

# **etol tools user manual v 0.9.8**

---

user manual for tools distributed with tol libraries

**Vittorio Rigamonti**

---

This manual is for etol tools (version 0.9.8, 25 October 2004), a set of tools that produces good quality 3D surface mesh, for numerical simulation based on finite element method. This edition documents version 0.9.8.

Copyright ©2004 Vittorio Rigamonti

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover texts being “A GNU Manual,” and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled “GNU Free Documentation License.”

(a) The FSF’s Back-Cover Text is: “You have freedom to copy and modify this GNU Manual, like GNU software. Copies published by the Free Software Foundation raise funds for GNU development.”

Published by Vittorio Rigamonti

## Table of Contents

<b>1</b>	<b>Introduction</b> .....	<b>1</b>
<b>2</b>	<b>The triangulation optimization process</b> .....	<b>2</b>
<b>3</b>	<b>Input Format</b> .....	<b>3</b>
<b>4</b>	<b>Configuration</b> .....	<b>4</b>
<b>5</b>	<b>March</b> .....	<b>5</b>
<b>6</b>	<b>Umbrella Smooth</b> .....	<b>7</b>
<b>7</b>	<b>Landing</b> .....	<b>8</b>
<b>8</b>	<b>Coarse</b> .....	<b>9</b>
<b>9</b>	<b>Refine</b> .....	<b>11</b>
<b>10</b>	<b>Swap</b> .....	<b>13</b>
<b>11</b>	<b>tolCut</b> .....	<b>14</b>
<b>12</b>	<b>Improving cutting surface quality</b> .....	<b>15</b>
	12.1 Improving quality by refining .....	15
	12.2 Improving quality by swapping .....	15
<b>13</b>	<b>Converting GTS to OOGL (Geomview) or SURF (Netgen)</b> .....	<b>16</b>
<b>14</b>	<b>Modular Architecture</b> .....	<b>17</b>
	14.1 Changing regular function .....	20
	<b>Appendix A Copying This Manual</b> .....	<b>23</b>
	A.1 GNU Free Documentation License .....	23
	A.1.1 ADDENDUM: How to use this License for your documents .....	29
	<b>Index</b> .....	<b>30</b>

# 1 Introduction

ETOL (Extensible Triangulation Optimization Library) Tools are a suite of applications designed to produce good quality surface triangulation starting from an analytical description of the surface.

The reference field of application is the study of vascular districts haemodynamics ruled by partial differential equations.

Tools are provided to produce, regularize, refine, coarsen the triangulation and swap edges.

Each of the tools is designed with a modular architecture, in order to allow the user to modify most of the algorithms implemented as default behavior.

Surface triangulations are described inside the tools and outside (serialized) as GTS objects. Go to the official [Gnu Triangulated Surface Library Home Page](#) for more informations about GTS, surface triangulation and a lot of interesting documentation on computational geometry.

## 2 The triangulation optimization process

ETOL implements an iterative mesh optimization process driven by the control of the geometric error, which is the error introduced in the model due to the approximated description of the boundary surface through a triangular mesh.

The final surface mesh is the result of the following optimization process (described in *Ottimizzazione di triangolazioni su superfici implicite per la simulazione numerica*, Vittorio Rigamonti, Degree Thesis (2004), Politecnico di Milano):

1. construction of the first mesh (marching algorithm)
2. regularization by mean of discrete laplacian operator (umbrella smoothing)
3. vertices landing (foot point algorithm)
4. coarsening
5. refining
6. edge swapping

A reference book for construction, coarsening, refining is *Mesh Generation: Application to Finite Elements*, P.J. Frey. Kogan Page, 2000. In particular, when in the following we will refer to the *Frey metric* we intend the metric calculated in order to control the interpolation error as described in this book.

The umbrella smoothing implemented in this version of ETOL tools, is a very simple algorithm that increases the mesh quality through nodes moving. This algorithm assumes that the polygon described by each vertex's neighborhood is convex, if this assumption fail there a possibility to produce a self intersecting mesh (user can check if the mesh has this kind of artifacts with `gtscheck`).

The foot point algorithm is an implementation of the algorithm described in *On the curvature of curves and surfaces defined by normalforms*. E.Harmann, *Comp. Aided. Geometry. Des.* 16, 355-376, 1999.

## 3 Input Format

Conceptually input data for ETOL is an implicit surface, thus a surface defined by the roots of a regular function provided by first and second derivatives. This kind of representation can describe in a compact way surfaces with complex topology, such as the walls of vessel trunks.

For the default configuration of the tools the implicit definition contains functions belonging to the radial basis functions, because these functions are proven to be well suited for interpolation of surfaces described by point-cloud data, i.e. data produced by scanner or tomography. (*Reconstruction and representation of 3D objects with radial basis functions, J.C.Carr et al, ACM SIGGRAPH 8/2001*).

An RBF can be defined by the type of the radial function used (i.e. linear, cubic...) and a list of pairs (centers, coefficient). The default configuration of ETOL utilizes linear radial basis functions and reads the list of centers, coefficients from a file ‘.cnt’ with the following ASCII format:

```
<number of pairs N>
<coordinate for centers 1> <coefficient 1>
<coordinate for centers 2> <coefficient 2>
...
<coordinate for centers N> <coefficient N>
```

More examples can be found in the ‘cnt’ directory.

## 4 Configuration

The algorithms behavior can be controlled by means of configurable parameters that can be provided using a configuration file or by command line. The configuration process is almost the same for each ETOL tool and parameters get their values in the following order:

1. a default value
2. from a file with the same name of the tool and extension `‘.pot’`
3. or from the file specified in the command line option `‘--pot’`
4. values read from the file can be overridden by command line options.

The syntax for the parameter file and for the command line options are explained for each tool in the relative chapter, wich also explains the parameters semantic.

etol tools use GetPot class to process configuration parameters, more information can be found at [GetPot Home Page](#).

### **ETOL debugging messages waste your display?**

Try to run the tools adding `2>/dev/null`.

## 5 March

Produces a triangulation of the specified implicit surface and writes it to the standard output in GTS format. Configuration parameters can be defined in `march.pot` file (or in the file specified with `--pot` option and overridden in the command line).

```
march    -p,-pot configuration file .pot
          -h -help
```

### Configuration Options

Name	Default	Description
file		Name of the <code>.cnt</code> coefficients file
smoothing	2	Smoothing factor for the implicit surface
march/margin_x	10	Margin of the bounding box relative to the x dimension in %
march/margin_y	10	Margin of the bounding box relative to the y dimension in %
march/margin_z	10	Margin of the bounding box relative to the z dimension in %
march/division_x	20	division of the bounding box along the x dimension
march/division_y	20	division of the bounding box along the y dimension
march/division_z	20	division of the bounding box along the z dimension

### Customization Options (*For advanced users only*)

Name	Default	Description
implicit_func_module		Module containing the regular function for the surface
march/module	marchExt/marchingCubes.so	file <code>.so</code> with the triangulation algorithm
march/tol_triangulate	tol_triangulate	name of the function that implements the triangulation
march/grid_key	cartesian_grid	hash table key name of the grid object
implicit_surface_key	implicit_surface	hash table key name of the implicit surface object

The name `march` come from the fact the triangulation algorithm initially developed was based on marching cubes, however it is possible, with this version of the tools, develop user triangulation algorithm (e.g. advancing front) and link it run time using the `'march/module'` option.

### Example

In this example we will produce the first triangulation of the implicit surface defined by the file `'crane-knee2.cnt'` (bundled in the tarball). Type:

```
march file=../cnt/crane-knee2.cnt > ck2.gts
```

`march` used the `'march.pot'` as a configuration file. The marching cubes use a working volume that exceed the surface bounding box by (5%,10%,10%) along the three dimension and splits it with a grid (30,30,20). The `.cnt` file that describe the implicit surface is `'../cnt/crane-knee2.cnt'`, this value gived as variable in the command line overrides the *file* value in the `'march.pot'` file. Te result can be seen with `mview` or converted in OOGL format with `gts2oogl` and displayed with `geomview`.

## 6 Umbrella Smooth

Regularizes the triangulation received from the standard input using an umbrella operation (discrete laplacian) and write it to the standard output.

```
umbrellaSmooth
    '-p,--pot' 'configuration file .pot'
    '-h --help'
```

<b>Name</b>	<b>Default</b>	<b>Description</b>
implicit_func_module		Module containing the regular function for the surface
file		Name of the .cnt coefficients file
smoothing	2	Smoothing factor for the implicit surface

### Example

Now we regularize the mesh produced in the previous example launching the command:  
`umbrellaSmooth < ck2.gts > ck2Umb.gts`

## 7 Landing

It projects the vertices of the triangulation received from the standard input to the defined implicit surface and write the new result to the standard output. Configuration parameters can be defined in ‘landing.pot’ file (or in the file specified with ‘--pot’ option) and overridden in the command line.

```
landing  '-p,--pot' configuration file .pot
        '-h --help'
```

### Configuration Options

Name	Default	Description
file		Name of the .cnt coefficients file
smoothing	2	Smoothing factor for the implicit surface
landing/precision	1e-3	precision for the Hartmann foot point algorithm
landing/intersection	no	if yes program will check and avoid self intersection

### Customization Options (*For advanced users only*)

Name	Default	Description
implicit_func_module		Module containing the regular function for the surface
landing/module	landExt/harmannFootPoint.so	file .so with the foot point algorithm
landing/foot_point	tol_foot_point	function that implements the foot point alg.
implicit_surface_key	implicit_surface	hash table key name of the implicit surface object

### Example

To project on the implicit surface the ‘ck2Umb.gts’ produced in the previous example type this command: `landing < ck2Umb.gts > ck2UmbLand.gts`

## 8 Coarse

It simplifies the triangulation received from the standard input by means of edge collapsing, according to the metric produced from the minimum acceptable error specified by the user and writes it to the standard output. Configuration parameters can be defined in ‘coarse.pot’ file (or in the file specified with ‘--pot’ option and overridden in the command line).

```
coarse    -p,-pot configuration file .pot
          -h -help
```

### Configuration Options

Name	Default	Description
file		Name of the .cnt coefficients file
smoothing	2	Smoothing factor for the implicit surface
coarse/oogl		.oogl output filename
coarse/foot_point/precision	1e-3	precision for the Harmann foot point algorithm
coarse/metric/error	1e-4	geometric error used to build the Frey metric

### Customization Options (*For advanced users only*)

Name	Default	Description
implicit_func_module		Module containing the regular function for the surface
<b>Prefix [/]</b>		
implicit_surface_key	implicit_surface	hash table key name of the implicit surface object
user_metric_key	user_metric	hash table key name of the user metric function
<b>Prefix [/coarse/]</b>		
foot_point/module	landExt/hartmannFootPoint.so	file .so with the foot point alg. extension
metric/module	metricExt/freyMetricTools.so	file .so with the metric extensions
metric/vertex_metrify	tol_vertex_metrify	function that build the vertex metric
metric/edge_metrify	tol_edge_metrify	function that build the edge metric
algo/module	algoExt/tolCoarseAlgo.so	file .so with the coarse algorithm extensions
algo/edge_cost	tol_metric_length_cost	edge cost function
algo/edge_stop	tol_until_edges_are_short	stop function
algo/edge_middle_vertex	tol_middle_vertex_on_surface	new vertex generator function
algo/edge_collapse	tol_edge_collapse	local coarse algorithm function
foot_point/foot_point_key	foot_point	hash table key for the foot point function object

## Example

At this point it's possible to coarsen the surface mesh described by the file 'ck2UmbLand.gts', according with the metric generated with reference geometric error specified by the parameter *coarse/metric/error*. To do this type the command: `coarse < ck2UmbLand.gts > ck2UmbLandCoa.gts`

## 9 Refine

It refines the triangulation received from the standard input by means of vertex insertion, according to the metric produced from the maximum acceptable error specified by the user and writes it to the standard output. Configuration parameters can be defined in the ‘refine.pot’ file (or in the file specified with ‘--pot’ option and overridden in the command line).

```
refine    '-p,--pot' configuration file .pot
          '-h --help'
```

### Configuration Options

Name	Default	Description
file		.cnt file
smoothing	2	smoothing factor
refine/oogl		.oogl output filename
refine/foot_point/precision	1e-3	precision for the Harmann foot point algorithm
refine/metric/error	1e-4	geometric error used to build the Frey metric

### Customization Options (*For advanced users only*)

Name	Default	Description
implicit_func_module		Module containing the regular function for the surface
<b>Prefix [/]</b>		
implicit_surface_key	implicit_surface	hash table key name of the implicit surface object
user_metric_key	user_metric	hash table key name of the user metric function
<b>Prefix [/refine/]</b>		
foot_point/module	landExt/hartmannFootPoint.so	file .so with the foot point alg. extension
metric/module	metricExt/freyMetricTools.so	file .so with the metric extensions
metric/vertex_metrify	tol_vertex_metrify	function that build the vertex metric
metric/edge_metrify	tol_edge_metrify	function that build the edge metric
cost/module	algoExt/tolRefineAlgo.so	file .so with the refine cost and stop extensions
algo/edge_cost	tol_metric_inv_length_cost	edge cost function
algo/edge_stop	tol_until_edges_are_long	stop function
algo/module	algoExt/tolRefineAlgo.so	file .so with the refine algorithm extensions
algo/edge_middle_vertex	tol_middle_vertex_on_surface	new vertex generator function

<code>algo/edge_split</code>	<code>tol_middle_vertex_insertion</code>	local refine algorithm function
<code>foot_point/foot_point_key</code>	<code>foot_point</code>	hash table key for the foot point function object

## Example

At this point it's possible to refine the surface mesh described by the file '`ck2UmbLand.gts`', according with the metric generated with reference geometric error specified by the parameter `refine/metric/error`. To do this type the command: `refine < ck2UmbLandCoa.gts > ck2UmbLandCoaRef.gts`

## 10 Swap

It swaps the edges of the triangulation received from the standard input according to the quality criterion defined by the user and writes the resulting triangulation to the standard output. Configuration parameters can be defined in the ‘swap.pot’ file (or in the file specified with ‘--pot’ option and overridden in the command line.

```
refine    '-p,--pot' configuration file .pot
          '-h --help'
```

### Configuration Options

Name	Default	Description
file		.cnt file
smoothing	2	smoothing factor
swap/metric/error	1e-4	geometric error used to build the Frey metric

### Customization Options (*For advanced users only*)

Name	Default	Description
implicit_func_module		Module containing the regular function for the surface
<b>Prefix</b> [/]		
implicit_surface_key	implicit_surface	hash table key name of the implicit surface object
user_metric_key	user_metric	hash table key name of the user metric function
<b>Prefix</b> [/swap/]		
metric/module	metricExt/freyMetricTools.so	file .so with the metric extensions
metric/vertex_metrify	tol_vertex_metrify	function that build the vertex metric
metric/edge_metrify	tol_edge_metrify	function that build the edge metric
cost/module	algoExt/tolRefineAlgo.so	file .so with the swap cost and stop extensions
algo/edge_cost	tol_metric_inv_length_cost	edge cost function
algo/edge_stop	tol_until_edges_are_long	stop function
algo/module	algoExt/tolRefineAlgo.so	file .so with the swap algorithm extensions
algo/func	tol_edge_swap	swap function name

## 11 tolCut

It cuts the surface by means of a cutting plane (actually a cutting triangle) in order to isolate the closed volume for the simulation. The result of the intersection of 2 surfaces are 4 surfaces. The cutted and closed surface written on the standard output is the union of 2 of these surfaces, the ones to choose depends on the surfaces orientation (see *which* parameter).

```
tolCut    '-p,--pot' configuration file .pot
          '-h --help'
```

Name	Default	Description
xn, yn, zn		Cordinates of the cutting triangle
precision	1	Index of refinement for the triangle. Increase this for better result
which	in1in2	Which of the surfaces produced by the intersection the tool should give as result. Values are: <i>in1in2</i> , <i>in1out2</i> , <i>out1in2</i> , <i>out1out2</i>
prefix		Prefix for the output files. If none files are not produced
color1	-1	Color for the first surface generated by the intersection. If -1 the color attribute will be unchanged
color2	-1	Color for the second surface generated by the intersection. If -1 the color attribute will be unchanged

### Example

We now produce the closed mesh that describe the vascular truck under analisis. To do this we must cut the surface with two cutting plane.

```
tolCut < ck2UmbLandCoaRef.gts > cut1.gts
tolCut --pot tolCut2.pot < cut1.gts > cut2.gts
```

These two commands produce the expected result. The mesh is now complete. In the gts file the three differents parts of the surface mesh (vascular boundary, cutting plane 1, cutting plane 2) are marked with differents colors. This is useful if we want to assign different boundary conditions.

## 12 Improving cutting surface quality

The pieces of the triangulation generated by the cutting algorithm are often affected by poor quality. It is possible to improve the quality of these parts of the mesh by refining and swapping on a selected portion of the surface only.

### 12.1 Improving quality by refining

This can be done by the `refine` tool using the `'tolRefineCutNum.pot'` or `'tolRefineCutNum.pot'` as reference. As you can see, these files make the tool to operate on a particular part of the mesh, namely that defined by the `'refine/color'`. It uses an euclidean metric (it is expected that the tool works on a plane).

### 12.2 Improving quality by swapping

This can be done using the `swap` tool. Two configuration files are provided as example: `'swapTopo.pot'` that work on the optimization of the mesh topology, `'swapEucl.pot'` that work on the optimization of the aspect ratio computed with an euclidean metric.

#### Example

Now we improve the quality of the cutted surface by means swap driven by topology:

```
swap --pot swapTopo.pot swap/color=2 < cut2.gts > c2s.gts
```

```
swap --pot swapTopo.pot swap/color=3 < c2s.gts > cs.gts
```

It is possible to refine the resulted triangulation:

```
refine --pot refineCutMet.pot refine/color=2 < cs.gts > cs2r.gts
```

```
refine --pot refineCutMet.pot refine/color=3 refine/metric/scale=0.2 <  
cs2r.gts > csr.gts
```

## 13 Converting GTS to OOGL (Geomview) or SURF (Netgen)

The output format of the mesh is a “GTS++ standard”. To each face of the triangulation a color identifier is added. In this way it is possible to distinguish among different parts of the triangulation.

### tolGts2Oogl

This tool converts a GTS triangulation received from the standard input to OOGL format for Geomview and writes it to the standard output.

```
tolGts2Oogl
    '-p,--pot' configuration file .pot
    '-h --help'
```

### Configuration Options

Name	Default	Description
mapFile	colorMap.tol	file name of the color map

The color map defines the translation of the color code form the GTS file to the OOGL format. This is an example of the color map file:

```
1 1 0 0 1
2 0 0 1 1
3 1 1 1 1
```

One row for each color identifier of the GTS file: the first number is the color identifier, followed by a RGB triplet and a transparency coefficient (for more info check the GEOMVIEW documentation).

### tol2netgen

This tool converts a GTS triangulation received from the standard input to an SURF format for Netgen (<http://www.hpfem.jku.at/netgen/>) and writes it to the standard output.

```
tolGts2Oogl
```

## 14 Modular Architecture

Each of the TOL Tools is designed to be customizable by the user. This is an open source project so, user can effectively customize and improve every single line of code! However if the main behavior of a tool fits the user goal, a modular architecture is provided to modify functionalities without the need to rebuild all the application (e.g. If you like the refine algorithm, but you need to define your own metric, you can do that just writing a metric module). Custom functionalities are separated from the main application and organized in a set of shareables objects, located in the ‘\*Ext’ (Ext means extension) subdirectories of the tools directory.

The main program interacts whit the extension modules in several ways:

- passing the configuration parameters;
- passing objects or complex data structures created at runtime;
- calling function implemented in the extension.

The first two steps are accomplished in the initialization of the extension module. Each module must contain an implementation of *void tol\_configure(GetPot \*, GHashTable \*)* function, that executes the module configuration actions using two parameters: the GetPot object will contain the start options (from the ‘.pot’ file), the GHashTable will contain the objects created runtime by the main program. These parameters are provided by the main program when it calls *tol\_configure()*. Objects stored in the hash table are accessible through hash keys specified in the ‘.pot’ file.

A good example on how to access parameters passed by the main program is the *tol\_configure()* of the metric module ‘metricExt/freyMetricTool.cc’. This metric implementation needs the implicit surface object, the reference error and a user metric in order to work properly. This is the piece of code that configures the metric module:

```
...
static TolImplicitSurface *tis=0;
static TolDouble          eps;
static UserMetricT        userMetric;

/** Configure the module */
int tol_configure(GetPot *uGp,GHashTable *uConfig)
{
    tis = g_hash_table_lookup(uConfig,"implicit_surface");
    eps = (*uGp)("metric/error",0.0);
    userM= (UserMetricT)g_hash_table_lookup(uConfig,"user_metric");
    return 0;
}
...
```

As results of the call to `tol_configure` the metric module can now access the required objects as static variables.

This way of doing enforce flexibility, since a user has the possibility to define every types of data structures in its own module, and performance, because algorithms can access,

during the execution time, the configuration parameters as static variables (with little computational overhead). Having completed the configuration procedure the module is ready to be used.

When the main tool requires a service implemented by an extension module, it calls the function specified in the configuration file for that service: as example the `landing` tool needs an implementation of a foot point algorithm, in the default distribution this algorithm is based on the Hartmann foot point for implicit surface. The implementation of this algorithm is in the `'landExt'` directory and the name of the function is specified in the `'pot'` file. User can implement modules with its own implementation and let the tool call them specifying the module name in the `'pot'` file (the signature of the functions cannot be changed).

## Example

Let's suppose that we want to equip the `refine` with an euclidean metric scaleable by a factor *scale*.

We need to do the following:

1. implement `tol_configure()`;
2. implement `tol_vertex_metrify(gpointer, gpointer)`;
3. implement `tol_edge_metrify(gpointer, gpointer)`;
4. add to `'refine.pot'` the new parameter for the scale factor;
5. modify in the `'refine.pot'` file the `refine/metric/module` to point to the new module;

An example of `tol_configure()` could be the following:

```
static double scale;

/** Configure the module */
int tol_configure(GetPot *uGp,GHashTable *uConfig)
{
    scale = (*uGp)("metric/scale",1.0);
    return 0;
}
```

This implementation read from the `GetPot` object the scale factor and save it into a static variable.

Then we need an implementation of `tol_vertex_metrify(gpointer, gpointer)` that calculates the metric matrix for a given triangulation's vertex and an implementation of `tol_edge_metrify(gpointer, gpointer)` that calculates the length of the edge.

```
int tol_vertex_metrify(gpointer item, gpointer data)
{
    TolGtsMetricVertex *uV= MOX_GTS_METRIC_VERTEX(item);
    TolMatrix &m=(TolMatrix)(*uV->metric);
    m[0][0]=scale*scale;m[1][1]=scale*scale;m[2][2]=scale*scale;
    m[0][1]=m[0][2]=m[1][2]=0;
    m[1][0]=m[2][0]=m[2][1]=0;
    return 0;
}
```

```

}

int tol_edge_metrify(gpointer item, gpointer data)
{
    GtsEdge *edge= (GtsEdge *)item;
    TolGtsMetricEdge *me= MOX_GTS_METRIC_EDGE(edge);
    TolGtsMetricVertex *mv1= (MOX_GTS_METRIC_VERTEX(edge->segment.v1));
    TolMatrix *m1= (TolMatrix *) (mv1->metric);

    TolDouble x1= edge->segment.v1->p.x;
    TolDouble y1= edge->segment.v1->p.y;
    TolDouble z1= edge->segment.v1->p.z;
    TolDouble x2= edge->segment.v2->p.x;
    TolDouble y2= edge->segment.v2->p.y;
    TolDouble z2= edge->segment.v2->p.z;

    TolVector l(x2-x1,y2-y1,z2-z1);
    TolVectorRow lr(l.getTranspost());

    //
    // Computation of the edge length according
    // to the metric
    //
    me->length = std::sqrt(lr>(*m1)*l);
    return 0;
}

```

Of course we could complain that this is a solution with poor performance and decide to provide an implementation with an empty `tol_vertex_metrify(gpointer, gpointer)`

```

int tol_vertex_metrify(gpointer item, gpointer data)
{
    return 0;
}

int tol_edge_metrify(gpointer item, gpointer data)
{
    GtsEdge *edge= (GtsEdge *)item;

    TolDouble x1= edge->segment.v1->p.x;
    TolDouble y1= edge->segment.v1->p.y;
    TolDouble z1= edge->segment.v1->p.z;
    TolDouble dx= edge->segment.v2->p.x-x1;
    TolDouble dy= edge->segment.v2->p.y-x2;
    TolDouble dz= edge->segment.v2->p.z-x3;
}

```

```

    me->length = scale*std::sqrt(dx*dx+dy*dy+dz*dz);
    return 0;
}

```

Now we must wire together the extension module with the `refine` tool. Let us suppose that our implementation is contained in a file named `'euclideanMetric.cc'` and compiled into `'euclideanMetric.so'` We need to change according the parameter `refine/metric/module` and add the new parameter for the scaling factor:

```

refine/metric/module = metricExt/euclideanMetric.so
refine/metric/scale = 2.0

```

Done! The `refine` tool can now be run with the new metric!

## 14.1 Changing regular function

Another module the user may want to customize is the form of the implicit surface.

ETOL was initially designed to work with RBF, so the default regular function used is linear radial basis function. However, in order to work properly, the application only need a function equipped with first and second derivatives (this come from the fact that the Frey metric is function of the curvatures of the surface). The modular framework can help the user to define its own regular function, as example the steps describe below show how to define an ellipsoidal surface implicitly defined.

### Subclass TolRegularFunction

First of all the regular function must be implemented:

```

#ifdef __cplusplus
extern "C"
{
#endif

class MyEllipsoid : public TolRegularFunction
{
private:
    TolDouble a,b,c;
public:
    MyEllipsoid(TolDouble uA, TolDouble uB, TolDouble uC)
        : a(uA),b(uB),c(uC) {}
    TolDouble f(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return a*a*uX*uX+b*b*uY*uY+c*c*uZ*uZ; }
    TolDouble fx(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 2*a*a*uX; }
    TolDouble fy(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 2*b*b*uY; }
    TolDouble fz(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 2*c*c*uZ; }
    TolDouble fxx(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 2*a*a; }
    TolDouble fxy(TolDouble uX, TolDouble uY, TolDouble uZ)

```

```

    { return 0; }
    TolDouble fxz(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 0; }
    TolDouble fyx(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 0; }
    TolDouble fyy(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 2*b*b; }
    TolDouble fyz(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 0; }
    TolDouble fzx(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 0; }
    TolDouble fzy(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 0; }
    TolDouble fzz(TolDouble uX, TolDouble uY, TolDouble uZ)
    { return 2*c*c; }
    void getBBox(TolDouble &uX1, TolDouble &uY1, TolDouble &uZ1
                , TolDouble &uX2, TolDouble &uY2, TolDouble &uZ2)
    { uX1=-1/a; uY1=-1/b; uZ1=-1/c;
      uX2=-uX1; uY2=-uY1; uZ2=-uZ1;
    }
};
#ifdef __cplusplus
}
#endif

```

*Implementing the support functions* Then two functions must be implemented `tol_configure()` and a factory function `get_function()`:

```

...
static TolDouble a,b,c;

/** Configure the module */
int tol_configure(GetPot *uGp,GHashTable *uConfig)
{
    a = (*uGp)("axis_coefficients/a",1.0);
    b = (*uGp)("axis_coefficients/b",1.0);
    c = (*uGp)("axis_coefficients/c",1.0);
    return 0;
}
...
TolRegularFunction *get_function()
{
    return new MyEllipsoid(a,b,c);
}

```

In order to be dynamically loaded into the main program, the custom modules must be compiled correctly, as example in a Linux environment with gcc compiler the ‘-shared’ must be used. User can refer to the ‘Makefile.am’ of the ‘\*Ext’ directories.

ETOL libraries and ETOL tools use GLib and GetPot, more information about these components can be found at: [GLib Reference Manual](#) and [GetPot Home Page](#).

# Appendix A Copying This Manual

## A.1 GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.  
59 Temple Place, Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

### 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

### 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

- be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
  - C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
  - D. Preserve all the copyright notices of the Document.
  - E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
  - F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
  - G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
  - H. Include an unaltered copy of this License.
  - I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
  - J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
  - K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
  - L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
  - M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
  - N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
  - O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

### A.1.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C) year your name.  
Permission is granted to copy, distribute and/or modify this document  
under the terms of the GNU Free Documentation License, Version 1.2  
or any later version published by the Free Software Foundation;  
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.  
A copy of the license is included in the section entitled ‘‘GNU  
Free Documentation License’’.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being list their titles, with  
the Front-Cover Texts being list, and with the Back-Cover Texts  
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

## Index

FDL, GNU Free Documentation License . . . . . 23