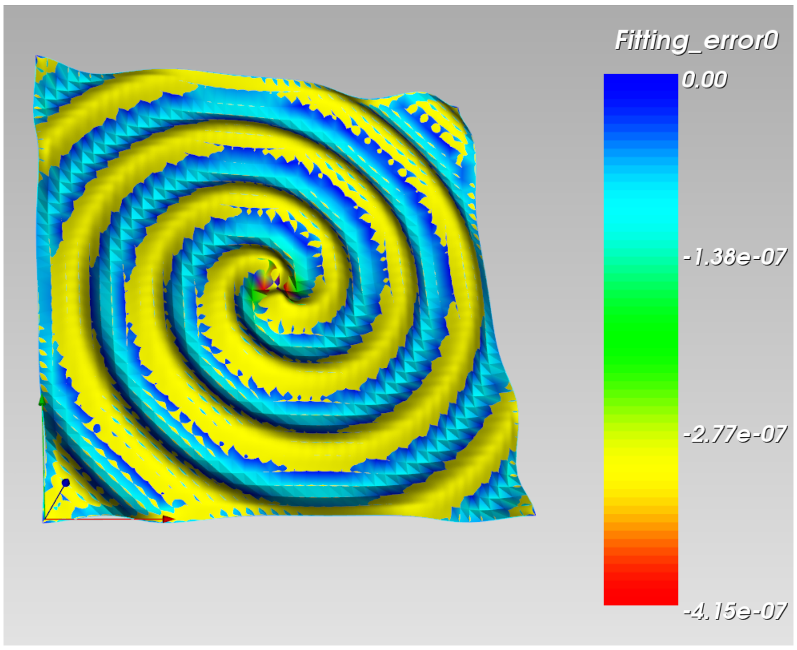
Axel fitting process user manual

[Nicolas Douillet ; last update : 04/08/2014]



1 Purpose

The goal of this process is to provide an efficient, reliable, and adaptable algorithm to fit a given data surface $(P_i)_{1\leq i\leq N}$ with a B-spline surface $\sigma(u, v)$.

2 Fitting process general principle (figure 1)

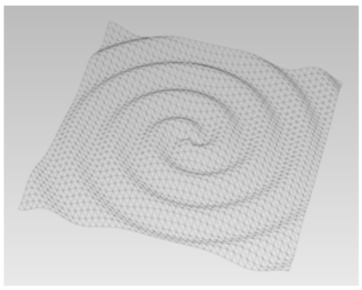
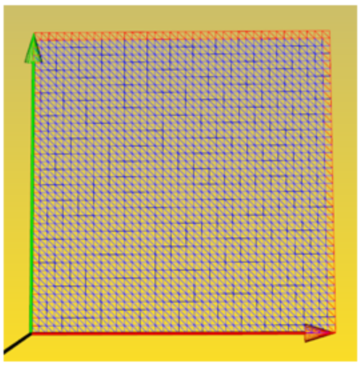
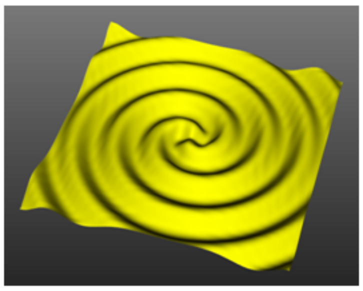
(1) 4 edges surface mesh (input data)	(2) Parameter space (secondary hidden input)	(3) Fitting surface (output)
$(p_i)_{1\leq i\leq N} = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix}_{1\leq i\leq N} \in \mathbb{R}^{3N}$ 	$M_i(u_i, v_i)_{1\leq i\leq N} \in [0,1]^{2N}$ 	$\sigma(u,v) = \sum_{j=1}^{D} C_j B_j(u,v)$ 

Figure 1:

2.1 Isoparameterization of a collection of patches

We recall here the problem we consider and detail in our approach. The description of manufactured objects in CAGD is based on B-spline or NURBS models. These are parameterized representations which are usually trimmed and assembled to describe the boundary of a volume. To control (five-axis) milling machines that will manufacture the corresponding objects, a important task is to compute the global parametric representation of the boundary surface. This allows for instance to compute efficiently the complete pathof the milling machine. Here is our setting. We denote by $S = \cup_i S_i$ the surface of the tridimensional space described by a collection of trimmed NURBS patches S_i with their adjacy relations. We assume moreover that 4 curves $\Gamma_1, \ldots, \Gamma_4$ are given which form the boundary ∂S of S. We assume that $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ are such that $\partial S = \cup_{i=1}^{4} \Gamma_i$, Γ_i and Γ_{i+1} intersect in one point, Γ_i and Γ_{i+2} do not intersect, for $1 \leq i \leq 4$ (with the convention $\Gamma_j = \Gamma_{j-4}$ if $j \geq 4$).

The aim is to construct an injective parameterization from the unit square $D = [0, 1] \times [0, 1]$ to the surface S :

$$\begin{aligned} &\sigma : D \to S \\ &(u,v) \to \sigma(u,v) = (x(u,v), y(u,v), z(u,v)) \end{aligned}$$

which is bijective and of regularity at least C^1 and such that

$$\begin{aligned} \sigma([0,1],0) &= \Gamma_1 \\ \sigma(0,[0,1]) &= \Gamma_2 \\ \sigma([0,1],1) &= \Gamma_3 \\ \sigma(1,[0,1]) &= \Gamma_4 \end{aligned}$$

The general approach we have followed can be decomposed in two steps :

- Construct (an approximation of) $\sigma^{-1} : S \to D$ which maps S into a square D ;
- Fit σ by a B-spline representation from D to S.

2.1.1 Mapping a 3-D surface into a square

We now detail our approach to construct a bijective map $(u,v) : S \to D$ transforming S into the square $D = [0,1] \times [0,1]$. To guarantee the injectivity of this map, we look for coordinate functions u, v which satisfy the Laplace-Beltrami equation on S :

$$\begin{aligned} \Delta_S(u) &= 0 \\ \Delta_S(v) &= 0 \end{aligned}$$

with some boundary conditions of the form :

$$\begin{cases} u = 0 \text{ on } \Gamma_1, u = 1 \text{ on } \Gamma_3, u = u_2 \text{ on } \Gamma_2, u = u_4 \text{ on } \Gamma_4. \\ v = 0 \text{ on } \Gamma_2, v = 1 \text{ on } \Gamma_4, v = v_1 \text{ on } \Gamma_1, v = v_3 \text{ on } \Gamma_3. \end{cases}$$

where v_1, u_2, v_3, u_4 are given functions, monotonous between 0 and 1 on $\Gamma_1, \ldots, \Gamma_4$. The solutions u, v of these Laplace equations are Harmonic functions. They are minimizing the Dirichlet energy :

$$E(u,v) = \int_S \|grad_S(u)\|^2 + \|grad_S(v)\|^2 dS$$

under the given boundary conditions.

A first result to ensure that the coordinate map $f = (u, v)$ is bijective is given by the Rado-Kneser-Choquet theorem :

Theorem *If $S \subset \mathbb{R}^2$ is a planar domain and $f : S \to C$ is an harmonic map from S to a convex planar domain C which maps ∂S homeomorphically to ∂C, then f is bijective.*

A classical approach to construct or to approximate a function that minimizes the Dirichlet energy is to fix a finite dimensional function space in which we will represent our coordinate functions as linear combinations of a given basis. A typical example isto approximate the surface S by a mesh supported on a set of vertices $p_1, \ldots, p_N$ and to approximate f by a continuous picewise linear function on this mesh, that is

$$f = \sum_{i=1}^{N} f_i \Phi_i$$

where Φ_i is the picewise linear function which is 1 at the vertex p_i of the mesh and 0 at all other vertices. The Dirichlet boundary condtions are constructed as follows : the distance from a point on a boundary curve is computed and normalized to define a function which is 0 at the begining of the curve and 1 at the end. These functions on each side of the surface are used to specify the non-constant boundary conditions of the Laplace equations. Then, minimizing the Dirichlet energy under the given boundary conditions leads to a linear system of the form

$$f_i = \sum_{j \in N_i} \lambda_{i,j} f_j$$

where

- f_i is fixed by the boundary conditions if p_i is a boundary vertex,
- N_i is the set of indices $j \neq i$ of the vertices p_j of the mesh, which are connected to p_i by an edge of the mesh,
- $\lambda_{i,j} = \frac{\omega_{i,j}}{\sum_{k \in N_i} \omega_{i,k}}$ so that $\sum_{j \in N_i} \lambda_{i,j} = 1$.

It is shown in [1] that such a map is injective as soon as $\lambda_{i,j} \geq 0$ and $\sum_{j \in N_i} \lambda_{i,j} = 1$. In our experiments, we consider the following constructions :

2.1.2 Constructing the B-spline parameterization

At this step of the algorithm, we suppose we are given a collection of points $p_i \in \mathbb{R}^3$ and corresponding parameter values $(u_i, v_i) \in D$ for $i = 1, \ldots, N$. We are going to construct the parameterization map $\sigma : D \to \mathbb{R}^3$ by solving the interpolation problem

$$\sigma(u_i, v_i) = p_i, \; i = 1, \ldots, N \tag{1}$$

2.1.3 Specification

Here is the formal description of the algorithm :

Algorithm : ISOPARAMETERIZATION OF A COLLECTION OF PATCHES

Input : A surface $S \subset \mathbb{R}^3$ given by a collection of trimmed NURBS surfaces, with 4 boundary curves.

Output : A B-spline surface parameterizing globally an approximation of S

- Compute a mesh $\tilde{S}$ approximating the surface S ;
- Define a map σ from $\tilde{S}$ to the square $[0, 1] \times [0, 1]$,satisfying boundary conditions ;
- Construct the best spline function from $[0, 1] \times [0, 1]$ to $\mathbb{R}^3$ which approximates the discrete map σ^{-1}.

Step 1 requires the evaluation of the initial parametric model at different parameter pointsand the connectivity of these points of the surface S. It is described by the mesh $\tilde{S}$. The possibility to use only a point set representation for S will be investigated.

Step 2 requires the resolution of a sparse system which size is related to the number of points producted in step 1. This computation is used to define a correspondance between parameter points in a planar domain and points on the surface S.

Step 3 requires the resolution of a linear system for the optimal solution of a fitting problem. The size of the problem depends on the number of (spline) basis functions used to represent the output surface. An adaptation strategy can be considered to refine the representation in regions where the error is too important.

3 Methods and algorithms

3.1 The 3 methods used for the computation of the parameter mesh

3.1.1 Tutte [2]

$$\omega_{ij} = 1 \tag{2}$$

3.1.2 Harmonic / cotangent [3]

$$\omega_{ij} = \text{cotan}(\alpha_{ij}) + \text{cotan}(\beta_{ij}) \tag{3}$$

Where α_{ij}, β_{ij} are the angles of the triangle adjacent to the edge (p_i, p_j), at the vertices distinct from (p_i, p_j).

3.1.3 Mean value / tangent [4]

$$\omega_{ij} = \frac{\tan\left(\frac{\gamma_{ij}}{2}\right) + \tan\left(\frac{\gamma_{ij}^{'}}{2}\right)}{\|p_i p_j\|} \tag{4}$$

3.2 The 3 methods used for the computation of the error ([5], [6])

Performances curves comparison is given on figure 2.

3.2.1 PDM : Point Distance Method

This method uses the basic expression of the distance between a data point p_i and its corresponding approximated point $\sigma(u_i, v_i)$ on the B-spline surface to compute the error, ϵ :

$$\epsilon_{PD} = \frac{1}{2}\sum_{i=1}^{N} \|\sigma(u_i, v_i) - p_i\|^2 \tag{5}$$

3.2.2 TDM : Tangent Distance Method

This computation uses the normal vector $n_i = n(u_i, v_i)$ of the B-spline surface σ to reduce the distance and the error :

$$\epsilon_{TD} = \frac{1}{2}\sum_{i=1}^{N} \|(\sigma(u_i, v_i) - p_i) \cdot n_i\|^2 \tag{6}$$

3.2.3 SDM : Squared Distance Method

This last method combines all together the normal vector $n_i = n(u_i, v_i)$ of the B-spline surface σ and the tangent vectors in u : $t_{u,i} = \partial_u\ \sigma(u_i, v_i)$ and v : $t_{v,i} = \partial_v\ \sigma(u_i, v_i)$ directions to improve the error expression :

$$\epsilon_{SD} = \frac{1}{2}\sum_{i=1}^{N} \frac{d}{d+\rho_{i,1}} \|(\sigma(u_i, v_i)) - p_i \cdot t_{u,i}\|^2$$
$$+ \frac{d}{d+\rho_{i,2}} \left[\|(\sigma(u_i, v_i)) - p_i \cdot t_{v,i}\|^2 + \|(\sigma(u_i, v_i)) - p_i \cdot n_i\|^2\right]$$

where d is the distance between p_i and its foot point on the current surface, $\rho_{i,1}$ and $\rho_{i,2}$ are absolute values of the principal curvature radii.

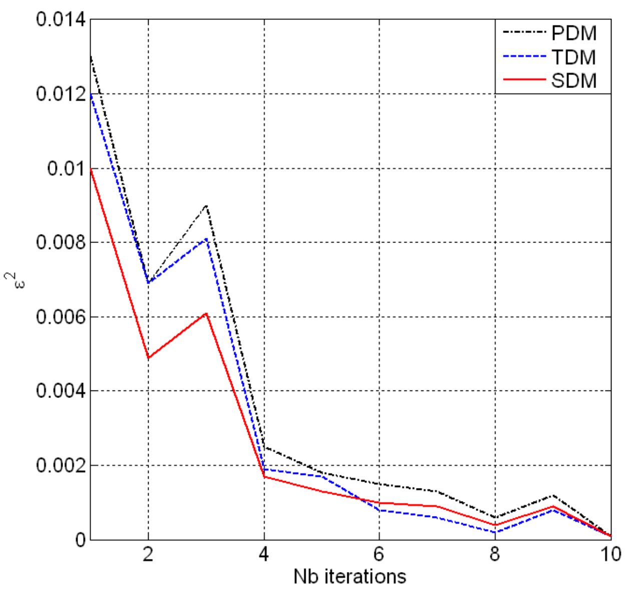


Figure 2: PDM, TDM, and SDM error methods convergence comparison on a surface exemple (archimedian spiral).

3.3 The 3 methods used for the computation of the fitting surface

3.3.1 Direct inversion of the system matrix (global method)

Since the B-spline fitting surface we are looking for can be written as a weighted sum of 2D B-spline basis :

$$\sigma(u,v) = \sum_{j=1}^{D} c_j B_j(u,v) \tag{7}$$

where $(B_j)_{1\leq j \leq D}$, is the 2D basis function used to represent σ, minimizing ϵ leads in its principle to solve a linear system of the form :

$$AX = b \tag{8}$$

where X is the three column vector (X, Y, Z) of the unkown D control points coordinates,

$$A_{n,p} = \sum_{i=1}^{N} B_n(u_i,v_i) B_l(u_i,v_i) \quad 1 \leq n,p \leq D \tag{9}$$

is the system matrix and

$$b_n = \sum_{i=1}^{N} p_i B_n(u_i, v_i) \quad 1 \leq n, p \leq D \tag{10}$$

is the "constraints" vector.

3.3.2 "Edges fit"

The philosophy of this method is to notice the importance of the surface corners and edges to compute a good fitting surface. In this algorithm, the four edges and the inside surface are then processed separately.

3.3.3 Coons patch / barycentric patch + iterative optimization (figure 3)

This method consists in 5 steps :

- (1) : Extraction of the edges data points,
- (2) : Computation of each corresponding parameters vectors using curvi-linear abscissa,
- (3) : B-spline curve fitting of these 4 edges using the parameter mesh (figure 3.2),
- (4) : Computation of a first inside B-spline surface -Coons patch [7] (figure 3.4) or barycentric patch- using the edges control points coordinates computed at the previous step,
- (5) : Iterate on the inside surface, with recomputing at each step the inside parameter mesh with closest point method, to estimate a new B-spline surface to fit the data mesh (figure 3.6).

4 Dependancies

4.1 Context

Since *Axel* is based on INRIA *dtk* platform, all the *Axel* objects classes actually inherit from *dtk* ones. This means an *axlmesh* object or a *SplineSurface* object which are also some *axlAbstractData* objects are also, at the most general level, some *dtkAbstractData* objects.

4.2 C++ libraries dependancies

Since it is part of Axel, the *FittingParamProcess* use extensively *Qt* objects (mainly in the dialog process), but also *Eigen* library (in the fitting core computations), and of course the *std*, standard C++ library.

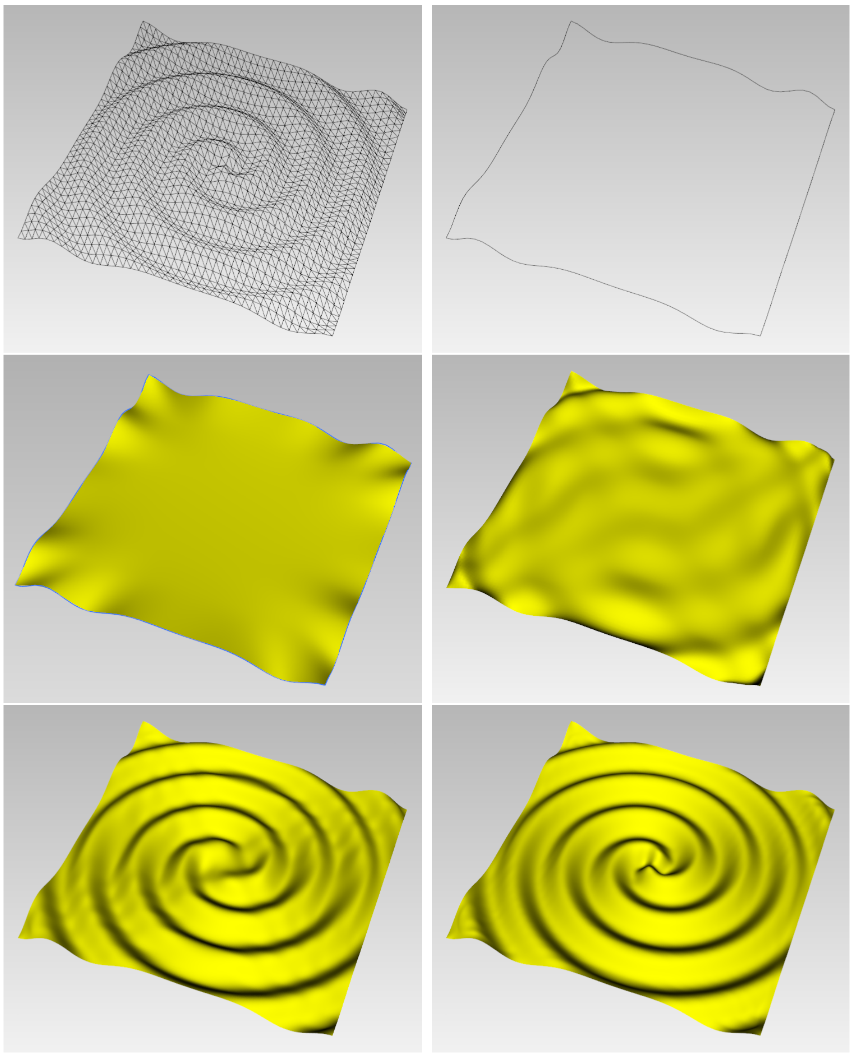
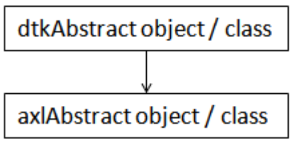


Figure 3: 1 : input data mesh / surface to fit. 2 : its boundaries. 3 : resulting B-spline surface after the curve fitting of the four edges only. 4 : the Coons patch (provides the first parameterization) computed from the previous surface. 5 : resulting B-spline surface computed after one projections of the Coons patch. 6 : resulting B-spline surface computed after two projections of the Coons patch.



4.3 Other Axel plugins and process dependancies

The source code is located in :

/paramtools/include/paramtools and in /paramtools/axl/IgaParam.

Inside paramtools, the *FittingParamProcess* calls the *MeshParamProcess* at its very begining to compute its second -private- input : the parameter mesh.

Inside Axel, the *FittingParamProcess* depends of course on *Axel-sdk* which is the core of Axel, but also a lot on *Bsplinetools* (which lays itself on *Gotools*, a geometric C++ library from the SINTEF), and a little on *Solitools* plugin.

5 Using the *FittingParamProcess* in *Axel*

In *Axel* objects inspector (open with CTRL+I), select the "objects" tab and then click on the data mesh you want to fit. The object should appear highlighted and labelled "selected" or "editable" (not "passive"). Then select the "tools" tab, look in it for the "MeshParam-Process" and click on it. Then, in the drop down menu which just appear below, double click on "fitting". Now, just tune each parameter (section 7) in the menu below, and once done, run the fitting process by clicking the run button at the bottom of this dialog process.

6 Input and output

The *FittingParamProcess* takes a data *axlMesh* as its unique input. This mesh must have **4 edges**. As an output it returns an *axlAbstractSurfaceBSpline* object.

7 Parameters

Table 1 provides a list of *Axel* surface fitting process parameters, their types, range values, and default settings.

7.1 Influence and tuning

- *(nbf_u, nbf_v)*, the numbers of control points -basis functions in u and v directions- are automatically computed at the begining of the process taking into account geometrical properties, i.e the dimensions, of the surface. The default density is one control points for each four data vertices in a row. At each step of the algorithm, these numbers are doubled in order to reduce ϵ. Figure 4 gives the evolution of this error on a given example surface as a function of the number of basis,

- *nb_it*, the number of iterations runs the number of times the fitting algorithm will be called on the same data surface, doubling each time the number of basis in u and v. Since this grows very quickly, this parameter is limited to a small integer to avoid the process to freeze,

- *Epsilon* (ϵ) is the final absolute precision / error on the surface wished by the user, expressed in *Axel* graphical unit. Since reaching this value is one of the two conditions to break the fitting algorithm loop (the other beeing have reached the maximun number of iterations), it is good to set it small enough to avoid it to be a limiting parameter,

- *paramethod*, the parameterization methods are described in 3.1. The first one (Tutte) appears to be the most reliable, but also the less precise, whereas this is the opposite for the tangent and cotangent methods which are more sensitive to the quality of the input data mesh,

- *err_method*, the error method (see 3.2 and figure 2),

- *opt_method*, the approximation / optimization method (detailed in 3.3) decides of the generic approach to fit the discrete $(P_i)_{1 \leq i \leq N}$ data surface. The edges fitting method gives more precision on the edges for a same number of iterations but is also almost twice slower than the global method in term of computational cost. That is the reason why its use should be restricted to quick fits with a low number of iterations on surface which edges are difficult to fit,

- λ, the multiplying coefficient of the laplacian correction matrix : to help A, the system matrix (equation 9), to be invertible we sometimes need to add a "laplacian correction matrix", M, (figures 5 and 6) such that $A' = A + \lambda M$. This coefficient is quite sensitive to adjustate and may have a big influence on the resulting fitting surface since it impacts on the whole system. Basically, it is used to add some "tension" between the control points of the B-spline surface, in order to remove some "wave effects". Thus it makes the surface smoother, from its outside (edges and corners) toward its inside. Too small λ has no effect, but too big, it tends to modify the shape of the B-spline surface. In the case of the edges fit method, it has no effect on the edges and corners control points since once computed those ones stay static at a given step of the algorithm. In the case of the global method, λ influences all the control points but the four corners ones which stay stuck to the data surface corners points. Switching from the global method to the edges fitting method, it is necessary to take a λ ten times, or even hundred times bigger.

8 Performances and limitations

8.1 Data set and parameterization

Since the fitting process is based on these two meshes (input data surface and its corresponding parameters mesh), and since the parameters mesh is built only from the data, the quality of the process output -the resulting B-spline surface- entirely depends on it. That is the reason why, to assure good fitting results, the input data mesh should always be a conformal mesh, preferably without hole, neither flat triangle, nor duplicated vertex.

8.2 Triangular surfaces or more than 4-edges surfaces

The fitting process is design for 4-edges surfaces. However, since it is possible to define this edges in the .axl input data file by yourself, one could try to fit a triangular surface with

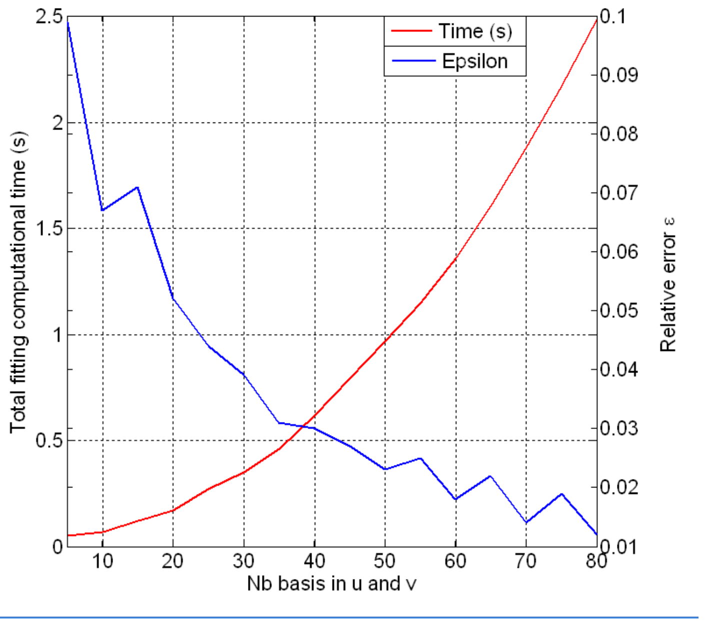


Figure 4: Global fitting method : error and computational time as functions of the number of B-spline basis (example on a 41 × 41 vertices surface).

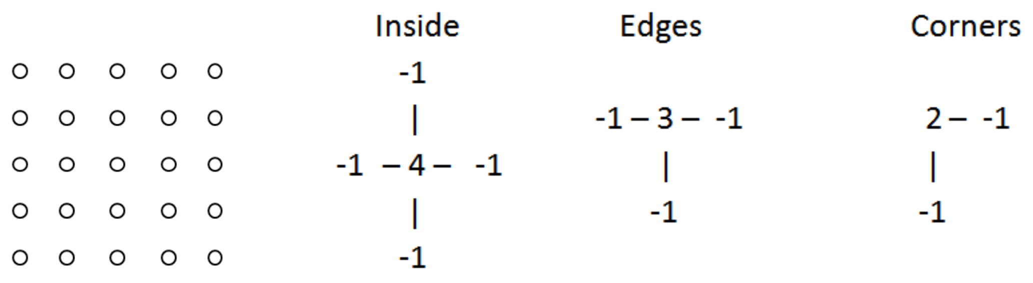


Figure 5: Left : 5 × 5 control points network. Right : laplacian matrix constraints applied to this network (v4 neighbourhood).

Parameter name	Meaning	Type	Range values	Default value	Channel #
nbf_u	starting number of basis functions in u direction	int	$[4; 512]$	dynamic	(0)
nbf_v	starting number of basis functions in v direction	int	$[4; 512]$	dynamic	(1)
nb_it	maximum number of iterations	int	$[1; 5]$	1	2
nb_it_proj_mthd	number of iterations for projective method parameter computation	int	$[1; 5]$	2	3
paramethod	parameter mesh computing method	int	$[0; 3]$	3 (projective)	4
λ	laplacian correction matrix multiplying coefficient	double	$[0; 1]$	10^{-2}	5
epsilon	required error (relative)	double	≥ 0	10^{-9}	6
err_method	error method	int	$[0; 2]$	2 (SDM)	7
opt_method	optimization / fitting method	int	$[0; 2]$	0 (global)	8
colormap	colormap	bool	0/1	1	(9)

Table 1: Fitting process parameters. The last column corresponds to the channel number provided to the setParameter() method in Axel. () in the last column means this parameter is disable for the moment.

```
2 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1 3 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -1 3 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 -1 3 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 -1 2 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1 0 0 0 0 3 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -1 0 0 0 -1 4 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 -1 0 0 0 -1 4 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 -1 0 0 0 -1 4 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 -1 0 0 0 -1 3 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 -1 0 0 0 0 3 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 -1 0 0 0 -1 4 -1 0 0 0 -1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 -1 0 0 0 -1 4 -1 0 0 0 -1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 -1 0 0 0 -1 4 -1 0 0 0 -1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 -1 0 0 0 -1 3 0 0 0 0 -1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 3 -1 0 0 0 -1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 -1 4 -1 0 0 0 -1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 -1 4 -1 0 0 0 -1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 -1 4 -1 0 0 0 -1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 -1 3 0 0 0 0 -1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 2 -1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 -1 3 -1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 -1 3 -1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 -1 3 -1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 -1 2
```

Figure 6: Example of a laplacian correction (sparse) matrix applied on a 5×5 control points net (figure 5).

restricting an edge to a simple point. This won't prevent the fitting process from working, but the quality of the resulting fitting surface might be affected because of the non homogeneous density of the control points network.

For more than 4-edges surfaces, following the same principle one idea could be to merge two edges (or more) into one, but here again the quality of the resulting fitting surface should suffer from this, since B-splines are not convenient to fit sharp corners. Hence the risk of getting some "wave effects" on these edges. Then the best way to do should be to divide the input surface into several quadrangular surfaces, and fit them separately.

8.3 Closed surfaces

The fitting process also works quite sucessfully on closed surfaces such as torus or 2 edges surfaces like Moebius band (figure 7 left and right), since it is possible to artificially define them as 4 edges surfaces in their .axl file or whilst their construction process (meaning this kind of surfaces may be "cut" to get a surface which is homeomorphic to a quadrangular). However, the geometric continuity (C^0) is not assured since the control points coordinates of two edges "stuck together" may be different.

9 Upcoming developments

- Develop a method to automatically compute λ, the correction matrix coefficient.
- List in a table the fitting functions, with their input / output variables, types, significations, and default values.

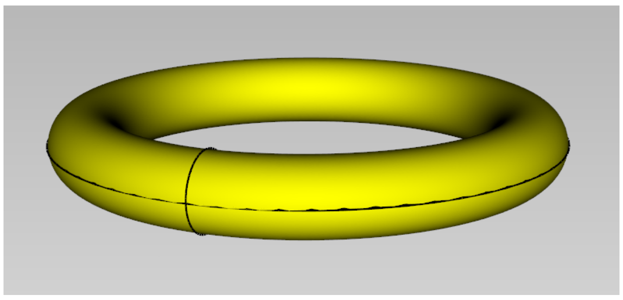
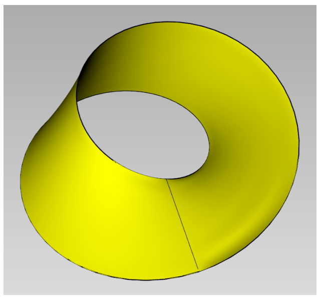



Figure 7: Yellow : resulting B-spline closed surfaces of the fits of the torus (left) and the Moebius band (right) ; black : the four edges of their corresponding data surfaces.

10 Infos, warnings or error messages (visible in the default display output only if you run Axel from a terminal or in a code editor)

They are explicit. However here is an exhaustive list :

- ERROR Fitting process : NO ARGUMENT ; please select a data mesh to fit.
- ERROR Fitting process : TOO MANY ARGUMENTS ; please select ONE data mesh to fit.
- ERROR Fitting process : only works on MESHES !
- ERROR Fitting process : only works on DATA MESHES !
- ERROR Fitting process : fill failed : wrong channel.
- ERROR Fitting process : no goSurfaceBSpline available.
- ERROR Fitting process : no output.
- WARN Fitting process : DUPLICATED INDICES VALUES (. . .) DETECTED IN EDGE # . . . at index # . . . !
- INFO Fitting process : initial surface = Coons patch.
- INFO Fitting process : initial surface = barycentric patch.
- INFO Fitting process : PDM max error = . . .
- INFO Fitting process : TDM max error = . . .
- INFO Fitting process : SDM max error = . . .
- INFO Fitting process : dialog process update.
- INFO Fitting process : doesn't run.
- INFO Fitting process : projective parameterization method cpu time : . . .

- INFO Fitting process : recomputed starting parameters : nbf_u = . . . / nbf_v = . . .
- INFO Fitting process : surface fitting cpu time = . . .
- INFO Fitting process : nbf_u = . . . ; nbf_v = . . . ; paramethod = . . . ; err_mthd = . . . ; opt_mthd = . . . ; lambda = . . . ; nbmax_it = . . .
- INFO Fitting process : updated output error = . . .
- INFO Fitting process : required precision reached :)
- INFO Fitting process : required precision not reached :|

The current step surface fitting CPU time and the parameters set values are also displayed in the terminal.

11 References

11.1 Programming and computationnal concerns

Axel webpage at inria : axel.inria.fr

Galaad webpage at inria : www-sop.inria.fr/galaad/

dtk webpage at inria : dtk.inria.fr

Qt project webpage : qt-project.org

Eigen library webpage : eigen.tuxfamily.org

11.2 Theoritical concerns

11.2.1 On parameterization techniques

[1] M.S. Floater. One-to-one picewise linear mappings over triangulations *Mathematics of Computation,* 72:685-696, 2003.
[2] W.T. Tutte. How to draw a graph. Proceedings of the London mathematical society, s3-13 (1) : 743-767, 1963.

[3] M.S. Floater and K. Hormann. Surface parameterization : a tutorial and survey. In N.A. Dogson, M.S. Floater, and M.A. Sabin editors, Advances in multiresolution for geometric modelling, mathematical and visualization, p157-186, Berlin, Heidelberg, 2005.

[4] M.S. Floater. Mean value coordinates, CAGD 20(1), 19-20 March 2003.

11.2.2 On error methods

[5] P. Bo, R. Ling, W. Wang : A revisit to fitting parametric surfaces to point clouds, Computer & Graphics, 36, p534-540, (2012).

[6] W. Wang, H. Pottmann, Y. Liu : Fitting B-Spline curves to point clouds by curvature-based square distance minimization, ACM transaction on graphics, vol 25, no 2, April 2006, p214-238.

11.2.3 On Coons patch

[7] G. Farin, D. Hansford : Discrete Coons patch, CAGD, 16, p691-700 (1999).

11.2.4 On B-splines surface fitting

http://www.geometrie.tuwien.ac.at/floery/papers/simon_floery_dr.pdf

11.2.5 On B-splines curves fitting

http://www.cs.mtu.edu/ shene/COURSES/cs3621/NOTES/INT-APP/CURVE-APP-global.html

http://www.geometrie.tuwien.ac.at/floery/papers/simon_floery_da.pdf