_total_grid_points_y = 40,
number_of_total_grid_points_z = 25.

run_test.sh

```
"layer_interp" : {
  "in_grid_layer_file" : "$EXEC_DIR/test/test_grid.gdlay",
  "refine_factor" : [ 1, 1, 1 ],
  "horizontal_start_index" : [ 20, 35 ],
  "vertical_ToFreeSurf_resample_index" : 10
}
```

the calculation area index range of x is 20->69,
the calculation area index range of y is 35->74,
the calculation area index range of z is 16->40.
In the x direction, ghosts points is 17->19, and 70->72,
in the y direction, ghosts points is 32->34, and 75->77,
In the z direction, ghosts points is 13->15, and 41->42.

We provide a more complex model in the example/prep_grid directory.

Chapter 2

How to give media parameters and run wave simulations for different media types

par.json

```
"medium" : {
    "type" : "elastic_iso",
    "#code" : 1,
    "infile_layer" : "/home/usr1/testCGFD3D/prep_medium/basin.md3lay",
    "#infile_grid" : "/home/usr1/testCGFD3D/prep_medium/basin.md3grd",
    "#equivalent_medium_method" : "har",
    "#import" : "/home/usr1/testCGFD3D/output/media"
},
"is_export_media" : 1,
"media_export_dir"  : "/home/usr1/testCGFD3D/output/media",

"#visco_config" : {
    "type" : "graves_Qs",
    "Qs_freq" : 1.0
}
```

- `type` is the flag of the solver, that is, what media do you want to simulate, and four options are supported in this code:
  - `acoustic_iso`: choose this to call the solver of acoustic wave equation of isotropic media.
  - `elastic_iso`: choose this to call the solver of the elastic wave equation of isotropic media.
  - `elastic_vti`: choose this to call the solver of the elastic wave equation of vertical transversely isotropic media.
  - `elastic_aniso`: choose this to call the solver of the elastic wave equation of anisotropic media.

  The media type is given in `import`, `code`, `infile_layer` or `infile_grid`.
- The media can be given in four ways:
  - `code`: Choose this flag if you want to give the media by code. You should edit **forward/md_t.c** file and recompile the code.
  - `infile_layer`: If you want to give media by the intereface line, this option should be helpful. The media parameters are given by the **md3lay** file, and the file format is shown in Section 2.2
  - `infile_grid`: If you want to give media by a given grid media, this option should be helpful. The media parameters are given by the **md3grd** file, the file format is shown in Section 2.1.

  - `import`: If you already generated media from the above three options, and do not want to re-generate media, select this option and give the folder of the exported media.
- `equivalent_medium_method`: If you give media by `infile_layer` or `infile_grid`, equivalent medium parameterization methods can be applied. For different media, we provide different equivalent medium parameterization methods, please see Section 2.3 for detail.
- `is_export_media` should be **1** if you want to export the media, and a **media_export_dir** should be given.
- The `visco_config` is selected if the media is viscoelastic, the visco_type and Qs frequency are needed.

2.1 Format of .md3grd

If you want to get the model from the interface line, you can select the option `infile_layer`. And the media are given by a **md3lay** file. The file format is:

- The first line is the media type, it can be
  `one_component`,
  `acoustic_isotropic`,
  `elastic_isotropic`,
  `elastic_vti_prem`,
  `elastic_vti_thomsen`,
  `elastic_vti_cij`,
  `elastic_tti_thomsen`,
  `elastic_tti_bond`,
  `elastic_aniso_cij`.
- The second line is the number of layer (`NL`), if `NL` > 1, there is a designated interface, and the equivalent medium parameterization methods can be applied across the interfaces.
- The next `NL` lines are the number of grids in the z-direction of each layer.
- Next, the information of the interface mesh is given: `NX NY MIN_X MIN_Y SPACING_X SPACING_Y`
  `NX` and `NY` are the number of points along x and y direction;
  `MIN_X` and `MIN_Y` are the minimal x and y coordinates;
  `SPACING_X` and `SPACING_Y` are spacing between points along x and y.
- After then, the elevation, media parameters are given in every grid points. Take the `elastic_isotropic` media for example, the media parameters are read as:

```
for (nl=0; nl< NL; nl++)
  for (ip=0; ip<np[nl]; np++)
    for(iy=0; iy<NY; iy++)
      for (ix=0; ix<NX;ix++)
        fscanf(in_3grd_file, "%f %f %f %f",
            &elevation, &rho, &vp, &vs);
```

  The media parameters are calculated by linear interpolation of the values at the given grid points.

  For the points higher than the interface, assigned by the uppermost medium.

  For different media type, different media parameters are given,

  - `one_component`: var.
  - `acoustic_isotropic`: ρ, vp.
  - `elastic_isotropic`: ρ, vp, vs.
  - `elastic_vti_prem`: ρ, vph, vpv, vsh, vsv, η. Please see [Dziewonski and Anderson, 1981] for detail.
  - `elastic_vti_thomsen`: ρ, α_0, β_0, ε, δ, γ. Please see [Thomsen, 1986] for detail.
  - `elastic_vti_cij`: ρ, c_{11}, c_{33}, c_{55}, c_{66}, c_{13}.
  - `elastic_tti_thomsen`: ρ, α_0, β_0, ε, δ, γ, Φ, θ. Please see [Thomsen, 1986] for detail.

- `elastic_tti_bond`: ρ, c_{11}, c_{33}, c_{55}, c_{66}, c_{13}, Φ, θ. Φ is azimuth angle, θ is the dip angle.
- `elastic_aniso_cij`: ρ, c_{11}, c_{12}, c_{13}, c_{14}, c_{15}, c_{16}, c_{22}, c_{23}, c_{24}, c_{25}, c_{26}, c_{33}, c_{34}, c_{35}, c_{36}, c_{44}, c_{45}, c_{46}, c_{55}, c_{56}, c_{66}.

if `NL` > 1, there is a designated interface; and the elevation of ng[il]+1 needs to be the same as ng[il]. The equivalent medium parameterization method can be applied on the points cutting by this interface.

You can also find the example in the **example/prep_medium** directory.

2.2 Format of .md3lay

If you want to get the model from the interface line, you can select the option `infile_layer`. And the media are given by a **md3lay** file. The file format is:

- The first line is the media type, it can be
  `one_component`,
  `acoustic_isotropic`,
  `elastic_isotropic`,
  `elastic_vti_prem`,
  `elastic_vti_thomsen`,
  `elastic_vti_cij`,
  `elastic_tti_thomsen`,
  `elastic_tti_bond`,
  `elastic_aniso_cij`.
- The second line is the number of interface (`NI`).
- The third line is the information of the given interface mesh: `NX NY MIN_X MIN_Y SPACING_X SPACING_Y`
  `NX` and `NY` are the number of points along x and y direction;
  `MIN_X` and `MIN_Y` are the minimal x and y coordinates;
  `SPACING_X` and `SPACING_Y` are spacing between points along x and y.
- After then, the elevation, media parameters are given in every interface points. Take the `elastic_isotropic` media for example, the media parameters are read as:

```
for (ni=0; ni<NI; ni++)
  for (iy=0; iy<NY; iy++)
    for (ix=0; ix<NX; ix++)
      fscanf(in_3lay_file, "%f %f %f %f %f %f %f",
             &elevation,
             &rho, &rho_grad, &rho_pow,
             &vp, &vp_grad, &vp_pow,
             &vs, &vs_grad, &vs_pow);
```

  `*_grad` and `*_pow` is the gradient and power of the media parameters in z-direction. The media parameters below interface are calculated by

$$var^{grid\ point} = var + (z\_\text{interface} - z^{grid\ point})^{var\_pow} * var\_grad.$$

For the points higher than the interface, assigned by the uppermost medium.

For different media type, different media parameters and the corresponding `*_grad` and `*_pow` are given, the media parameters for different media can be found in Section 2.1

You can also find examples in the **example/prep_medium** directory.

2.3 Note about equivalent medium parameterization methods

When there are strong interfaces in the model, using grid points to discrete models directly may cause interface error and the generation of artificial diffraction from stair step interfaces. In this code, we provide some equivalent medium parametrization methods to reduce this interface error.

If you give the model by `infile_layer` or `infile_grid`, you can apply different equivalent medium parametrization methods by give different `equivalent_medium_method`.

If the media type in the **md3grd** or **md3lay** file is `one_component`, `equivalent_medium_method` can be

- `loc`: using local value to discrete model.
- `ari`: volume integral arithmetic average.

$$\langle var \rangle = \frac{1}{\Delta V} \int_{k-1/2}^{k+1/2} \int_{j-1/2}^{j+1/2} \int_{i-1/2}^{i+1/2} var \; dxdydz \tag{2.1}$$

- `har`: volume integral harmonic average,

$$\langle var \rangle^H = \frac{\Delta V}{\int_{k-1/2}^{k+1/2} \int_{j-1/2}^{j+1/2} \int_{i-1/2}^{i+1/2} \frac{1}{var} dxdydz} \tag{2.2}$$

If the media type is `acoustic_isotropic`, `equivalent_medium_method` can be

- `loc`: using local media parameters to discrete model.
- `har`: applying harmonic average to κ, and applying arithmetic average to density ρ.
- `ari`: applying arithmetic average to κ and ρ.

If the media type is `elastic_isotropic`, `equivalent_medium_method` can be

- `loc`: using local media parameters to discrete model.
- `har`: applying harmonic average to elastic modulus, and applying arithmetic average to density ρ. Please see Moczo et al. [2002] and [Moczo et al., 2014] for detail.
- `ari`: applying arithmetic average to elastic modulus and density.

If the media type is `elastic_vti_*`, `equivalent_medium_method` can be

- `loc`: using local media parameters to discrete model.
- `har`: applying harmonic average to elastic modulus c_{ij}, and applying arithmetic average to density ρ.
- `ari`: applying arithmetic average to elastic modulus c_{ij} and density.

If the media type is `elastic_aniso_cij` or `elastic_tti_*`, `equivalent_medium_method` can be

- `loc`: using local media parameters to discrete model.
- `har`: applying harmonic average to elastic modulus c_{ij}, and applying arithmetic average to density ρ.
- `ari`: applying arithmetic average to elastic modulus c_{ij} and density.

Chapter 3

Input Source into the Simulation

Seismic wave is actived by the seimisc source. In seismic wave equations, the seismic source is represented by the force term ($\mathbf{F}$) and/or stress gluss term ($\dot{\mathbf{M}}$) at the right-hand side (RHS) of the equations:

$$\rho \frac{\partial \mathbf{v}}{\partial t} = \nabla \cdot \sigma + \mathbf{F}, \tag{3.1}$$

$$\frac{\partial \sigma}{\partial t} = \mathbf{C} : \frac{1}{2} (\nabla \mathbf{v} + \mathbf{v} \nabla) - \dot{\mathbf{M}}. \tag{3.2}$$

The force term is used for sources acting as an external force to the simulatin region, while the stress gluss term could be used to implement a general moment tensor souce. It is relative easy to implement the source in the code by adding the respective terms onto the RHS values. We just need a way to input different combinations of how many sources, the locations, source time funcion (STF) and source mechanism for a single simulation into the code.

Different applications may require a source in different formats, e.g,

1. single force at a single point, which is a common source for Green function and exploration seismology;
2. general moment source at a single point, which is used for small earthquake and explosive source;
3. general moment sources along a fault, which is also called finite-fault source model and used for large earthquake;
4. time-reversed waveforms along surface, which is used for time reversal imaging;
5. plane wave incident for teleseismic wave problems.

The first two sources act at a single point, while the last three types are implemented by adding many single point sources simultaneously along a fault, a surface or a pre-defined line (plane). The first two types can be taken as a special case of the last three types when number of the source becomes one.

The CGFD3D package tries to implement different sources to support different applications. Thus we allow to input a single point source or many point sources. For easy usage, we also support:

- the location of the single point could be grid index or coordinate;
- the vertical coordinate could be given by the absolute axis or the depth relative the free surface (to be done!);
- STF of each single point source could be an analytical wavelet function or discrete values obtained by real data inversion or source dynamic simulation;
- the source could be a force source and/or a moment one;
- the moment source could be given by the six tensor components or the mechanism angles plus μDA.

To allow different input combinations of above different parameters, we use several flags in the input file to determine the specific input meaning of the related parameters. In the following, we will describe the format of the source input file of the CGFD3D package and give several examples to show how to input different sources.

3.1 Source related configurations in main JSON file

There are three paramters in the .json file for the source input:

- `in_source_file`: set the input source file;
- `is_export_source`: determine if export the internal discreted source for QC (not implemented yet);
- `source_export_dir`: the path for source exporting.

Currently, only `in_source_file` takes effect.

An example of source settings in .json

```
"in_source_file" : "$INPUTDIR/test.src",
"is_export_source" : 1,
"source_export_dir" : "$SOURCE_DIR",
```

3.2 Description of the source input file (.src)

The .src file is designed to input all the required source information using different representations. Considering parallel computing using MPI on a multi-nodes cluster, we want to allocate large array holdding discrete STF values for only the source located at the node. Thus we divide the information in the .src into three regions (line numnber refer that in the example):

1. global information related to all the sources (line 1-20);
2. location information of each source (line 21-25);
3. STF and component information of each source (after line 26).

In other words, we specify the global information/flags first, then list the locations of all the sources, and finally give the STF and component values of each source. The last part may be very large for discrete STF input.

A sample .src file using analytical wavelet function as STF is shown below. We will use this sample .src file to explain the file format of .src file.

Listing 3.1: Source input file using analytical wavelet

```
1  # name of this input source
2  event_1
3  # number of source
4  1
5  # flag for stf
6  #  1st value : 0 analytic stf or 1 discrete values
7  #    for analytical, 2nd value is time length
8  #    for discrete,  2nd value is dt and 3rd is nt
9  #    e.g.,
10 # 0 4.0
11 # 1 0.05 20
12 0 1.0
13 # flag for source component and mechanism format
14 #  1st value: source components, 1(force), 2(momoment), 3(force+moment)
15 #  2nd value: mechanism format for moment source: 0 moment, 1 angle + mu + D + A
16 2 0
17 # flag for location
18 #   1st value: meaning of the location: 0 computational coordinate, 1 physical
       coordinate
```

```
19 #   2nd value: 0 absolute axis or 1 depth relative to the free surface of the third
       coordinate
20 1 0
21 # location of each source
22 #   sx sy sz
23 #80 49 50
24 #80 49 50
25 8020 4930 -950
26 # stf and cmp
27 0.0 ricker 2.0 0.5   # t0  stf_name  ricker_fc ricker_t0
28 1e16  1e16  1e16 0 0 0
```

From above sample file, We see that we can insert comment line in the .src file using "#" as the line start, similar to the Bash script syntax. What we should set in the .src file are:

- name of the whole source (represented by EVTNM, following sac notation) (line 2), type should be string without whitespace. This name is used for output file naming. E.g., the output sac file will start with "event_1" for this sample file.
- number of the sources (NS) (line 4), type is interger. The second part should contain NS locations and the third part should contains NS STF and cmp settings.
- settings about STF (line 12). Two values or three values depends on the first value, which tells the code how STF is given (STF_flag):
  - 0: the STF whill be set using wavelet name and related coefficiences in the second part for each source. The second value here sets the same total time length of all the STF to reduce using computer memory to hold discrete STF in simulation.
  - 1: the STF will be given as discrete values. Then the second value is the time step (stf_dt) and the third one is the total number of time step (stf_nt).

  We should note that in the third part, we will give each source a separte start time to implement finite-fault source combining above time window setting.
- two values to set force and/or moment type of each source, and how the moment source is given (line 16). The first value (CMP_flag):
  - 1: force;
  - 2: moment;
  - 3: force and momemnt.

  The second value (Mechanism_flag) determines how the mechanism is given if moment source is used:
  - 0: values of the six components of the moment tensor;
  - 1: strke, dip and rake angels, plus μ, D, and A.
- two flags about the meaning of the location values (line 20). The first flag is the meaning of the location (LOC_flag),
  - 0: the location is given by the computational coordinate. The value is same to the grid index in this implementation. The decimal part is the relative shift of the source to the grid point;
  - 1: the location is given as the physical coordinate values, which is the Cartesian coordinate in this implementation.

  The second flag (Vertical_flag) is used when the first flag set 1 to determine the third coordinate is
  - 0: absolute z-axis value;
  - 1: depth relative to the free surface (need to be implemented!).
- Then the second part, the locations of each sources (line 22-25). There should be NS lines for NS sources. Each line contains three values representing the location of one source. If LOC_flag = 0 (computational coordinate), the location should be given by grid index with decimal as

Listing 3.2: Source location by computational coordinate

```
1 # location of each source (e.g., for NS = 3)
```

```
20.0 20.0 50.0
30.0 29.0 50.0
40.5 50.2 55.1
```

If LOC_flag = 1 (physical coordinate), the location should be given by its coordinate values as

Listing 3.3: Source location by physical coordinate

```
# location of each source (e.g., for NS = 3)
2000.0 2000.0 5000.0
3000.0 2900.0 5000.0
4050.0 5020.0 5510.0
```

- The third part specifies STF and component values of each source (after line 26). The formats are different for different STF_flag.
  If STF_flag = 0 (wavelet name), each source has two lines,

  1. two or more values to set the STF:
     - first value: the activated (or start) time, which is important for finite-fault model to represent fault rupture.
     - second value: a string, the wavelet function name. The valid wavelet names for current CGFD3D are shown in table 3.1 (more wavelent functions will be implemented in the future). You can easily add a wavelet by modifying function `src_cal_wavelet` in src_t.c.
     - 3rd-12th values: the coefficients required by the wavelet function. Different wavelet may require different number of coefficients. Here we allow up to maximum 10 coefficients to specify the values. It will no problem only setting one or two values if the wavelet only needs one or two coefficients. Please see table 3.1 for the number and meanings of coefficients of current supported wavelets.
  2. 3, 6 or 9 values (depending on CMP_flag) to give the magnitudes of the force and/or moment components.
     - CMP_flag=1: 3 values to set values of F_x F_y and F_z;
     - CMP_flag=2: 6 values to set values of the moment tensor, ordered as
       - $m_{xx}, m_{yy}, m_{zz}, m_{yz}, m_{xz}, m_{xy}$ following the Viogt nonation (Fig 3.1) if Mechanism_flag=0
       - strike, dip, rake, μ, D, A if Mechanism_flag=1.
     - CMP_flag=3: 9 values to set values of both the force and the moment tensor, ordered as
       - $F_x, F_y, F_z, m_{xx}, m_{yy}, m_{zz}, m_{yz}, m_{xz}, m_{xy}$ if Mechanism_flag=0
       - F_x, F_y, F_z, strike, dip, rake, μ, D, A if Mechanism_flag=1.

  If STF_flag = 1 (discrete values), each source has (1+stf_nt) lines (see sample Listing 3.4),

  1. first line: one value, the activated (or start) time.
  2. stf_nt lines of 3, 6 or 9 values (depending on CMP_flag) to give the magnitudes of the force and/or moment components at each time step:
     - CMP_flag=1: 3 values of F_x F_y and F_z;
     - CMP_flag=2: 6 values of the moment tensor, ordered as
       - $m_{xx}, m_{yy}, m_{zz}, m_{yz}, m_{xz}, m_{xy}$ following the Viogt nonation (Fig 3.1) if Mechanism_flag=0
       - strike, dip, rake, μ, D, A if Mechanism_flag=1.
     - CMP_flag=3: 9 values of both the force and the moment tensor, ordered as
       - $F_x, F_y, F_z, m_{xx}, m_{yy}, m_{zz}, m_{yz}, m_{xz}, m_{xy}$ if Mechanism_flag=0
       - F_x, F_y, F_z, strike, dip, rake, μ, D, A if Mechanism_flag=1.

Listing 3.4: Source input file using discrete STF

```
# name_of_this_input_src
evt_1_stf_value
# number of source
1
# meaning of the location: 0 computational coordinate, 1 physical coordinate
#     axis or depth of the third coordinate: 0 axis, 1 depth
```

Table 3.1: Implemented Wavelet functions and their coefficients.

wavelet name	coef[0]	coef[1]
ricker	cetral frequency	central time shift
ricker_deriv	cetral frequency	central time shift
gaussian	RMS value	time shift
gaussian_deriv	RMS value	time shift

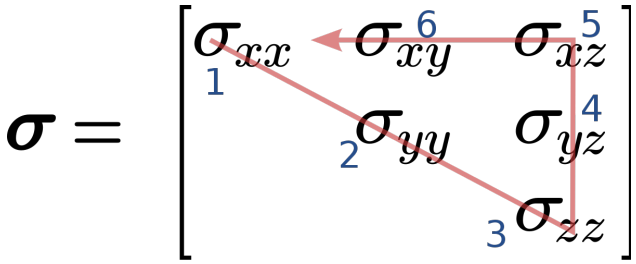

Figure 3.1: Viogt notation (From: Wiki)

```
0 1
# stf_input_type and time length info
#  0 4.0 : analytic and time window length of each stf
#  1 0.05 20  : 1 value and time_step num_of_step
1 0.025 40
# 1(force) 2(momoment) 3(force+moment)
#    mechanism type for moment source: 0 moment, 1 angle + mu + D + A
2 0
# meta data of each source
#   sx sy sz
80 49 50
# value data for each source
#  t0
0.0
# Mxx etc
4.29488e+12    4.29488e+12   4.29488e+12     0.0  0.0  0.0
1.0088e+13     1.0088e+13    1.0088e+13      0.0  0.0  0.1
2.24061e+13    2.24061e+13   2.24061e+13     0.0  0.0  0.0
4.70162e+13    4.70162e+13   4.70162e+13     0.0  0.0  0.0
9.31041e+13    9.31041e+13   9.31041e+13     0.0  0.0  0.0
1.73751e+14    1.73751e+14   1.73751e+14     0.0  0.0  0.0
3.05036e+14    3.05036e+14   3.05036e+14     0.0  0.0  0.0
5.02611e+14    5.02611e+14   5.02611e+14     0.0  0.0  0.0
7.7483e+14     7.7483e+14    7.7483e+14      0.0  0.0  0.0
1.11265e+15    1.11265e+15   1.11265e+15     0.0  0.0  0.0
1.47863e+15    1.47863e+15   1.47863e+15     0.0  0.0  0.0
1.79991e+15    1.79991e+15   1.79991e+15     0.0  0.0  0.0
1.97157e+15    1.97157e+15   1.97157e+15     0.0  0.0  0.0
1.87557e+15    1.87557e+15   1.87557e+15     0.0  0.0  0.0
1.41548e+15    1.41548e+15   1.41548e+15     0.0  0.0  0.0
5.58831e+14    5.58831e+14   5.58831e+14     0.0  0.0  0.0
-6.2831e+14    -6.2831e+14   -6.2831e+14     0.0  0.0  0.0
-1.97263e+15   -1.97263e+15  -1.97263e+15    0.0  0.0  0.0
-3.22222e+15   -3.22222e+15  -3.22222e+15    0.0  0.0  0.0
-4.1098e+15    -4.1098e+15   -4.1098e+15     0.0  0.0  0.0
-4.43113e+15   -4.43113e+15  -4.43113e+15    0.0  0.0  0.0
-4.1098e+15    -4.1098e+15   -4.1098e+15     0.0  0.0  0.0
```

```
-3.22222e+15   -3.22222e+15  -3.22222e+15    0.0  0.0  0.0
-1.97263e+15   -1.97263e+15  -1.97263e+15    0.0  0.0  0.0
-6.28308e+14   -6.28308e+14  -6.28308e+14    0.0  0.0  0.0
5.58831e+14    5.58831e+14   5.58831e+14     0.0  0.0  0.0
1.41548e+15    1.41548e+15   1.41548e+15     0.0  0.0  0.0
1.87557e+15    1.87557e+15   1.87557e+15     0.0  0.0  0.0
1.97157e+15    1.97157e+15   1.97157e+15     0.0  0.0  0.0
1.79991e+15    1.79991e+15   1.79991e+15     0.0  0.0  0.0
1.47863e+15    1.47863e+15   1.47863e+15     0.0  0.0  0.0
1.11265e+15    1.11265e+15   1.11265e+15     0.0  0.0  0.0
7.7483e+14     7.7483e+14    7.7483e+14      0.0  0.0  0.0
5.02611e+14    5.02611e+14   5.02611e+14     0.0  0.0  0.0
3.05036e+14    3.05036e+14   3.05036e+14     0.0  0.0  0.0
1.73751e+14    1.73751e+14   1.73751e+14     0.0  0.0  0.0
9.3104e+13     9.3104e+13    9.3104e+13      0.0  0.0  0.0
4.70162e+13    4.70162e+13   4.70162e+13     0.0  0.0  0.0
2.24061e+13    2.24061e+13   2.24061e+13     0.0  0.0  0.0
1.0088e+13     1.0088e+13    1.0088e+13      0.0  0.0  0.0
```

Copyright

Main historical authors:

Evolution of the code:

Bibliography

A. M. Dziewonski and D. L. Anderson. Preliminary reference earth model. *Physics of the earth and planetary interiors*, 25(4):297–356, 1981.

P. Moczo, J. Kristek, V. Vavrycuk, R. J. Archuleta, and L. Halada. 3D heterogeneous staggered-grid finite-difference modeling of seismic motion with volume harmonic and arithmetic averaging of elastic moduli and densities. *Bulletin of the Seismological Society of America*, 92(8):3042–3066, 2002.

P. Moczo, J. Kristek, and M. Gális. *The finite-difference modelling of earthquake motions: waves and ruptures*. Cambridge University Press, New York, 2014. ISBN 978-1-107-02881-4.

L. Thomsen. Weak elastic anisotropy. *Geophysics*, 51(10):1954–1966, 1986.