Specification of the grid managment library GRIDMAN

Vladislav Kotov, FZJ

October 4, 2017

Contents

1 General structure

GRIDMAN is a library of tools to store and manipulate unstructured grids. Its primary application is to prepare grids for the linear Monte-Carlo transport modelling (test particle Monte-Carlo). Different pieces of the problem may be addressed by different codes which use different grids and formats. GRIDMAN provides a universal format for describing such grids, and tools which allow to combine the grids with each over, cut parts of the modelled domains (sub-modelling), divide cells etc. In order to set the background for the transport modelling the combined and transformed grid has to preserve connections to its origins. Essential element of the library which ensures those connections are tables of indexes. The indexes can be defined for each cell or edge of the original grid, they are then stay invariant through all transformation. That is, the indexes do not change when the numbering of cells and edges change.

The library consists of the following essential elements:

- Unified grid format - a data structure which can store various unstructured grids (GRIDMAN_GRID)
- Data structure where tables of indexes are stored (GRIDMAN_INDEX)
- Auxiliary data structure to store lists of elements with variable number of indexes in each (GRIDMAN_INDLIST)
- Set of methods generic for each grid type, such as allocate/deallocate, check, read/write, copy, eliminate cells or edges etc.
- Methods specific for each grid-type: 2D or 3D grid.

In addition to that, there are tools to convert legacy grid formats known in the tokamak edge plasma research (SONNET, TRIA etc.) into GRIDMAN format, and back. At the moment, only 2D grid option is fully implemented, see Section 5, there are only a few 3D specific subroutines implemented.

```
Files and folders of GRIDMAN

gridman.f   : definition of objects (types),
              global variables and interfaces
index.f     : methods of object GRIDMAN_INDEX
indlist.f   : methods of object GRIDMAN_INDLIST
grid1.f     : methods of object GRIDMAN_GRID
grid2.f     : advanced methods of object GRIDMAN_GRID
geom.f      : basic geometrical routines

grid2d      : subroutines specific for the 2D grid type
formats     : reading and writing of grids in formats
              other than GRIDMAN_GRID
convert     : converting grids to and from GRIDMAN_GRID
tests       : test programs (unit tests)
apps        : simple applications based on GRIDMAN
```

In the level diagram below the hierarchy of program units is shown. Units from upper level use units from lower level.

GRIDMAN	
CONVGRID MERGEGRID CUTGRID TRIANG	
GRID2D (grid2d cut merge triang)	CONVERT (tria carre addsurf template vtk)
GRID1 GRID2	
GRID	
INDEX INDLIST FORMAT	
GRIDMAN GRIDMAN_LIB	

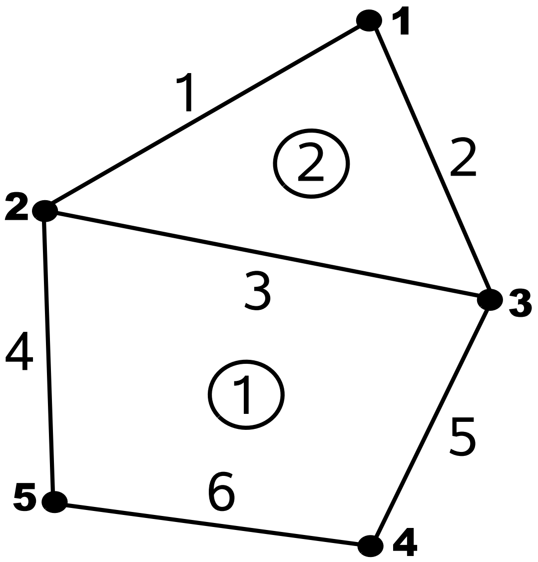
2 Conventions

The names of all public subroutines and other public entities of the library starts from GRIDMAN_. This prefix is typically followed by the name of the data-structure to which the method is related, e.g. GRIDMAN_GRID_, GRIDMAN_INDEX_ .

All public variables and definitions of the data structures are collected in the module GRIDMAN. Explicit interfaces to subroutines are collected in the module GRIDMAN_LIB.

Every subroutine returns error code IERR: 0 for normal operation, < 0 for warnings and > 0 for errors. Both error and warning messages are printed into standard output (global variable GRIDMAN_UNIT). Warning messages begin with string
WARNING from <name of the subroutine>. Errors begin with string
ERROR in <name of the subroutine>.

For each data-type there is a subroutine _CHECK which checks if the data are conformal the defined rules. Normally, other subroutines assume that the input data have passed _CHECK. Check of input data in all subroutines can be enforced by setting GRIDMAN_CHECK=.true.. Printing of extra debugging information can be switched on by setting GRIDMAN_DBG=.true. In particular, messages are printed at the beginning and upon completion of each subroutine.



NEDGES=6; NPOINTS=5; NCELLS=2
PDIM=2; EDIM=2;

CELLS(1,1)=2; CELLS(2,1)=0
CELLS(1,2)=2; CELLS(2,2)=0
CELLS(1,3)=2; CELLS(2,3)=1
CELLS(1,4)=1; CELLS(2,4)=0
CELLS(1,5)=1; CELLS(2,5)=0
CELLS(1,6)=1; CELLS(2,6)=0

POINTS(1,1)=1; POINTS(2,1)=2
POINTS(1,2)=1; POINTS(2,2)=3
POINTS(1,3)=2; POINTS(2,3)=3
POINTS(1,4)=2; POINTS(2,4)=5
POINTS(1,5)=3; POINTS(2,5)=4
POINTS(1,6)=4; POINTS(2,6)=5

X(1:2,1:5)

Figure 1: An example of a simple 2D grid and GRIDMAN_GRID structure which describes it

One more important global variable is `GRIDMAN_TOL` - the accuracy (tolerance) parameter. This parameter is used when two floating point numbers are compared. Numbers `X1` and `X2` are considered to be equal if `abs(X1-X2)<GRIDMAN_TOL*(abs(X1)+abs(X2)+EPS)`. Where `EPS` is the "machine zero" taken to be `10*tiny(X1)`. `EPS` is required for the case when both `X1` and `X2` equal to zero.

3 Description of the grid (GRIDMAN_GRID)

The grid is defined as a set of edges. Each edge can be connected to up to 2 cells. Actual geometrical location of edges is defined by points. Each point, in turn, is defined by it's coordinates. In the simplest case the points are the geometrical grid nodes. In general "a point" may refers to any combination of numbers which are used to define the properties of edge - curvature radius etc. Number of points required to define one edge (`EDIM`), number of point indices required to define one point (`PDIM`) depend on the grid type.

An example of a simple 2D grid is shown in Figure 1. Array `CELLS(:,IEDGE)` contains indexes of cells connected to the edge `IEDGE`. `CELLS(I,IEDGE)<1` (`I=1,2`) means that this side of the edge is not connected to any cell - open boundary of the grid. Negative numbers are allowed to reserve the possibility of "teleportation". If `CELLS(I,IEDGE)<0` then `IEDGE1=-CELLS(I,IEDGE)` is the index of edge which is connected with `IEDGE` directly. It can be that both `CELLS(1:2,IEDGE)<1`, in this case the `IEDGE` does not belong to any cell - free edge. Moreover, the `NCELLS=0` is allowed. Array `POINTS(IP,IEDGE)` contains references to the coordinates in the array `X`. Coordinates of the "vertex" '`IP` of edge `IEDGE` are accessed as follows `IPOINT=POINTS(IP,IEDGE)`, `X(:,IPOINT)`.

Coordinates of the grid generated and used by different programs may be expressed in different units: SI units - meter, centimeter is often used in physics, and millimeter in engineering applications. It is therefore important to store the units of coordinates together with the grid. Text description is stored in the variable `UNITS`. Automatic translation of coordinates when e.g. to grids are merged into one is enabled by the variable `UNIT2SI` which stores translation factor into SI units (Meter). E.g. if the grid is defined in centimeter, then `UNIT2SI=1e-2`.

Tables of edges belonging to each cell, and edges connected to each point can be calculated on demand (`GRIDMAN_GRID_CELLS`) and `GRIDMAN_GRID_POINTS`). Auxiliary data-type `GRIDMAN_INDLIST` is introduced to store such tables with variable number of elements in each entry.

4 Index table (GRIDMAN_INDEX)

Two types of indices can be defined on a grid. Indexes connected with cells - "cell indexes", and connected with edges - "edge indexes". "Point indexes" could be added as well, but they are not implemented at the moment. Same storage format is used for both types. Index table is a 2D array INDEXES(0:NINDEX,NELEMENTS). First column INDEXES(0,:) is the number of the object - cell or edge, to which the combination of indexes is connected. That is, the combination INDEXES(1:NINDEX,IE) is connected to IOBJ=INDEXES(0,IE). E.g. if the cell ICELL refers to a cell of a regular 2D grid, then X and Y indexes on the 2D grid can be accessed as: ICELL=INDEXES(0,IE), IX=INDEXES(1,IE), IY=INDEXES(2,IE).

It is not necessary that the first column INDEXES(0,:) mentions all cells or edges of the grid - the objects may stay unindexed. At the same time, one element may appear in this column more than once - in this case several combinations of indexes are defined for one object. This may be necessary if e.g. a cell is obtained by combining several cells of a finer grid.

The method of indexing applied here is the most flexible one, although it is not always the most optimal one it terms of memory consumption. If only one index is defined for each object, then the method used here requires twice as much memory as the optimal solution (w/o column 0) would require. This is the worst case, in other cases the storage penalty imposed by the applied method is smaller.

5 2D grid

Edges of 2D grid are intervals defined on an (x, y) plane. Each cell of the grid represents a closed chain of points, the edges do not intersect each over. There is an auxiliary subroutine for this grid type which finds closed chain of points for each cell: GRIDMAN_GRID2D_CHAINS.

5.1 Cross-sectional areas of the cells

(GRIDMAN_GRID2D_CROSSECT)

The polygon area is calculated as:

$$S = \frac{1}{2} \sum_{i=1}^{N} |(x_{i+1} + x_i) \cdot (y_{i+1} - y_i)| = \frac{1}{2} \left| \sum_{i=1}^{N} (x_i y_{i+1} - x_{i+1} y_i) \right| \tag{1}$$

5.2 Cylindrical areas of edges

(GRIDMAN_GRID2D_CYLAREAS)

The 2D grid type can also represent an axi-symmetric grid in cylindrical coordinates. In this case x plays the role of the radial coordinate. Cylindrical areas of edge are calculated as side areas of a conical frustum:

$$S = \pi (x_1 + x_2) \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2}$$

5.3 Cell volumes

(GRIDMAN_GRID2D_CYLVOLUMES)

Volumeof a cell of axi-symmetric grid (x is the radial coordinate) is calculated as a sum of the signed volumes of the conical frustums, see Figure 2a:

$$V = \frac{\pi}{3} \left| \sum_{i=1}^{N} \left(x_{i+1}^2 + x_i^2 + x_{i+1} x_i\right) \cdot (y_{i+1} - y_i) \right| = \frac{\pi}{3} \left| \sum_{i=1}^{N} (x_{i+1} + x_i) \cdot (x_i y_{i+1} - x_{i+1} y_i) \right| \tag{3}$$

This formula is valid for a closed polygon without self-intersections. By convention the sign of the individual term of the sum is positive if the points are arranged counter-clockwise and $y_{i+1} > y_i$.

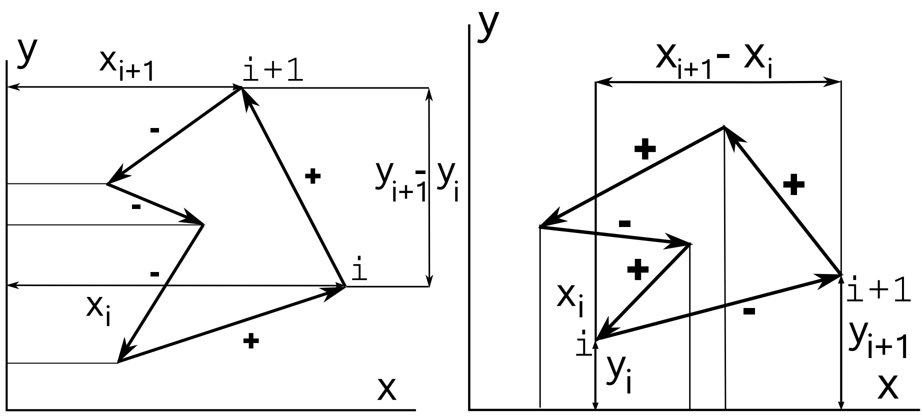
5.4 Cell centers

(GRIDMAN_GRID2D_CENTER)

As cell centers the centers of mass of the polygons (X_c, Y_c) are taken.

$$X_c = \frac{1}{S} \int x dx dy = \frac{V_y}{2\pi S}; \quad Y_c = \frac{1}{S} \int y dx dy = \frac{V_x}{2\pi S}; \quad S = \int dx dy$$

Area S is calculated from Equation (1) without taking absolute value.



(a) V_y (b) V_x

Figure 2: Calculating volume of the toroidall cell revolving a) around x axis and b) around y axis

The calculation of V_y is given by Equation (3) (without taking absolute value), therefore:

$$X_c = \frac{1}{6S}\sum_{i=1}^{N}\left(x_{i+1}^2 + x_i^2 + x_{i+1}x_i\right)\cdot\left(y_{i+1} - y_i\right) = \frac{1}{6S}\sum_{i=1}^{N}\left(x_{i+1} + x_i\right)\cdot\left(x_i y_{i+1} - x_{i+1}y_i\right) \tag{4}$$

Here the terms of the sum have a positive sign if the points are arranged counter-clock wise and $x_i > x_{i+1}$. This ensures that the sum is always positive for the counter-clock wise arrangement of points, same as in Equation (3).

The volume V_x is calculated in a way similar to V_y as a sum of the volumes of the conical frustums, see Figure 2b:

$$Y_c = \frac{1}{6S}\sum_{i=1}^{N}\left(y_{i+1}^2 + y_i^2 + y_{i+1}y_i\right)\cdot\left(x_i - x_{i+1}\right) = \frac{1}{6S}\sum_{i=1}^{N}\left(y_{i+1} + y_i\right)\cdot\left(x_i y_{i+1} - x_{i+1}y_i\right) \tag{5}$$

The equalities are valid for closed polygons.

For Equation (1) one gets:

$$\sum_{i=1}^{N}\left(x_{i+1} + x_i\right)\cdot\left(y_{i+1} - y_i\right) = \sum_{i=1}^{N}\left(x_i y_{i+1} - x_{i+1}y_i + x_{i+1}y_{i+1} - x_i y_i\right) =$$

$$\sum_{i=1}^{N}\left(x_i y_{i+1} - x_{i+1}y_i\right) + \sum_{i=1}^{N} x_{i+1}y_{i+1} - \sum_{i=1}^{N} x_i y_i =$$

$$\sum_{i=1}^{N}\left(x_i y_{i+1} - x_{i+1}y_i\right) + \sum_{i=2}^{N} x_i y_i + x_{N+1}y_{N+1} - \sum_{i=2}^{N} x_i y_i - x_1 y_1 = \sum_{i=1}^{N}\left(x_i y_{i+1} - x_{i+1}y_i\right) \tag{6}$$

$x_1 y_1 = x_{N+1}y_{N+1}$ since the polygon is closed.

Equation (5) can be transformed as well:

$$\sum_{i=1}^{N}\left(y_{i+1}^2 + y_i^2 + y_{i+1}y_i\right)\cdot\left(x_i - x_{i+1}\right) = \sum_{i=1}^{N}\left(x_i y_{i+1}^2 + x_i y_i^2 + x_i y_{i+1}y_i - x_{i+1}y_{i+1}^2 - x_{i+1}y_i^2 - x_{i+1}y_{i+1}y_i\right) =$$

$$\sum_{i=1}^{N}\left(x_i y_{i+1}^2 + x_i y_{i+1}y_i - x_{i+1}y_i^2 - x_{i+1}y_{i+1}y_i\right) + \sum_{i=1}^{N} x_i y_i^2 - \sum_{i=1}^{N} x_{i+1}y_{i+1}^2 =$$

$$\sum_{i=1}^{N}\left[y_{i+1}\left(x_i y_{i+1} - x_{i+1}y_i\right) + y_i\left(x_i y_{i+1} - x_{i+1}y_i\right)\right] = \sum_{i=1}^{N}\left(y_{i+1} + y_i\right)\cdot\left(x_i y_{i+1} - x_{i+1}y_i\right) \tag{7}$$

Here since $x_{N+1} = x_1$ and $y_{N+1} = y_1$ (closed polygon):

$$\sum_{i=1}^{N} x_i y_i^2 - \sum_{i=1}^{N} x_{i+1}y_{i+1}^2 = \sum_{i=2}^{N} x_i y_i^2 + x_1 y_1^2 - \sum_{i=2}^{N} x_i y_i^2 - x_{N+1}y_{N+1}^2 = 0$$

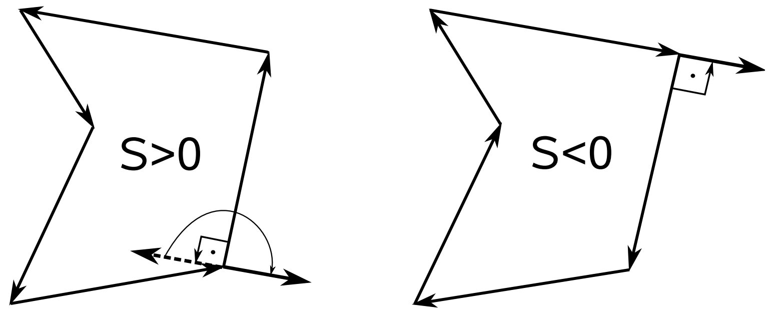


Figure 3: Rule for choosing the direction of the edge normal vector

5.5 Normals to the edges

(`GRIDMAN_GRID2D_NORM`)

Coordinates of the surface normal which points to the left from the direction (x_1, x_2) can be found with rotation matrix: turning to the angle $\pi/2$ in the counter-clockwise direction:

$$\begin{Bmatrix} \cos\pi/2, & -\sin\pi/2 \\ \sin\pi/2, & \cos\pi/2 \end{Bmatrix} \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} = \begin{Bmatrix} 0, & -1 \\ 1, & 0 \end{Bmatrix} \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} = \begin{pmatrix} y_1 - y_2 \\ x_2 - x_1 \end{pmatrix}$$

To chose the direction of the normal vector the signed area of the polygon is calculated according to Equation (1) (w/o taking absolute value). If the edge in question is oriented counter-clockwise, then the area S is positive and the sign of the normal vector has to be changed, see Figure 3, left. Otherwise, if $S < 0$ then the sign is correct, Figure 3, right. At the end the normal vector is also normalized. The final formula reads:

$$\begin{pmatrix} x_N \\ y_N \end{pmatrix} = \frac{-\mathrm{sign}\,(S)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \begin{pmatrix} y_1 - y_2 \\ x_2 - x_1 \end{pmatrix}$$

5.6 Cells are convex or not?

(`GRIDMAN_GRID2D_ISCONVEX`)

Cells are convex if all corners are smaller than 180^o. This is true if the combination:

$$v = (x_i - x_{i-1})(y_{i+1} - y_i) - (x_{i+1} - x_i)(y_i - y_{i-1})$$

has the same sign for each corner (each i), $v = 0$ are counted separately.

5.7 Merging two grids

(`GRIDMAN_GRID2D_MERGE`)

Merging means trying to attach boundary edges of the two grids to each over. That is, edges with at least one `CELLS(I,ICELL)`$\leq$0. If an edge of one grid matches to the set of edges of another grid, then this edges is replaced be this set of edges. Duplicate points are removed.

To find if a point (end of one edge) belongs to the interval (another edge) the following formulas are used. Distance between point and line:

$$\delta = \frac{Ax + By + C}{\sqrt{A^2 + B^2}}$$

Equation of straight line:

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

Or:

$$x(y_2 - y_1) + y(x_1 - x_2) + y_1(x_2 - x_1) - x_1(y_2 - y_1) = 0$$

That means $A = y_2 - y_1$, $B = x_1 - x_2$, $C = y_1(x_2 - x_1) - x_1(y_2 - y_1)$ and:

$$\delta = \frac{(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

Finally, the condition that that point (x, y) lies on the line $((x_1, y_1)(x_2, y_2))$ is written as:

$$(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1) = 0 \tag{8}$$

It has to be taken into account, however, that due to finite precision the coordinates known to the code may differ from true coordinates. Therefore, Equation (8) is fulfilled not for (x, y) and $((x_1, y_1)(x_2, y_2))$, but for $(x + \Delta x, y + \Delta y)$ and $((x_1 + \Delta x_1, y_1 + \Delta y_1)(x_2 + \Delta x_2, y_2 + \Delta y_2))$:

$$(x - x_1 + \Delta x - \Delta x_1)(y_2 - y_1 + \Delta y_2 - \Delta y_1) - (y - y_1 + \Delta y - \Delta y_1)(x_2 - x_1 + \Delta x_2 - \Delta x_1) = 0$$

Neglecting terms of the order Δ^2 yields:

$$(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1) =$$

$$= (x_2 - x_1)(\Delta y - \Delta y_1) + (y - y_1)(\Delta x_2 - \Delta x_1) - (y_2 - y_1)(\Delta x - \Delta x_1) - (x - x_1)(\Delta y_2 - \Delta y_1) \tag{9}$$

Thus:

$$|(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1)| \leq$$

$$\leq |x_2 - x_1|(|\Delta y| + |\Delta y_1|) + |y - y_1|(|\Delta x_2| + |\Delta x_1|) + |y_2 - y_1|(|\Delta x| + |\Delta x_1|) + |x - x_1|(|\Delta y_2| + |\Delta y_1|) \tag{10}$$

Inequality (10) is fulfilled if Equation (9) is fulfilled, thus, this inequality gives necessary condition of fulfilling the Equation (9). Further, Δ are not known but can be estimated as $|\Delta x| < \epsilon |x|$, $|\Delta y| < \epsilon |y|$ etc. where ϵ is some prescribed constant which describes relative accuracy of coordinates, e.g. $\epsilon = 10^{-5}$. Accuracy ϵ is mostly determined by accuracy of the data representation in the files.

Finally, Equation (10) is translated into the form:

$$|(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1)| \leq$$

$$< [|x_2 - x_1|(|y| + |y_1|) + |y - y_1|(|x_2| + |x_1|) + |y_2 - y_1|(|x| + |x_1|) + |x - x_1|(|y_2| + |y_1|)]\,\epsilon$$

This inequality is used in the code as condition that point (x, y) lies on the line $((x_1, y_1)(x_2, y_2))$. To find out if the point, in addition, lies inside the interval, obvious extra conditions have to be fulfilled (non-zero length of the interval is assumed):

$$\begin{cases} |x_2 - x_1| < |y_2 - y_1|, & \begin{cases} y_2 > y_1, & y_1 < y < y_2 \\ y_1 > y_2, & y_1 < y < y_2 \end{cases} \\ |x_2 - x_1| \geq |y_2 - y_1|, & \begin{cases} x_2 > x_1, & x_1 < x < x_2 \\ x_1 > x_2, & x_1 < x < x_2 \end{cases} \end{cases}$$

To check if the point (x, y) coincides with one of the ends, that is $x = x_1$ and same for y, same technique as above is used to find the necessary condition:

$$x + \Delta x = x_1 + \Delta x_1 \Rightarrow |x - x_1| \leq |\Delta x| + |\Delta x_1| < \epsilon(|x| + |x_1|)$$

5.8 Grid triangulation

See DOXYGEN documentation and source code of `GRIDMAN_GRID2D_TRIANG`

5.9 Cut grid by a polygone

See source code of `GRIDMAN_GRID2D_CUT`

6 CONVGRID

CONVGRID is a versatile grid convertor based on GRIDMAN library. Examples of the input files
`convgrid.parameters` are shown below in Section 6.2.

6.1 Input parameters

```
Usage:

./convgrid help
 prints documentation

./prospect <option> <file1> <file2> <file3>
  convert <file?> into VTK format.
  Input format depends on <option>
   -s  : sonnet (carre) grid in file1
   -f  : fort.30 grid in  file1
   -t  : EIRENE triangular grid in file1 file2 file3
         (fort.33,34,35 - nodes, elements, neigbors)
   -e2 : 2D Additional Surfaces in EIRENE input file file1
   -e3 : 3D Additional Surfaces in EIRENE input file file1
   -p  : template in file1
   -g  : gridman grid in file1

Commands -fp, -tp, -ep produce template text files which
can be viewed e.g. in DG

Commands which start from '--':
--s, --f, --t, --e2, --e3, --p, --g
only print metadata of the grids

./convgrid < convgrid.parameters
  output is controlled by the FORTRAN namelist CONVGRID:
&CONVGRID
....
....
/
<new line>

Variablies to be defined in FORTRAN namelist CONVGRID:

DESCRIPTION : string (256 character max) which describes
              the data in the output

SONNET_IN   : name of the input file with CARRE grid
              in SONNET format
FORT30_IN   : name of the input file with CARRE grid
              in EIRENE fort.30 format
FORT33_IN   : name of the input files with
FORT34_IN   : triangular grid in
FORT35_IN   : EIRENE fort.33,34,35 format (nodes, elements, neigbors)
TEMPLATE_IN : name of simple template file (as produced
              e.g. by DG) with polygon definition
GRID_IN     : name of the input file
              in GRIDMAN_GRID format
EIRENE_IN   : name of the EIRENE input file from which
              Additional Surfaces are taken
RLBND       : this variable defines the type of Additional Surfaces
              which will be read from the EIRENE input file.
              Relevant only for EIRENE_IN option.
              If RLBND=2 (default) then 2D intervals are read.
              If RLBND=3 the 3D triangles are read

Either SONNET_IN or FORT30_IN or FORT33,34,35_IN or GRID_IN has to be
```

```
specified. Ambiguity leads to error message.

LEIRENE       : is a switch which affects cell index mapping
                generated from SONNET or FORT.30 file.
                If LEIRENE=.TRUE. then the cell indices are
                generated in accordance with EIRENE definition.
                Default LEIRENE=.FALSE., that is B2 definition.

NEXCLUDE      : Number of blocks to be exluded from the grid.
                Default value 0.
  IXE(2,:)    : Cells lying between IXE(1) and IXE(2) in X direction,
  IYE(2,:)    : and between IYE(1) and IYE(2) in Y direction
                are taken away from the grid.
                This option is only  applicable to grids with
               CELLINDEX(1)%NINDEX>1, and has no effect if
                this is not the case.
               IX=CELLINDEX(1)%INDEXES(1,:)
               IY=CELLINDEX(1)%INDEXES(2,:)

TRIANGULATE : if .TRUE. then the resulting grid is triangulated.
                Default is .FALSE.

IEIND         : if IEIND>0 then only edge index number IEIND
                is taken over in the resulting grid
                if IEIND==0 tnen no edge indices
               from the input are taken over
                if IEIND<0 (default) tnen all edge indices
               from the input are taken over
ICIND         : same as IEIND for cell indices

VTK_OUT         : name of the file with output in VTK format
                  which can be used by graphics programms, e.g. Paraview
                  Cell indeces added to the plot as cell-centered
                  scalar values
GRID_OUT        : name of the file with output in GRIDMAN_GRID format
FORT33_OUT      : names of the the files with output
FORT34_OUT      : in the form of EIRENE triangular grid
FORT35_OUT      : fort.33,34,35 format
TEMPLATE_OUT : name of the file with output in simple
                  template format, can be used e.g. by DG

FSCALE    : factor which translates units into Meter
             e.g. if coordinates have to be in CM then FSCALE=1e-2
             If fscale<=0 the no unit translation is applied
             Default value FSCALE=-1.
UNITS     : string which describes the units after translation,
             applied only if FSCALE>0. Default 'Not defined'

DBGMOD  : if .TRUE. then extra debugging output is produced.
           Default is .FALSE.
LCHECK  : if .TRUE. then standard check in GRIDMAN subroutines
           are enforced.  Default is .FALSE.

Default units:

 SONNET_IN      : Meter
 FORT30_IN      : Meter
 FORT33_IN      : Centimeter
 FORT34_IN
```

```
FORT35_IN     :
EIRENE_IN     : Centimeter
TEMPLATE_IN   : Millimeter
```

6.2 Examples

```
Read triangular grid in EIRENE format and
write it in GRIDMAN_GRID format,
generate *.vtk output for graphic programs
&CONVGRID
DESCRIPTION='Triangular grid for Alcator C-mod'
FORT33_IN='input/fort.33.cmod',
FORT34_IN='input/fort.34.cmod',
FORT35_IN='input/fort.35.cmod',
FORT33_OUT='fort.33',
FORT34_OUT='fort.34',
FORT35_OUT='fort.35',
VTK_OUT='cmod_tria.vtk',
GRID_OUT='cmod_tria.grd',
LCHECK=.TRUE.,
/

Read CARRE grid in fort.30 format and
translate it into GRIDMAN_GRID format.
Exclude core cells. Translate units into Centimeter.
Generate *.vtk output for graphic programs
&CONVGRID
DESCRIPTION='CARRE grid for Alcator C-mod'
FORT30_IN='input/fort.30.cmod',
GRID_OUT='cmod_eliminate.grd',
VTK_OUT='cmod_eliminate.vtk',
FSCALE=1e-2,
UNITS='CM',
NEXCLUDE=1,
IXE(1,1)=25,72,
IYE(1,1)=0,16,
LCHECK=.TRUE.,
/

Read CARRE grid in SONNET format.
Exclude core part. Triangulate.
Generate EIRENE triangular grid,
*.vtk output for graphic programs,
as well as simple template file
for programs like DG.
Flag LEIRENE is used to get indices
of SONNET grid in accordance with
EIRENE definition
&CONVGRID
DESCRIPTION='Triangulated ITER SOL grid',
SONNET_IN='input/iterm.carre.105',
LEIRENE=.TRUE.,
NEXCLUDE=1,
IXE(1,1)=22,72,
IYE(1,1)=0,12,
TRIANGULATE=.TRUE.,
VTK_OUT='iter_sol.vtk',
FORT33_OUT='fort.33',
FORT34_OUT='fort.34',
FORT35_OUT='fort.35',
```

```
LCHECK=.TRUE.,
TEMPLATE_OUT='tria.txt',
/

Read EIRENE Additional Surfaces and
convert them into GRIDMAN_GRID object
and VTK plot
&CONVGRID
DESCRIPTION='EIRENE Additional Surfaces',
EIRENE_IN='./input/input.eir',
GRID_OUT='eirene.grd',
VTK_OUT='eirene.vtk',
LCHECK=.TRUE.,
/
```

7 MERGEGRID

MERGEGRID is an interface to `GRIDMAN_GRID2D_MERGE` which merges two grids.

7.1 Input parameters

```
Variablies to be defined in FORTRAN namelist MERGEGRID:
&MERGEGRID
....
....
/

GRID1_IN : name of the file with 1st grid object
GRID2_IN : name of the file with 2nd grid object
           which will be merged into the first one

MTOL     : tolerance parameter for GRIDMAN_GRID2D_MERGE which
           defines relative accuracy of the poits coordinates.
           Default value is 1e-5. It might be necessary to increase
           or decrease TOL in some individual cases. If TOL is too large
           than the code can mistakenly take two different points
           for one point. If TOL is too small then the code cannot
           recognise what a point is sitting on the grid edge.

GRID3_OUT: name of the file where the resulting combined grid
           is stored

Grids are read and stored in GRIDMAN_GRID format

DBGMOD   : if .TRUE. then extra debugging output is produced.
           Default is .FALSE.
LCHECK   : if .TRUE. then standard check in GRIDMAN subroutines
           are enforced.  Default is .FALSE.
```

8 TRIANG

TRIANG combines plasma grid with triangular grid between plasma and wall, and generates triangular grid for EIRENE. This program can replace TRIAGEOM.

8.1 Input parameters

```
Usage:

./triang help
```

```
 prints documentation

./triang < triang.parameters
  output is controlled by the FORTRAN namelist TRIANG D:
&TRIANG
....
....
/
<new line>

Variablies to be defined in FORTRAN namelist TRIANG:

FORT30_IN   : name of the input file with CARRE grid
              in EIRENE fort.30 format
FORT33_IN   : name of the input files with
FORT34_IN   : triangular grid in
FORT35_IN   : EIRENE fort.33,34,35 format (nodes, elements, neigbors)

TOL         : tolerance parameter for the mergegrid algorithm
              Default value TOL=1e-5

FORT33_OUT : names of the the files with resulting combined grid
FORT34_OUT : combined trianular grid in fort.33,34,35 format
FORT35_OUT : (nodes, elements, neigbors)


VTK_OUT        : name of the file with output in VTK format
                 which can be used by graphics programms, e.g. Paraview
GRID_OUT       : name of the file with output in GRIDMAN_GRID format
TEMPLATE_OUT : name of the file with output in simple
                 template format, can be used e.g. by DG

DBGMOD      : if .TRUE. then extra debugging output is produced.
              Default is .FALSE.
LCHECK      : if .TRUE. then standard check in GRIDMAN subroutines
              are enforced.  Default is .FALSE.
```

8.2 Example

```
Generate combined triangular grid out of
plasma grid and triangular grid
in the region between plasma and wall
&TRIANG
FORT30_IN='input/fort.30.jet',
FORT33_IN='input/fort.33.jet.tria',
FORT34_IN='input/fort.34.jet.tria',
FORT35_IN='input/fort.35.jet.tria',
TOL=1e-5,
FORT33_OUT='fort.33',
FORT34_OUT='fort.34',
FORT35_OUT='fort.35',
VTK_OUT='combined.vtk',
GRID_OUT='combined.grd',
TEMPLATE_OUT='combined.txt',
DBGMOD=.FALSE.,
LCHECK=.FALSE.,
/
```

9 CUTGRID

CUTGRID is an interface to `GRIDMAN_GRID2D_CUT` which cuts part of the grid inside or outside of prescribed contour.

9.1 Input parameters

```
Variablies to be defined in FORTRAN namelist CUTGRID:
&CUTGRID
....
....
/

DESCRIPTION     : string (256 characters max) which describes
                  the resulting grid and mappings
GRID_IN         : name of the file with grid in
                  GRIDMAN_GRID format
CONTOUR_IN         : name of the input file with cloused contour
                     (in template format) which is used to cut the grid

UNIT2METER         : if UNIT2METER>0., then this is a
                     factor which translates the units of CONTOUR_IN
                     into Meter. E.g. if coordinates in CONTOUR_IN
                     are defined in Centimeter then UNIT2METER=1e-2
                     If UNIT2METER<=0., then the transformation
                     is not applied, it is assumed that the coordinates
                     in CONTOUR_IN are defined in Millimeter.
                     Default UNIT2METER=-1.,

LEXCLUDE        : if .FALSE. then part of the grid inside
                  CONTOUR_IN is taken to the resulting grid,
                  if .TRUE. then CONTOUR_IN is excluded from
                  the resulting grid. Default value .FALSE.

IEIND : if IEIND>0 then edge index number IEIND of GRID_IN is
        merged into indexes of segments in CONTOUR_IN and
        is stored as first edge index of GRID_OUT

GRID_OUT           : name of the file where the resulting grid is stored

DBGMOD  : if .TRUE. then extra debugging output is produced.
          Default is .FALSE.
LCHECK  : if .TRUE. then standard check in GRIDMAN subroutines
          are enforced.  Default is .FALSE.
TOL     : tolerance parameters, default is GRIDMAN_TOL

CUTGRID may not handle properly the situation when the polygon
(contour) segments intersect the grid nodes, or when the grid
edges intersect the polygon vertices. To identify such intersections
it is recommended to run the program with LCHECK=.TRUE.
```

9.2 Example

```
&CUTGRID
DESCRIPTION='ITER F57, wide grid. Cut by first wall',
GRID_IN='ITER_F57_wide.grd',
CONTOUR_IN='input/ITER_F57.wall',
GRID_OUT='ITER_F57_wide_cut.grd',
IEIND=1,
/
```