CGFD3D-elastic
User Manual

October 22, 2021

Contents

Chapter 1

Media

run_test.sh

```
"media_input" : {
    "#import" : "$MEDIA_DIR",
    "#code_generate" : 1,
    "#in_3lay_file" : "${IN_MEDIA_3LAY_FILE}", # *.md3lay
    "in_3grd_file" : "${IN_MEDIA_3GRD_FILE}",  # *.md3grd
    "equivalent_medium_method" : "har",       # default: "loc"
},
"is_export_media" : 1,                         # if export media
"media_export_dir"  : "$MEDIA_DIR",
```

In the configuration of media, if `in_3lay_file` is selected, there need to be a .md3lay file, and the file format is shown in Section 1.1. If `in_3grd_file` is selected, there need to be a .md3grd file, and the file format is shown in Section 1.2.

1.1 Layer to Model

The **media_el_iso_layer2model** function is used to discretize the given layer model to the grid model. It provides two medium parameterization method: using the local point values (`loc`), and volume arithmetic and harmonic averaging method (`har`) [**??**].

1.1.1 File Format (.md3lay)

3D Layer Velocity Model File (`.md3lay`)

The following description ignores comment lines and blank lines.

- The first line is the number of interface (`NI`).
- The second 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, velocity and density are given as:

```
for (ni=0; ni<NI; ni++)
  for (iy=0; iy<NY; iy++) {
    for (ix=0; ix<NX; ix++) {
      fscanf(layer_file, "%f %f %f %f %f %f %f",
             &elevation[ni][iy][ix],
             &vp[ni][iy][ix], &vp_grad[ni][iy][ix],
             &vs[ni][iy][ix], &vs_grad[ni][iy][ix],
             &rho[ni][iy][ix], &rho_grad[ni][iy][ix]);
    }
  }
}
```

For each interface (from the free surface to bottom), a set of elevation values (elevation), v_p (vp), the gradient of v_p (vp_grad), v_s (vs), the gradient of v_s (vs_grad), ρ (rho) and the gradient of ρ (rho_grad) on the regular 2D grid is required.

The velocities and density below interface(x, y, elevation) are calculated by

$$v_p^{grid\ point} = vp + (elevation - z^{grid\ point}) * vp\_grad.$$

1.1.2 Example

A model with a horizontal interface can be given as:

test.md3lay

```
# NI
  2
# NX   NY   MIN_X  MIN_Y  SPACING_X  SPACING_Y
  2    2    0.0    0.0    2000.0     2000.0
# elevation   vp   vp_grad   vs   vs_grad   rho   rho_grad
# interface #1 free surface
    0.0    2500.0  0.0  1500.0  0.0  1500.0  0.0
    0.0    2500.0  0.0  1500.0  0.0  1500.0  0.0
    0.0    2500.0  0.0  1500.0  0.0  1500.0  0.0
    0.0    2500.0  0.0  1500.0  0.0  1500.0  0.0
# interface #2
 -1000.0   4000.0  0.0  2400.0  0.0  2400.0  0.0
 -1000.0   4000.0  0.0  2400.0  0.0  2400.0  0.0
 -1000.0   4000.0  0.0  2400.0  0.0  2400.0  0.0
 -1000.0   4000.0  0.0  2400.0  0.0  2400.0  0.0
```

We provide a more complex model in the `test/` directory.

1.2 Grid to Model

The **media_el_iso_grid2model** function is used to discretize the given grid model to the grid model. It also provides two medium parameterization method: using the local point values (`loc`), and volume arithmetic and harmonic averaging method (`har` in) [**??**].

1.2.1 File Format (.md3grd)

The following description ignores comment lines and blank lines.

- The first line is the number of layer (`NL`), if `NL` > 1, there is a designated interface.
- the next `NL` lines are the number of grids in the z-direction of each layer

- 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, velocity and density are given in every grid points:

```
for (ig=0; ig<ng_z; ig++)
  for (iy=0; iy<NY; iy++) {
    for (ix=0; ix<NX; ix++) {
      fscanf(grid_file, "%f %f %f %f", &elevation[ig][iy][ix],
            &vp[ig][iy][ix], &vs[ig][iy][ix], &rho[ig][iy][ix]);
    }
  }
}
```

The velocities and density are calculated by interpolation of the values at the given grid points.

1.2.2 Example

A model with a horizontal interface can be given as:

test.md3lay

```
# NL
  2
# How many z-grids are in each layer
  2
  2

# NX    NY    MIN_X    MIN_Y    SPACING_X    SPACING_Y
  2     2     0.0      0.0      2000.0       2000.0
# elevation    vp       vs       rho
# z-grid #1: Top - free surface
    0.0      2500.0   1500.0   1500.0
    0.0      2500.0   1500.0   1500.0
    0.0      2500.0   1500.0   1500.0
    0.0      2500.0   1500.0   1500.0
# z-grid #2
 -1000.0     2500.0   1500.0   2400.0
 -1000.0     2500.0   1500.0   2400.0
 -1000.0     2500.0   1500.0   2400.0
 -1000.0     2500.0   1500.0   2400.0
# z-grid #3 (the elevation needs to be the same as #2)
 -1000.0     4000.0   2400.0   2400.0
 -1000.0     4000.0   2400.0   2400.0
 -1000.0     4000.0   2400.0   2400.0
 -1000.0     4000.0   2400.0   2400.0
# z-grid #4
 -2000.0     4000.0   2400.0   2400.0
 -2000.0     4000.0   2400.0   2400.0
 -2000.0     4000.0   2400.0   2400.0
 -2000.0     4000.0   2400.0   2400.0
```

if `NL` > 1, there is a designated interface; and the elevation of the ng[il]+1 needs to be the same as ng[il]. The equivalent medium parameterization method can be applied on this interface. We provide a more complex model in the `test/` directory.

Chapter 2

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 2.1.
`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.

2.1 Layer Interp

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

2.1.1 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.

2.1.2 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 3

Source

run_test.sh

```
    "source_input" : {
        "in_par" : {
            "name" : "evt_by_par"
            "source" : [
                {
                    "#index" : [40, 50, 30],
                    "coord" : [4000, 5000, -2900],
                    "wavelet_name" : "ricker",
                    "ricker_center_frequency" : 0.5,
                    "ricker_peak_time" : 2.0,
                    "#wavelet_name" : "gaussian",
                    "#gaussian_rms_width" : 2.0,
                    "#gaussian_peak_time" : 0.5,
                    "start_time" : 0.0,
                    "end_time" : 4.0,
                    "force_vector" : [1.0e13, 1.0e10, 1.0e11],
                    "moment_tensor" : [1.2e13, 1.0e10, 1.0e14, 1.0e12, 1.0e11, 1.0e12]
                },
                {
                    "#index" : [30, 20, 30],
                    "coord" : [3000, 3000, -2900],
                    "wavelet_name" : "ricker",
                    "ricker_center_frequency" : 0.5,
                    "ricker_peak_time" : 2.0,
                    "#wavelet_name" : "gaussian",
                    "#gaussian_rms_width" : 2.0,
                    "#gaussian_peak_time" : 0.5,
                    "start_time" : 0.0,
                    "end_time" : 4.0,
                    "force_vector" : [1.0e13, 1.0e10, 1.0e11],
                    "moment_tensor" : [1.2e13, 1.0e10, 1.0e14, 1.0e12, 1.0e11, 1.0e12]
                }
            ]
        },
        "#in_source_file" : "$INPUTDIR/source.anasrc"   #could choose source.valsrc
  },
  "is_export_source" : 1,                            # if export source
  "source_export_dir"  : "$SOURCE_DIR",
```

3.1 JSON read source format

In the configuration of source, if `in_par` is selected, that means the source information is read by JSON. `index` is source grid index location and `coord` is source distance location, Input order is x, y, z. Select one of the two representations to determine the location of the source. `wavelet_name` include ricker, ricker_deriv, gaussian, gaussain_deriv. if wavelet_name choose ricker or ricker_deriv, should give `ricker_center_frequency` and `ricker_peak_time`. if wavelet_name choose gaussian or gaussian_deriv, should give `gaussian_rms_width` and `gaussian_peak_time`. `start_time` is source function start time, and `end_time` is source function end time. `force_vector` is single force source component, the input order is Fx, Fy, Fz. `moment_tentor` is double couple source component, the input order is Mxx, Myy, Mzz, Myz, Mxz, Mxy. `force_vector` and `moment_tentor` could choose together. if `in_source_file` is selected, there need to be a .anasrc file or .valsrc file, and the .anasrc file format is shown in Section 3.2. The .valsrc file format is shown in Section 3.3.

3.2 analytic source format

A .anasrc file can be given as:

source.anasrc

```
test_event_1
3 2
4.0
3000.0 2000.0 -3000.0
1.0e15 1.0e13 1.0e14
ricker
2.0 0.5
0.0
1000.0 2000.0 -3000.0
1.0e15 1.0e13 1.0e14
gaussian
2.0 0.5
1.0
1000.0 2000.0 -3000.0
1.0e15 1.0e13 1.0e14
gaussian_deriv
1.0 5.0
1.0
5000.0 2000.0 -2000.0
moment_tensor
1.0e15 1.0e13 1.0e14 1.0e15 1.0e13 1.0e14
gaussian_deriv
1.0 5.0
0.0
1500.0 2200.0 -1300.0
mechanism_angle
80.0 70.0 30.0 1.1e10 1.1 1.0e8
ricker_deriv
2.0 0.5
2.0
```

first line `test_event_1` is source name. second line `3 2` is force source number 3 and moment source number 2. third line `4.0` is source function time length, be equal to end_time subtract start_time. All source functions have the same time length. Next lines, the information of each source is given in order.

A force source, first line give coordinates `3000.0 2000.0 -3000.0`, input order is x, y, z. Second line give force_vector `1.0e15 1.0e13 1.0e14`, input order is Fx, Fy, Fz. Third line is wavelet name, include ricker, ricker_deriv, gaussian, gaussian_deriv. The first force source wavelet name is ricker. Fourth line is ricker_center_frequency and ricker_peak_time, respectively, if wavelet name is ricker or ricker_deriv. Fourth line is gaussian_rms_width and gaussian_peak_time, respectively, if wavelet name is gaussian or gaussian_deriv. The first force source ricker_center_frequency is 2.0, ricker_peak_time is 0.5. The fifth line is source function start time, The first force source start time is 0.

A moment source, first line give coordinates. The first moment source coordinate is `5000.0 2000.0 -2000.0`, input order is x, y, z. Second line give moment source representation type, include moment_tensor and mechanism_angle. The first moment source representation is `moment_tensor`, so third line give monmen_tensor component, input order is Mxx, Myy, Mzz, Myz, Mxz, Mxy. Mxx = 1.01e15, Myy = 1.0e13, Mzz = 1.0e14, Myz = 1.0e15, Mxz = 1.0e13, Mxy = 1.0e14. The fourth line is wavelet name, include ricker_deriv, gaussian_deriv. The first moment source wavelet name is gaussian_deriv. The fith line is gaussian_rms_width 1.0 and gaussian_peak_time 5.0. The sixth line is start time 0.0.

The second moment source representation type is `mechanism_angle`, so next line input strike dip rake u D A. u is shear modulus, D is slip distance, unit is m, A is area. Strike = 80.0, dip = 70.0, rake = 30.0, u = 1.1e10, D = 1.1, A = 1.0e8.

3.3 value source format

A .valsrc file can be given as:

source.valsrc

```
test_event_1
1 1
0.02 20
3000.0 2000.0 -3000.0
2000.0 3000.0 -2000.0
1.0
1407.403198 14074.032227 140740.312500
5305.979492 53059.796875 530597.937500
19359.816406 193598.187500 1935981.750000
68359.867188 683598.687500 6835986.500000
233580.953125 2335809.750000 23358096.000000
772291.812500 7722918.500000 77229184.000000
2470593.750000 24705938.000000 247059376.000000
7646484.000000 76464848.000000 764648448.000000
22894096.000000 228940960.000000 2289409536.000000
66305560.000000 663055616.000000 6630556160.000000
185734304.000000 1857343104.000000 18573430784.000000
503157920.000000 5031579648.000000 50315792384.000000
1318030848.000000 13180308480.000000 131803086848.000000
3338105344.000000 33381054464.000000 333810532352.000000
8172575744.000000 81725759488.000000 817257578496.000000
19338747904.000000 193387479040.000000 1933874692096.000000
44220518400.000000 442205208576.000000 4422052085760.000000
97689878528.000000 976898818048.000000 9768988049408.000000
208449208320.000000 2084492148736.000000 20844921225216.000000
429488177152.000000 4294882099200.000000 42948817321984.000000
1.5
moment_tensor
15145.9 15145.9 15145.9 15145.9 15145.9 15145.9
28607.7 28607.7 28607.7 28607.7 28607.7 28607.7
51960.8 51960.8 51960.8 51960.8 51960.8 51960.8
90666.5 90666.5 90666.5 90666.5 90666.5 90666.5
151799.5 151799.5 151799.5 151799.5 151799.5 151799.5
243494.3 243494.3 243494.3 243494.3 243494.3 243494.3
373476.1 373476.1 373476.1 373476.1 373476.1 373476.1
546371.6 546371.6 546371.6 546371.6 546371.6 546371.6
759746.0 759746.0 759746.0 759746.0 759746.0 759746.0
999278.3 999278.3 999278.3 999278.3 999278.3 999278.3
1234194.8 1234194.8 1234194.8 1234194.8 1234194.8 1234194.8
```

```
1414798.1 1414798.1 1414798.1 1414798.1 1414798.1 1414798.1
1474371.2 1474371.2 1474371.2 1474371.2 1474371.2 1474371.2
1337506.4 1337506.4 1337506.4 1337506.4 1337506.4 1337506.4
935643.3 935643.3 935643.3 935643.3 935643.3 935643.3
228314.9 228314.9 228314.9 228314.9 228314.9 228314.9
-774273.6 -774273.6 -774273.6 -774273.6 -774273.6 -774273.6
-1994150.0 -1994150.0 -1994150.0 -1994150.0 -1994150.0 -1994150.0
-3281599.0 -3281599.0 -3281599.0 -3281599.0 -3281599.0 -3281599.0
-4430085.0 -4430085.0 -4430085.0 -4430085.0 -4430085.0 -4430085.0
```

first line `test_event_1` is source name. second line `1 1` is force source number 1 and moment source number 1. Third line `0.02 20` is input value source time step interval 0.02 and time steps 20. Fourth line is force source coordinate `3000.0 2000.0 -3000.0`, Fifth line is moment source coordinate `2000.0 3000.0 -2000.0`. Next lines, the values of each source is given in order.

A force source, first give start time. The first force start time is `1.0`. Then give each time step force vector, each line input order is Fx, Fy, Fz.

A moment source, first give start time. The first moment start time is `1.5`. The second line give moment source representation type, include moment_tensor and mechanism_angle. If moment source representation is `moment_tensor`, next each line give moment_tensor component, input order is Mxx, Myy, Mzz, Myz, Mxz, Mxy. If moment source representation is `mechanism_angle`, next each line give mechanism_angle component, input order is strike, dip, rake, u, D, A.

Copyright

Evolution of the code: