EQ Simulator Input Geometry File Format

Michael Barall 04/05/2013

Version 0.6

1. Overview

The geometry file specifies the geometry of the fault system. It contains *sections*, *vertices*, and *elements*. Each element can be either a *rectangle* or a *triangle*.

A section may be either an entire fault or a portion of a fault. We use the word *section* because a section may or may not correspond to a fault segment. For each section, the file contains:

- An integer identification number.
- A name.
- The number of vertices in the section.
- The number of triangles in the section.
- The number of rectangles in the section.
- If spherical coordinates are in use, the extent of the section in latitude, longitude, depth, and DAS.
- If rectangular coordinates are in use, the extent of the section in x, y, z, and DAS.
- A fault identification number.

(*DAS* stands for *distance along strike* and is described below.)

A vertex is a point on the fault surface. For each vertex, the file contains:

- An integer index number.
- If spherical coordinates are in use, the vertex location in latitude, longitude, depth, and DAS.
- If rectangular coordinates are in use, the vertex location in x, y, z, and DAS.
- A flag which indicates if the vertex is part of the fault trace.

An element is a patch of the fault surface. It may be either a rectangle or a triangle. For each element, the file contains:

- An integer index number.
- The index numbers of the 3 (for triangles) or 4 (for rectangles) vertices which lie at the corners of the element.
- The rake, strike, and dip angles.
- The slip rate.
- The aseismicity factor.
- For rectangles, a flag indicating if it is a perfect rectangle.

1.1. Version History

Version 0.6

Release date: 04/05/2013.

For version 0.6, the signature for an input geometry file is “EQSim_Input_Geometry_2” and the specification level is 4. So, the first line of the file contains the following signature record:

```
101 EQSim_Input_Geometry_2 4
```

The following changes were made in version 0.6:

- For the rectangular (x, y, z) coordinate system, the placement of x and y in the file are reversed from the positions they occupied in version 0.5. Specifically, in version 0.6 the x coordinate is identified with longitude, and the y coordinate is identified with latitude. This change was made so that the (x, y, z) coordinate system is a right-handed coordinate system.

Version 0.5

Release date: 03/25/2011.

For version 0.5, the signature for an input geometry file is “EQSim_Input_Geometry_2” and the specification level is 3. So, the first line of the file contains the following signature record:

```
101 EQSim_Input_Geometry_2 3
```

The following changes were made in version 0.5:

- The file format is modified to offer a choice of two coordinate systems: either spherical coordinates or rectangular coordinates. Version 0.4 only supported spherical coordinates.

- In the fault system summary record (kind 200), there is a new field called coord_sys, which specifies if the file uses spherical or rectangular coordinates.

- In the fault system summary record (kind 200), fields that were originally defined to hold latitude, longitude, and depth can now hold x, y, and z instead if rectangular coordinates are in use.

- In the fault section information record (kind 201), fields that were originally defined to hold latitude, longitude, and depth can now hold x, y, and z instead if rectangular coordinates are in use.

- In the vertex record (kind 202), fields that were originally defined to hold latitude, longitude, and depth can now hold x, y, and z instead if rectangular coordinates are in use.

Version 0.4

Release date: 10/31/2010.

For version 0.4, the signature for an input geometry file is “EQSim_Input_Geometry_2” and the specification level is 2. So, the first line of the file contains the following signature record:

```
101 EQSim_Input_Geometry_2 2
```

The following changes were made in version 0.4:

- The description of the fault model (section 2) is expanded to (a) better explain the DAS coordinate, and (b) explain the relationship between the contents of the geometry file and the higher-level fault model from which the geometry file is derived.

- The definition of rake angle is changed, to conform to the Aki and Richards convention.

- In the fault section information record (kind 201), there is a new field called fault_id which contains a fault identification number. It can be used to determine if two sections lie in the same fault.

- In the triangle record (kind 203), there are two new fields called strike and dip, which contain the strike and dip angles for the element. (Version 0.3 already included the rake angle.)

- In the rectangle record (kind 204), there are two new fields called strike and dip, which contain the strike and dip angles for the element. (Version 0.3 already included the rake angle.)

- In the rectangle record (kind 204), there is a new field called perfect_flag, which indicates if the element is a perfect rectangle (*i.e.*, if the four vertices are coplanar and form right angles).

Version 0.3

Release date: 05/25/2010.

For version 0.3, the signature for an input geometry file is “EQSim_Input_Geometry_2” and the specification level is 1. So, the first line of the file contains the following signature record:

```
101 EQSim_Input_Geometry_2 1
```

2. Fault System Model

2.1. Relationship of Geometry File to Fault Model

The purpose of the geometry file is to *transfer* a description of the fault system from some higher-level fault model to an earthquake simulator code. The higher-level fault model might reside in a relational database, or some other sophisticated environment. This geometry format provides a way to store information extracted from the higher-level model, in a form that is accessible and appropriate to the simulator code.

So, the contents of this geometry file should be viewed as being an *approximation* to the higher-level model. For example, some simulator codes require elements to be perfect rectangles. In such a case, the higher-level fault model might contain a smoothly curving continuous fault surface, but the geometry file might contain an approximation which consists of perfect rectangles separated by gaps or overlaps.

To some extent, the design of the geometry file implicitly assumes the existence of the higher-level fault model. In particular, the DAS coordinate (together with depth) provides a way to associate each vertex with its original location in the higher-level fault model. This is discussed below.

2.2. Definitions

The *fault system* is a set of surfaces at which there can be discontinuous displacements.

The fault system is composed of one or more *sections*. Each section corresponds to a single connected surface in the higher-level fault model. Each section has a *section identification number* or *SID*, which is a positive integer. Each section also has a *name*, which is a character string.

Two fault sections are *adjacent* if they adjoin end-to-end to form a single fault. Each section has a *fault identification number* which is used to tell if two sections lie within the same fault.

The *trace* is the intersection of a section with the surface of the earth. If a section is buried, then the trace can be taken to be the uppermost extreme of the section.

The trace is a curve that runs from an *initial point* to a *final point*. The choice of which endpoint is initial and which is final is arbitrary.

Each section has an *orientation* which specifies that one side of the surface is *positive* and the other side is *negative*. The orientation is defined as follows: if you stand on the initial point of the trace, and face toward the final point, then the positive side is on your right. (For example, if the fault runs north to south, with the initial point at the north end and the final point at the south end, then the west side of the fault is the positive side.)

Slip on the fault is the displacement on the positive side, minus the displacement on the negative side.

2.3. Coordinates

A geometry file can contain either spherical coordinates or rectangular coordinates. Models of natural fault systems typically use spherical coordinates (latitude, longitude, depth). Models of non-natural fault systems typically use rectangular coordinates (x, y, z). There is a flag in the file that specifies whether spherical or rectangular coordinates are in use.

A point on the fault surface is described by four coordinates. If spherical coordinates are in use, then three of the coordinates are *latitude*, *longitude*, and *depth*. Latitude and longitude are measured in decimal degrees (positive in the northern and eastern hemispheres respectively), and depth is measured in meters (negative underground).

If rectangular coordinates are in use, then three of the coordinates are x, y, and z, which are all measured in meters. Coordinate z is the same thing as depth, and is negative underground. The earth's surface is the plane $z = 0$.

The fourth coordinate is *DAS* which stands for *distance along strike*. The value of DAS is defined separately for each fault section, and measures how far along strike any given point on the fault surface is located. At the initial point on the fault trace, DAS has its minimum value. At the final point on the fault trace, DAS has its maximum value. DAS is measured in meters.

Conceptually, the pair (DAS, depth) is a 2-dimensional coordinate system for the fault surface *in the original higher-level fault model*. DAS allows us to tie each vertex back to where it came from the in the higher-level fault model. For example, suppose that the original model has a curved fault surface, and you approximate it by perfect rectangles with gaps and overlaps. Then, a vertex on the original curved fault surface may be split into two or more "child" vertices at the corners of perfect rectangles, which are at different locations because of the gaps between the perfect rectangles. Each child vertex should have the same DAS and depth as the "parent" vertex it came from, since all child vertices with a given parent refer to the same location on the original curved fault surface.

We will be using the pair (DAS, depth) to describe the location of earthquakes. This description is independent of the particular meshing used by a given code, and the DAS coordinate lets your results be matched up to the appropriate location in the higher-level fault model.

2.4. More on DAS

The following information is of interest mainly to people who create fault models. If you are writing a simulator code, you can use the DAS values in the geometry file without being too concerned over how they are created.

When you create a fault model, you need to define the DAS values for points in the model. If a fault section is planar, then the definition of DAS is straightforward: Let DAS be distance along the fault trace, and then extend the value of DAS directly down-dip.

If a fault section is curved, then defining DAS is a bit more difficult. One possible technique is to introduce a coordinate axis parallel to the average strike of the fault section, and take DAS to be the coordinate for that axis. An example of such a coordinate axis is the “Hayward Fault coordinate system” described by Graymer *et al.* (Geology, 2005). Another possible technique is to take DAS at the earth’s surface to be distance along the curved fault trace, and then, at each point on the trace, extend DAS down into the earth by plunging perpendicular to the fault strike.

When two fault sections are adjacent, DAS must be defined so it is continuous on the combined fault (i.e., so that DAS at the final point of the first section equals DAS at the initial point of the second section). (This was optional in earlier versions of the specification, but is now mandatory.)

The pair (DAS, depth) is a 2-dimensional coordinate system for the fault surface. In addition, the value of DAS answers the question “what point on the fault trace is ‘above’ a given point?” In particular, the point (DAS, 0) is considered to lie ‘above’ the point (DAS, depth), assuming that the section is not buried.

2.5. Mesh

The shape of the fault surface is described by a *mesh*, which consists of *vertices*, *triangles*, and *rectangles*.

Each fault section has a separate mesh. So, although a vertex belonging to one section might coincide with a vertex belonging to an adjacent section, they are considered to be two separate vertices.

A *vertex* is a point on the fault surface. The location of the vertex is described by four coordinates: latitude, longitude, depth, and DAS.

A *triangle* is a three-sided surface element, bounded by three vertices. The three vertices are always listed in counterclockwise order as viewed from the positive side of the fault.

A *rectangle* is a four-sided surface element, bounded by four vertices. The four vertices are always listed in counterclockwise order as viewed from the positive side of the fault.

A rectangle may be *perfect* or *imperfect*. A rectangle is perfect if all four of its angles are right angles, and all four of its vertices are coplanar. If a rectangle is imperfect, its shape is determined by bilinear interpolation from the vertices. (For those familiar with finite element methods, bilinear interpolation is the usual technique for defining "square" finite elements.)

We do impose one special requirement on rectangles: They must be "rectangular" in the (DAS, depth) coordinate system. That is, the coordinates of the four vertices must be of the form (d_2, z_2), (d_1, z_2), (d_1, z_1), and (d_2, z_1) for DAS coordinates $d_1 < d_2$ and depths $z_1 < z_2$. Note that we have listed the four vertices in counterclockwise order as viewed from the positive side of the fault, as required.

Many simulator codes require perfect rectangles. If such a code encounters an imperfect rectangle, it might reject the geometry file, or it might replace the imperfect rectangle with an approximation that is a perfect rectangle.

Each element has a *strike angle*, a *dip angle*, and a *rake angle*. The angles are measured in decimal degrees. Taken together, the three angles specify the direction of long-term slip on the element. Strike, dip, and rake are defined as given by Aki and Richards.

The Aki and Richards convention is as follows. First, determine which side of the fault is the hanging wall and which side is the footwall. (If the fault is vertical, then one side is arbitrarily chosen as the hanging wall.) Stand on the fault, looking along the trace, with the hanging wall on your right. The direction you are facing is the *strike angle*; 0 degrees is north, and 90 degrees is east. (If the fault is horizontal, then the strike angle may be chosen arbitrarily.) The *dip angle* is the angle between the fault plane and the horizontal; 0 degrees is a horizontal fault, and 90 degrees is a vertical fault. The *rake angle* is the direction that the hanging wall moves relative to the footwall, as measured in the fault plane. A rake angle of 0 degrees is pure strike-slip motion with the hanging wall moving in the strike direction (*i.e.*, left-lateral); 90 degrees is pure thrust motion; and -90 degrees is pure normal motion.

(If the dip angle in the file is negative, then to obtain angles that conform to the Aki and Richards convention you should change the sign of the dip angle and add 180 degrees to the strike angle; do not modify the rake angle.)

When rectangular coordinates are in use, the strike angle is computed by assuming that the x-axis points east and the y-axis points north.

Each element has a *slip rate* that gives the magnitude of the long-term slip rate on the element.

2.6. Trace Waypoints

Each vertex has a *trace flag* which indicates if the vertex is located on the fault trace. In most cases, a vertex is located on the fault trace if and only if its depth is zero. (But if a fault section is buried, then its trace is at non-zero depth.)

The trace flag is an integer which may have the following values:

> 0 – Vertex is not on the fault trace.
> 1 – Vertex is on the fault trace, but not at the initial or final point.
> 2 – Vertex is on the initial point of the fault trace.
> 3 – Vertex is on the final point of the fault trace.

If you extract the set of vertices that lie on the fault trace, and sort them according to DAS, you obtain a series of waypoints for the fault trace. This may be useful for plotting the fault trace on a map.

3. File Format

The input geometry file is a container, as described in the EQ Simulator Container Format. The container format lets us store different kinds of records in a single file.

3.1. Overall File Structure

The following table shows the overall structure of an input geometry file. Specific kinds of records are described later.

	Part	**Description**
1	Header	File header that contains the file signature, metadata, and record descriptors, as described in the container file specification.
2	Summary	One line that contains the fault system summary record. This record gives the total number of sections, vertices, triangles, and rectangles in the entire file.
3	Section info	One line that marks the start of a section. This record gives the name of the section and number of vertices, triangles, and rectangles within it.
4	Vertex list	Multiple lines that list the vertices within the current section, and give coordinates and properties for each.
5	Element list	Multiple lines that list the triangles and rectangles in the mesh.
6		Repeat 3, 4, and 5 for each fault section.
7	End-of-file	One line that marks the end of the file, as described in the container file specification.

The signature for an input geometry file is “EQSim_Input_Geometry_2”. The specification level of this document is 4. So, the first line of the file contains the following signature record:

```
101 EQSim_Input_Geometry_2 4
```

Refer to the container file specification for an explanation of signature and specification level.

It is expected that, typically, either all the elements are triangles, or all the elements are rectangles. However, the file format supports a mix of triangles and rectangles.

3.2. Indexing

Each vertex is assigned an index number. Indexes start at 1, and increase consecutively throughout the entire file. (That is, the index is not reset to 1 at the start of each fault section.) These are Fortran-style indexes.

Each element (triangle or rectangle) is assigned an index number. Indexes start at 1, and increase consecutively throughout the entire file. (That is, the index is not reset to 1 at the start of each fault section.) These are Fortran-style indexes.

4. Record Formats

The following table shows the standard kinds of data records for the input geometry file.

Kind	Name	Description
200	summary	Fault system summary record
201	section	Section information record
202	vertex	Vertex record
203	triangle	Triangle record
204	rectangle	Rectangle record

These are all data records, which means that each record contains a series of data fields. Each kind of record is explained below.

The names “summary” and so forth must be listed in the descriptor part of the file header.

4.1. Fault System Summary Record

```
200 n_section n_vertex n_triangle n_rectangle
    lat_lo lat_hi lon_lo lon_hi depth_lo depth_hi
    coord_sys comment_text
```

This must be the first data record in the file. It gives the total number of fault sections, vertices, triangles, and rectangles in the file. It is provided for the benefit of readers that need to allocate, in advance, all memory needed to store the fault geometry.

The record contains 11 data fields, described in the following table.

	Name	Type	Description
1	n_section	integer	The total number of fault sections in the file.
2	n_vertex	integer	The total number of vertices in the entire file.
3	n_triangle	integer	The total number of triangles in the entire file.
4	n_rectangle	integer	The total number of rectangles in the entire file.

5	lat_lo	real	If spherical coordinates are in use, the lowest value of latitude for any vertex in the entire file, in decimal degrees. If rectangular coordinates are in use, the lowest value of y for any vertex in the entire file, in meters.
6	lat_hi	real	If spherical coordinates are in use, the highest value of latitude for any vertex in the entire file, in decimal degrees. If rectangular coordinates are in use, the highest value of y for any vertex in the entire file, in meters.
7	lon_lo	real	If spherical coordinates are in use, the lowest value of longitude for any vertex in the entire file, in decimal degrees. If rectangular coordinates are in use, the lowest value of x for any vertex in the entire file, in meters.
8	lon_hi	real	If spherical coordinates are in use, the highest value of longitude for any vertex in the entire file, in decimal degrees. If rectangular coordinates are in use, the highest value of x for any vertex in the entire file, in meters.
9	depth_lo	real	The lowest value of depth or z for any vertex in the entire file, in meters.
10	depth_hi	real	The highest value of depth or z for any vertex in the entire file, in meters. Usually this value is 0.0.
11	coord_sys	integer	Coordinate system flag. Values: 0 – Spherical coordinates are in use. 1 – Rectangular coordinates are in use.

The names "n_section" and so forth must be listed in the descriptor part of the file header.

As in any data record, the fields must be separated by one or more blank spaces. The *comment_text* is optional, but if included it must be separated from the last field by one or more blank spaces.

4.2. Fault Section Information Record

```
201 sid name n_vertex n_triangle n_rectangle
    lat_lo lat_hi lon_lo lon_hi depth_lo depth_hi
    das_lo das_hi fault_id comment_text
```

This record introduces a fault section. Each fault section appears in the file as a section information record, followed by a series of vertex records, followed by a series of triangle and rectangle records.

The record contains 14 data fields, described in the following table.

	Name	**Type**	**Description**
1	sid	integer	The section identification number or SID. It is a positive integer. Fault sections must be listed in order of increasing SID, but the numbers do not have to be consecutive.
2	name	character	The name of the fault section, as a character string.
3	n_vertex	integer	The total number of vertices in the section.
4	n_triangle	integer	The total number of triangles in the section.
5	n_rectangle	integer	The total number of rectangles in the section.
6	lat_lo	real	If spherical coordinates are in use, the lowest value of latitude for any vertex in the section, in decimal degrees. If rectangular coordinates are in use, the lowest value of y for any vertex in the section, in meters.
7	lat_hi	real	If spherical coordinates are in use, the highest value of latitude for any vertex in the section, in decimal degrees. If rectangular coordinates are in use, the highest value of y for any vertex in the section, in meters.
8	lon_lo	real	If spherical coordinates are in use, the lowest value of longitude for any vertex in the section, in decimal degrees. If rectangular coordinates are in use, the lowest value of x for any vertex in the section, in meters.
9	lon_hi	real	If spherical coordinates are in use, the highest value of longitude for any vertex in the section, in decimal degrees. If rectangular coordinates are in use, the highest value of x for any vertex in the section, in meters.
10	depth_lo	real	The lowest value of depth or z for any vertex in the section, in meters.
11	depth_hi	real	The highest value of depth or z for any vertex in the section, in meters. Usually this value is 0.0.

12	das_lo	real	The lowest value of DAS (distance along strike) for any vertex in the section, in meters. This is also the value of DAS at the initial point of the fault trace.
13	das_hi	real	The highest value of DAS (distance along strike) for any vertex in the section, in meters. This is also the value of DAS at the final point of the fault trace.
14	fault_id	integer	Fault identification number. It is a positive integer. Two sections lie in the same fault if and only if they have the same fault identification number. The fault identification numbers do not have to be consecutive.

The names “sid” and so forth must be listed in the descriptor part of the file header.

As in any data record, the fields must be separated by one or more blank spaces. Character fields have a maximum length of 100 characters, and may contain only letters, digits, and the special characters underscore, hyphen, plus, and period. The *comment_text* is optional, but if included it must be separated from the last field by one or more blank spaces.

4.3. Vertex Record

```
202 index lat lon depth das trace_flag
    comment_text
```

There is one vertex record for each vertex. It specifies both the location of the vertex and the physical properties of the fault at that location.

The record contains 6 data fields, described in the following table.

	Name	**Type**	**Description**
1	index	integer	Index number for this vertex. The first vertex in the file must have index number 1, and each subsequent vertex increments this value.
2	lat	real	If spherical coordinates are in use, the latitude of this vertex, in decimal degrees. Positive in the northern hemisphere. If rectangular coordinates are in use, the y coordinate of this vertex, in meters.

3	lon	real	If spherical coordinates are in use, the longitude of this vertex, in decimal degrees. Positive in the eastern hemisphere. If rectangular coordinates are in use, the x coordinate of this vertex, in meters.
4	depth	real	The depth or z coordinate of this vertex, in meters. Negative underground.
5	das	real	The DAS or distance along strike of this vertex, in meters.
6	trace_flag	integer	A flag that indicates if this vertex lies on the fault trace. Values: 0 – Vertex is not on the fault trace. 1 – Vertex is on the fault trace, but not at the initial or final point. 2 – Vertex is on the initial point of the fault trace. 3 – Vertex is on the final point of the fault trace.

The names "index" and so forth must be listed in the descriptor part of the file header.

As in any data record, the fields must be separated by one or more blank spaces. The *comment_text* is optional, but if included it must be separated from the last field by one or more blank spaces.

Coordinates must be given with sufficient accuracy so that element strike and dip angles can be computed from the coordinates. If the coordinates are given with an accuracy of about 1.0E-04 times the size of the elements, then the angles can be computed with an accuracy of about 0.01 degrees.

4.4. Triangle Record

```
203 index vertex_1 vertex_2 vertex_3
    rake slip_rate aseis_factor strike dip
    comment_text
```

This record describes a triangular surface element.

The record contains 9 data fields, described in the following table.

	Name	Type	Description
1	index	integer	Index number for this element. The first triangle or rectangle in the file must have index number 1, and each subsequent triangle or rectangle increments this value.

2	vertex_1	integer	Index number of vertex #1. The vertex must be within the same fault section as the triangle.
3	vertex_2	integer	Index number of vertex #2. The vertex must be within the same fault section as the triangle.
4	vertex_3	integer	Index number of vertex #3. The vertex must be within the same fault section as the triangle.
5	rake	real	The average rake angle of this element, in decimal degrees.
6	slip_rate	real	The long-term average slip rate of this element, in meters/second.
7	aseis_factor	real	The aseismicity factor of this element. A dimensionless number between 0.0 and 1.0, indicating what portion of the slip is aseismic.
8	strike	real	The strike angle of this element, in decimal degrees.
9	dip	real	The dip angle of this element, in decimal degrees.

The names “index” and so forth must be listed in the descriptor part of the file header.

As in any data record, the fields must be separated by one or more blank spaces. The *comment_text* is optional, but if included it must be separated from the last field by one or more blank spaces.

Vertices must be listed in counterclockwise order, as viewed from the positive side of the fault.

4.5. Rectangle Record

```
204 index vertex_1 vertex_2 vertex_3 vertex_4
    rake slip_rate aseis_factor strike dip perfect_flag
    comment_text
```

This record describes a rectangular surface element.

The record contains 11 data fields, described in the following table.

	Name	Type	Description
1	index	integer	Index number for this element. The first triangle or rectangle in the file must have index number 1, and each subsequent triangle or rectangle increments this value.
2	vertex_1	integer	Index number of vertex #1. The vertex must be within the same fault section as the rectangle.

3	vertex_2	integer	Index number of vertex #2. The vertex must be within the same fault section as the rectangle.
4	vertex_3	integer	Index number of vertex #3. The vertex must be within the same fault section as the rectangle.
5	vertex_4	integer	Index number of vertex #4. The vertex must be within the same fault section as the rectangle.
6	rake	real	The average rake angle of this element, in decimal degrees.
7	slip_rate	real	The long-term average slip rate of this element, in meters/second.
8	aseis_factor	real	The aseismicity factor of this element. A dimensionless number between 0.0 and 1.0, indicating what portion of the slip is aseismic.
9	strike	real	The strike angle of this element, in decimal degrees.
10	dip	real	The dip angle of this element, in decimal degrees.
11	perfect_flag	integer	A flag that indicates if this is a perfect rectangle: 0 – Not a perfect rectangle. 1 – Perfect rectangle.

The names “index” and so forth must be listed in the descriptor part of the file header.

As in any data record, the fields must be separated by one or more blank spaces. The *comment_text* is optional, but if included it must be separated from the last field by one or more blank spaces.

Vertices must be listed in counterclockwise order, as viewed from the positive side of the fault.

The four vertices must have “rectangular” coordinates in the (DAS, depth) coordinate system. That is, the four vertices must lie at coordinates like (d_2, z_2), (d_1, z_2), (d_1, z_1), and (d_2, z_1) for DAS coordinates $d_1 < d_2$ and depths $z_1 < z_2$.