An iterative method for rational pole curve fitting

J. C. Chambelland
jcchambe@esil.univ-mrs.fr

M. Daniel
mdaniel@esil.univ-mrs.fr

J. M. Brun
jmbrun@esil.univ-mrs.fr

LSIS (UMR CNRS 6168). ESIL-Case 925, 163 Avenue de Luminy
13288 Marseille cedex 09 - (France)

ABSTRACT

This paper adresses the problem of least-square fitting with rational pole curves. The issue is to minimize a sum of squared Euclidean norms with respect to three types of unknowns: the control points, the node values, and the weights. A new iterative algorithm is proposed to solve this problem. The method alternates between three steps to converge towards a solution. One step uses the projection of the data points on the approximant to improve the node values, the two others use a gradient based technique to update the control point positions and the weight values. Experimental results are proposed with rational Bézier and NURBS curves.

Keywords

Least-square fitting, rational pole curves, optimization, iterative methods.

1. INTRODUCTION

Pole curve fitting techniques are often used in CAD softwares to smooth a set of data points. Most of these are least-square based methods and aim at minimizing, with respect to control points and node values, a sum of squared Euclidean norms measuring the distance between the set of data points and the curve to be fitted. Among this kind of method, one can emphasize the linear least square method [LS86, Dan96] with fixed parametrizations [Lee89, PT96, ZCM98, Far01], iterative "Hoschek like" methods [Hos88, SD03, CDB05], and global approaches [SKH98, AB01]. These methods are efficient for polynomial and piecewise polynomial pole curves but not for rational curves because the weights linked to the poles are not considered to improve the accuracy of the fitting. Based on the idea initially proposed in [CDB05] for non-rational pole curve fitting, this paper presents a new iterative method allowing to handle the influence of weights in this problem. The issue being to minimize a sum of squared Euclidean norms with respect to three types of unknowns (the control points, the node values, and the weights), the proposed method alternates between three steps to approach a solution. One step uses the projection of the data points on the approximation curve to improve the node values linked to data points. The two others use a robust gradient based technique to update the control point positions and the weight values.

Conference proceedings ISBN 80-86943-05-4
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.

The problem formulation is given in section 2. Our algorithm is detailed in section 3. Experimental results are proprosed with rational Bézier and NURBS curves in section 4.

2. PROBLEM FORMULATION

We want to fit a k-dimensional ($k > 1$) rational pole curve $\mathbf{C}(t)$ ($t \in [a,b]$) with $(n+1)$ control points to a given set of $(m+1)$ ordered k dimensional data points $\mathbf{D}$ in the least-square sense. This problem is to minimize a sum of squared Euclidean norms with respect to three kinds of unknowns: the set of control points $\mathbf{P}$, the set of node values $\mathbf{t}$ linked to data points, and the set of weight values $\mathbf{w}$ linked to control points. Let be:

- $\mathbf{D} = (\mathbf{d}_0, .., \mathbf{d}_m)$ the set of ordered points in $\mathbb{R}^k$
- $\mathbf{P} = (\mathbf{P}_0, .., \mathbf{P}_n)$ the set of control points in $\mathbb{R}^k$
- $\mathbf{w} = (w_0, .., w_n)$ the set of weights in $\mathbb{R}^{+*}$
- $\mathbf{t} = (t_0, .., t_m)$ the set of parametric nodes in $[a,b]$

The definition of the rational approximant is:

$$\mathbf{C}(t) = \frac{\sum_{i=0}^{n} w_i H_i(t) \mathbf{P}_i}{\sum_{i=0}^{n} w_i H_i(t)} \quad n \leq m, \quad t \in [a,b] \qquad (1)$$

And the problem is to minimize the function:

$$d(\mathbf{P}, \mathbf{t}, \mathbf{w}) = \sum_{j=0}^{m} \|\mathbf{d}_j - \mathbf{C}(t_j)\|_2^2 = \sum_{j=0}^{m} \|\mathbf{E}_j\|_2^2 \qquad (2)$$

fixing $t_0 = a$, $t_m = b$, $\mathbf{C}(t_0) = \mathbf{d}_0$, $\mathbf{C}(t_m) = \mathbf{d}_m$ such that the first and the last data points respectively match with the first and the last points of the curve.

3. PROPOSED ALGORITHM

Outline

Our algorithm allows the minimization of function $d(\mathbf{P},\mathbf{t},\mathbf{w})$ while improving the fitting of an initial approximation curve which provides guess values for $\mathbf{P}$, $\mathbf{t}$, and $\mathbf{w}$. This algorithm uses three steps to locate an optimal solution. As the problem depends on three kinds of unknowns, we opted for a relaxation approach which consists of alternately decreasing objective function $d(\mathbf{P},\mathbf{t},\mathbf{w})$ with respect to control points, node values and weight values. **The first step** minimizes $d(\mathbf{P},\mathbf{t},\mathbf{w})$ with respect to the node values, considering the control point position and the weight values fixed. This corresponds to minimize each term $\|\mathbf{d}_j - \mathbf{C}(t_j)\|^2$ with respect to the node t_j and leads to the solution of the non-linear problem ($\bullet$ is the inner product of two vectors):

$$(\mathbf{d}_j - \mathbf{C}(t_j)) \bullet \frac{d\mathbf{C}}{dt}(t_j) = 0 \qquad \forall j \in [0..m]$$

Geometrically, this problem corresponds to find parameter t_j such that $\mathbf{C}(t_j)$ is the orthogonal projection of data point $\mathbf{d}_j$ (see figure 1) on the approximant.

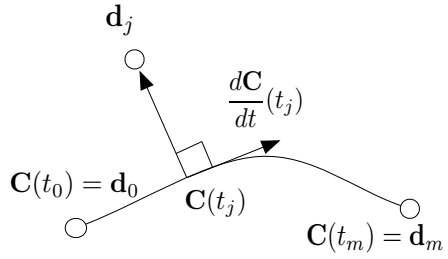

Figure 1: Data point projection

In order to find the closest point and its parametric value, simple "zero order" algorithms can be used. They consist in comparing Euclidean distances between the point to project and a dense set of sample points belonging to the parametric curve. This type of algorithm is inefficient for accurate projections, but useful to provide initial values to "Newton-like" algorithms which use first [Mor97, MH03] or/and second parametric derivatives of the curve [HW05]. According to their rate of convergence, second order algorithms converge theoretically faster than first order algorithms but require more computing time for each iteration and more memory to handle curvature information. In our implementation, we opted for a customized first order projection algorithm. We particularly enforced its robustness, limiting at each iteration the parametric displacement to the value $(b-a)/(m+1)$. This significantly reduces the risk of overshoot and in most cases, very accurate projections are obtained in less than 5 iterations.

The second step consists in reducing $d(\mathbf{P},\mathbf{t},\mathbf{w})$ with respect to the control points, considering the node values and the weight values fixed. Since $d(\mathbf{P},\mathbf{t},\mathbf{w})$ is infinitely differentiable with respect to $\mathbf{P}$, we can compute its gradient vector with respect to control points $\nabla^{\mathbf{P}} = (\nabla^{\mathbf{P}}_0,..,\nabla^{\mathbf{P}}_n)^\top$ and affirm that its negative direction is a direction of maximum rate of decrease ([PFTV02, AW95]). Thus, we can "locally" minimize $d(\mathbf{P},\mathbf{t},\mathbf{w})$, updating control points by the vector $-\alpha^{\mathbf{P}}\nabla^{\mathbf{P}}$ assuming that $\alpha^{\mathbf{P}}$ is a suitably computed positive scalar value. One way to compute the best scalar value $\alpha^{\mathbf{P}}_{max}$, which minimizes $d(\mathbf{P} - \alpha^{\mathbf{P}}\nabla^{\mathbf{P}},\mathbf{t},\mathbf{w})$ with respect to $\alpha^{\mathbf{P}}$, is to use the following equation:

$$\alpha^{\mathbf{P}}_{max} = \frac{\|\nabla^{\mathbf{P}}\|^2}{(H^{\mathbf{P}}\nabla^{\mathbf{P}})^\top \bullet \nabla^{\mathbf{P}}} = \frac{\sum_{i=0}^{n} \|\nabla^{\mathbf{P}}_i\|^2}{(H^{\mathbf{P}}\nabla^{\mathbf{P}})^\top \bullet \nabla^{\mathbf{P}}}$$

where $H^{\mathbf{P}}$ is the Hessian matrix of $d(\mathbf{P},\mathbf{t},\mathbf{w})$ with respect to $\mathbf{P}$. This result is used in the robust "steepest descent method" [PFTV02, AW95], but as a matter of fact, requires an explicit handling of the Hessian matrix of the objective function. This task can be sometimes difficult and computationally intensive, but in our case, the Hessian matrix with respect to control points allows us to simplify the denominator $(H^{\mathbf{P}}\nabla^{\mathbf{P}})^\top \bullet \nabla^{\mathbf{P}}$. Assuming for convenience that:

$$\phi_i(t_j) = \frac{w_i H_i(t_j)}{\sum_{i=0}^{n} w_i H_i(t_j)} \tag{3}$$

We can simplify the expression of α_{max} to the reduced expression:

$$\alpha^{\mathbf{P}}_{max} = \frac{\sum_{i=0}^{n} \|\nabla^{\mathbf{P}}_i\|^2}{2\sum_{j=0}^{m} \|\sum_{i=0}^{n} \phi_i(t_j)\nabla^{\mathbf{P}}_i\|^2} \tag{4}$$

Proof:

The gradient of $d(\mathbf{P},\mathbf{t},\mathbf{w})$ with respect to $\mathbf{P}$ is the $(n+1)$ dimensional column vector:

$$\nabla^{\mathbf{P}} = \begin{pmatrix} \nabla^{\mathbf{P}}_0 \\ \vdots \\ \nabla^{\mathbf{P}}_n \end{pmatrix} = \begin{pmatrix} \frac{\partial d(\mathbf{P},\mathbf{t},\mathbf{w})}{\partial \mathbf{P}_0} \\ \vdots \\ \frac{\partial d(\mathbf{P},\mathbf{t},\mathbf{w})}{\partial \mathbf{P}_n} \end{pmatrix}$$

The Hessian of $d(\mathbf{P},\mathbf{t},\mathbf{w})$ with respect to $\mathbf{P}$ is the $(n+1)*(n+1)$ symmetric matrix:

$$H^P = \begin{pmatrix} \frac{\partial^2 d(\mathbf{P},\mathbf{t},\mathbf{w})}{\partial \mathbf{P}_0^2} & \cdots & \frac{\partial^2 d(\mathbf{P},\mathbf{t},\mathbf{w})}{\partial \mathbf{P}_0 \partial \mathbf{P}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 d(\mathbf{P},\mathbf{t},\mathbf{w})}{\partial \mathbf{P}_n \partial \mathbf{P}_0} & \cdots & \frac{\partial^2 d(\mathbf{P},\mathbf{t},\mathbf{w})}{\partial \mathbf{P}_n^2} \end{pmatrix}$$

From equ. (2) and (3), we can easily state that:

$$\nabla^{\mathbf{P}} = \begin{pmatrix} -2\sum_{j=0}^{m} \phi_0(t_j)\mathbf{E}_j \\ \vdots \\ -2\sum_{j=0}^{m} \phi_n(t_j)\mathbf{E}_j \end{pmatrix}$$

and:

$$H^P = \begin{pmatrix} 2\sum_{j=0}^{m} \phi_0^2(t_j) & \dots & 2\sum_{j=0}^{m} \phi_0(t_j)\phi_n(t_j) \\ \vdots & \ddots & \vdots \\ 2\sum_{j=0}^{m} \phi_n(t_j)\phi_0(t_j) & \dots & 2\sum_{j=0}^{m} \phi_n^2(t_j) \end{pmatrix}$$

Consequently, the product $H^{\mathbf{P}}\nabla^{\mathbf{P}}$ is the $(n+1)$ dimensional column vector:

$$\begin{pmatrix} 2\sum_{j=0}^{m} \phi_0^2(t_j)\nabla_0^{\mathbf{P}} + & \dots + 2\sum_{j=0}^{m} \phi_0(t_j)\phi_n(t_j)\nabla_n^{\mathbf{P}} \\ \vdots & \ddots \ \vdots \\ 2\sum_{j=0}^{m} \phi_n(t_j)\phi_0(t_j)\nabla_0^{\mathbf{P}} + \dots & + 2\sum_{j=0}^{m} \phi_n^2(t_j)\nabla_n^{\mathbf{P}} \end{pmatrix}$$

which can be reduced to:

$$H^{\mathbf{P}}\nabla^{\mathbf{P}} = \begin{pmatrix} 2\sum_{j=0}^{m} \phi_0(t_j) \sum_{i=0}^{n} \phi_i(t_j)\nabla_i^{\mathbf{P}} \\ \vdots \\ 2\sum_{j=0}^{m} \phi_n(t_j) \sum_{i=0}^{n} \phi_i(t_j)\nabla_i^{\mathbf{P}} \end{pmatrix}$$

thus:

$$\begin{aligned} (H^{\mathbf{P}}\nabla^{\mathbf{P}})^{\top} \bullet \nabla^{\mathbf{P}} &= 2\sum_{k=0}^{n}\sum_{j=0}^{m} \phi_k(t_j) \sum_{i=0}^{n} \phi_i(t_j)\nabla_i^{\mathbf{P}} \bullet \nabla_k^{\mathbf{P}} \\ &= 2\sum_{j=0}^{m}\sum_{k=0}^{n} \phi_k(t_j)\nabla_k^{\mathbf{P}} \bullet \sum_{i=0}^{n} \phi_i(t_j)\nabla_i^{\mathbf{P}} \\ &= 2\sum_{j=0}^{m} \| \sum_{i=0}^{n} \phi_i(t_j)\nabla_i^{\mathbf{P}} \|^2 \end{aligned}$$

This allows us to state that:

$$\alpha_{max}^{\mathbf{P}} = \frac{\sum_{i=0}^{n} \|\nabla_i^{\mathbf{P}}\|^2}{2\sum_{j=0}^{m} \| \sum_{i=0}^{n} \phi_i(t_j)\nabla_i^{\mathbf{P}} \|^2}$$

q. e. d.

The third step consists in decreasing $d(\mathbf{P},\mathbf{t},\mathbf{w})$ with respect to the weight values, considering the control point position and the node values fixed. This step uses the direction of maximum rate of decrease ([PFTV02, AW95]) of the objective function with respect to weights, i.e the negative direction of the vector $\nabla^{\mathbf{w}}$ which is the gradient of $d(\mathbf{P},\mathbf{t},\mathbf{w})$ with respect to $\mathbf{w}$. The gradient of $d(\mathbf{P},\mathbf{t},\mathbf{w})$ with respect to $\mathbf{w}$ is the $(n+1)$ dimensional column vector:

$$\nabla^{\mathbf{w}} = \begin{pmatrix} \nabla_0^{\mathbf{w}} \\ \vdots \\ \nabla_n^{\mathbf{w}} \end{pmatrix} = \begin{pmatrix} \frac{\partial d(\mathbf{P},\mathbf{t},\mathbf{w})}{\partial \mathbf{w}_0} \\ \vdots \\ \frac{\partial d(\mathbf{P},\mathbf{t},\mathbf{w})}{\partial \mathbf{w}_n} \end{pmatrix}$$

Using equ. (2) and the sum and product derivative rules, we can state that:

$$\nabla^{\mathbf{w}} = \begin{pmatrix} 2\sum_{j=0}^{m} \mathbf{E}_j \bullet \frac{\partial \mathbf{E}_j}{\partial \mathbf{w}_0} \\ \vdots \\ 2\sum_{j=0}^{m} \mathbf{E}_j \bullet \frac{\partial \mathbf{E}_j}{\partial \mathbf{w}_n} \end{pmatrix} \tag{5}$$

Using the difference and the quotient derivative rules, we can state that:

$$\begin{aligned} \frac{\partial \mathbf{E}_j}{\partial \mathbf{w}_i} &= -\frac{H_i(t_j)\mathbf{P}_i \sum_{i=0}^{n} w_i H_i(t_j) - H_i(t_j)\sum_{i=0}^{n} w_i H_i(t_j)\mathbf{P}_i}{(\sum_{i=0}^{n} w_i H_i(t_j))^2} \\ &= -\frac{H_i(t_j)(\mathbf{P}_i \sum_{i=0}^{n} w_i H_i(t_j) - \sum_{i=0}^{n} w_i H_i(t_j)\mathbf{P}_i)}{(\sum_{i=0}^{n} w_i H_i(t_j))^2} \\ &= -\frac{H_i(t_j)(\mathbf{P}_i - \frac{\sum_{i=0}^{n} w_i H_i(t_j)\mathbf{P}_i}{\sum_{i=0}^{n} w_i H_i(t_j)})}{\sum_{i=0}^{n} w_i H_i(t_j)} \\ &= -\frac{H_i(t_j)(\mathbf{P}_i - \mathbf{C}(t_j))}{\sum_{i=0}^{n} w_i H_i(t_j)} \end{aligned} \tag{6}$$

Inserting eq. (6) in eq. (5) finally leads to:

$$\nabla^{\mathbf{w}} = \begin{pmatrix} -2\sum_{j=0}^{m} \dfrac{H_i(t_j)\mathbf{E}_j \bullet (\mathbf{P}_i - \mathbf{C}(t_j))}{\sum_{i=0}^{n} w_i H_i(t_j)} \\ \vdots \\ -2\sum_{j=0}^{m} \dfrac{H_i(t_j)\mathbf{E}_j \bullet (\mathbf{P}_i - \mathbf{C}(t_j))}{\sum_{i=0}^{n} w_i H_i(t_j)} \end{pmatrix} \quad (7)$$

While the computation of the suitable positive scalar value $\alpha^{\mathbf{P}}_{max}$ is efficiently handled for step 2, this task is more tedious with weight values. Indeed, the Hessian matrix with respect to $\mathbf{w}$, $H^{\mathbf{w}}$, does not allow any significant simplication for the computation of the "optimal" value which is given with respect to $\mathbf{w}$ by:

$$\alpha^{\mathbf{w}}_{max} = \frac{\|\nabla^{\mathbf{w}}\|^2}{(H^{\mathbf{w}}\nabla^{\mathbf{w}})^{\top} \bullet \nabla^{\mathbf{w}}} = \frac{\sum_{i=0}^{n} \|\nabla^{\mathbf{w}}_i\|^2}{(H^{\mathbf{w}}\nabla^{\mathbf{w}})^{\top} \bullet \nabla^{\mathbf{w}}}$$

Experimental results have shown that the use of this expression, which requires to compute and store the complex expression of the Hessian matrix $H^{\mathbf{w}}$, leads to inefficient results. On the other hand, a constant value multipling the gradient vector often entails the divergence of the objective function near the minimum [PFTV02, AW95]. The solution we opted for is to look for a "satisfactory" scalar value $\alpha^{\mathbf{w}}$ rather than the best one i.e $\alpha^{\mathbf{w}}_{max}$. A backtracking approach is applied to achieve this task. The search is initialized with a tiny positive scalar value for $\alpha^{\mathbf{w}}$ (for example 0.1), which is gradually increased while the decrease of the objective function is verified. Note that, we also take care to keep the positivity of the weight values in order to respect the definition of the rational approximant (see eq. 1) verifying $w_i - \alpha^{\mathbf{w}}\nabla^{\mathbf{w}}_i > 0 \ , \forall i \in [0..n]$. The three steps we described allow us to decrease the error function with respect to three kinds of unknowns. The first is an optimization step, while the two others are only reduction steps corresponding respectively to one step of the steepest descent method and one step of the gradient descent method with an estimated "satisfactory" scalar weighting coefficient for the gradient. Even if these steps could be mixed in different ways to minimize the objective function, we opted for a particular blend based on our most convincing experimental results. This aims at reducing a sum of minimum squared Euclidean norms (as proposed by J. Hoschek [Hos88]). Each odd iteration of our iterative algorithm consists in reducing a minimal sum of squared Euclidean norms with respect to control points (**step 1** + **step 2**) while even iterations aim at reducing a minimal sum of squared Euclidean norms with respect to weight values (**step 1** + **step 3**).

Convergence

The convergence is studied from the sequence given by the positive values of the objective function through iterations. Odd iterations achieve the projection of data points (**step 1**) and decrease the sum of squared Euclidean norms with respect to poles (**step 2**). Even iterations also project the data points (**step 1**) and decrease the sum of squared terms with respect to weight values (**step 3**). From guess values $\mathbf{P}$, $\mathbf{t}$, $\mathbf{w}$, we can easily state that for an odd iteration i, **step 1** leads to a new set $\mathbf{t}'$ such that $d(\mathbf{P},\mathbf{t},\mathbf{w}) \geq d(\mathbf{P},\mathbf{t}',\mathbf{w})$ and that **step 2** leads to a new set $\mathbf{P}'$ such that $d(\mathbf{P},\mathbf{t}',\mathbf{w}) > d(\mathbf{P}',\mathbf{t}',\mathbf{w})$.

If we consider the following even iteration $i+1$, this leads from **step 1** to a new set $\mathbf{t}''$ such that $d(\mathbf{P}',\mathbf{t}',\mathbf{w}) \geq d(\mathbf{P}',\mathbf{t}'',\mathbf{w})$ and **step 3** leads to a new set $\mathbf{w}'$ such that $d(\mathbf{P}',\mathbf{t}'',\mathbf{w}) > d(\mathbf{P}',\mathbf{t}'',\mathbf{w}')$. As illustated in figure 2, this proves that alternating between odd and even iterations ensures that the positive values of the objective function describe a decreasing lower bounded sequence which proves its convergence.

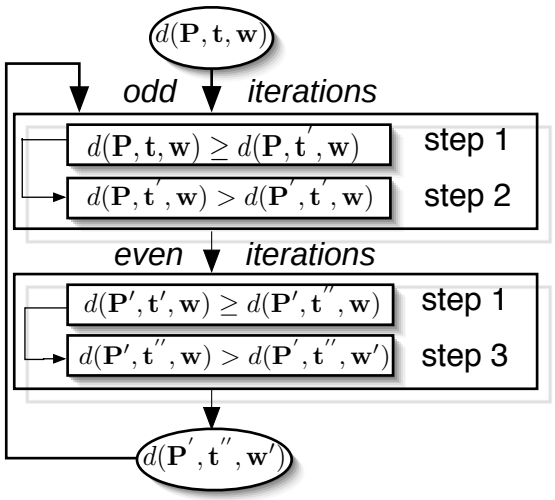

Figure 2: Illustration of the convergence

The convergence is an important robustness criterion but the theoritical efficiency of an iterative optimization method is often given by its rate of convergence. For classical objective functions depending of one type of unknowns, this information corresponds to the speed of convergence of the approximate solution towards the optimal solution. This rate can also be computed from a "residual" function converging towards zero at the solution (for example the squared Euclidean norm of the gradient). Unfortunately, such a study seems unsuitable for our method since three types of unknowns are alternatively modified. The efficiency of this algorithm can however be measured from practical consideration especially from the ratio between the computing time and the approximation accuracy. This analysis is proposed in the last section.

Stopping criteria

As for all iterative methods, a basic stopping criterion is the number of iterations achieved by the process in the case of a wrong expected accuracy or an extraordinary numerical problem. A more meaningful stopping criterion is an expected approximation accuracy for the fitting of a given curve to a given set of data points. When the degrees of freedom of the approximation curve is sufficient, this criterion is particularly suitable for our method, which ensures the strict decrease of the error function through iterations. Another stopping criterion used by classical optimization methods handling one type of unknowns, is the gradient norm of the objective function which converges towards 0 approaching a critical point. In our case, this provides two other stopping criteria, one linked to the norm of $\nabla^{\mathbf{P}}$, the other linked to the norm of $\nabla^{\mathbf{w}}$. Note that an optimum solution is reached when the two norms vanish.

Pseudo-code

*/**Control values**/*
N : positive integer.
ε_d, $\varepsilon_{\nabla^{\mathbf{P}}}$, $\varepsilon_{\nabla^{\mathbf{w}}}$: positive real values.

*/** Initialization**/*
iter=0
initialize $\mathbf{P}$ and $\mathbf{w}$
compute nodes $\mathbf{t} = (t_0 ... t_m)$ such that
$\|\mathbf{d}_j - \mathbf{C}(t_j)\| = \min_{t\in[a,b]} \|\mathbf{d}_j - \mathbf{C}(t)\|$
compute $d(\mathbf{P},\mathbf{t},\mathbf{w})$
compute $\nabla^{\mathbf{P}}$
compute $\nabla^{\mathbf{w}}$
*/**Main loop**/*
while (iter $< N$) **and** ($d(\mathbf{P},\mathbf{t}) \geq \varepsilon_d$) **and**
(($\|\nabla^{\mathbf{P}}\| \geq \varepsilon_{\nabla^{\mathbf{P}}}$) **or** ($\|\nabla^{\mathbf{w}}\| \geq \varepsilon_{\nabla^{\mathbf{w}}}$))
{
  iter=iter+1
  if (iter % 2 == 1) {
    compute $\alpha^{\mathbf{P}}_{max}$
    $\mathbf{P} = \mathbf{P} - \alpha^{\mathbf{P}}_{max}\nabla^{\mathbf{P}}$
  }
  else {
    compute a satisfactory positive value $\alpha^{\mathbf{w}}$
    $\mathbf{w} = \mathbf{w} - \alpha^{\mathbf{w}}\nabla^{\mathbf{w}}$
  }
  compute nodes $\mathbf{t} = (t_0 ... t_m)$ such that
  $\|\mathbf{d}_j - \mathbf{C}(t_j)\| = \min_{t\in[a,b]} \|\mathbf{d}_j - \mathbf{C}(t)\|$
  compute $d(\mathbf{P},\mathbf{t},\mathbf{w})$
  if (iter % 2 == 1)
    compute $\nabla^{\mathbf{w}}$
  else
    compute $\nabla^{\mathbf{P}}$
}
endwhile

4. EXPERIMENTAL RESULTS

In the following, approximation errors are measured from: $E_m = \sqrt{\frac{d(\mathbf{P},\mathbf{t},\mathbf{w})}{m+1}}$ and $E_s = \sup_{j=0,..,m} \|\mathbf{E_j}\|_2$.

Example 1

We fit a planar rational Bézier curve and a planar quadratic clamped NURBS curve to a set of 10 planar data points given in figure (3). The curves are defined on the range [a=0,b=1] and are controled by 5 poles. For the NURBS curve, the knot vector is fixed to $(0.0, 0.0, 0.0, 0.33, 0.66, 1.0, 1.0, 1.0)$. The weight values are initialized to 1 so that the two curves are respectively a Bézier curve and a B-Spline curve before starting the process. The initialization of control points $\mathbf{P}$ has been obtained by a linear least-square minimization with respect to control points using four types of parametrization: centripetal, chordal, Foley-Nielson and Zhang [Lee89, Far01, ZCM98]. Initial approximation errors are gathered in tables 1 and 2. Table 3 shows the initial Euclidean norms of both gradients ($\|\nabla^{\mathbf{P}}\|$ and $\|\nabla^{\mathbf{w}}\|$).

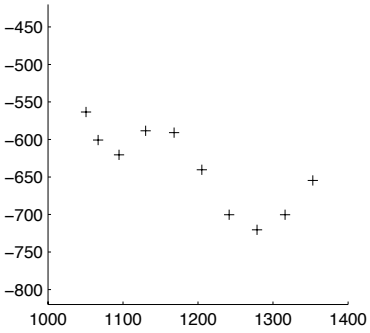

Figure 3: The 10 data points

Init.	$d(\mathbf{P},\mathbf{t},\mathbf{w})$	E_m	E_S
Centripetal	1289.25	11.35	17.62
Chordal	1469.27	12.12	18.65
Foley-N.	1429.62	11.95	19.68
Zhang	729.38	8.54	15.99

Table 1: Initial errors (rational Bézier curve)

Init.	$d(\mathbf{P},\mathbf{t},\mathbf{w})$	E_m	E_S
Centripetal	1182.93	10.87	19.00
Chordal	1642.61	12.81	20.54
Foley-N.	1382.21	11.75	20.21
Zhang	530.18	7.28	12.75

Table 2: Initial errors (NURBS curve)

	rational Bézier		NURBS curve	
Init.	$\|\nabla^{\mathbf{P}}\|$	$\|\nabla^{\mathbf{w}}\|$	$\|\nabla^{\mathbf{P}}\|$	$\|\nabla^{\mathbf{w}}\|$
Centripetal	18.81	1416.43	35.25	384.99
Chordal	21.62	1487.47	35.93	357.44
Foley-N.	21.77	1686.07	42.18	501.27
Zhang	14.84	1534.63	28.55	622.46

Table 3: Initial gradient Euclidean norms

For both types of curve, the best results are obtained with the Zhang parametrization. The corresponding approximations are given in figures 4 and 5.

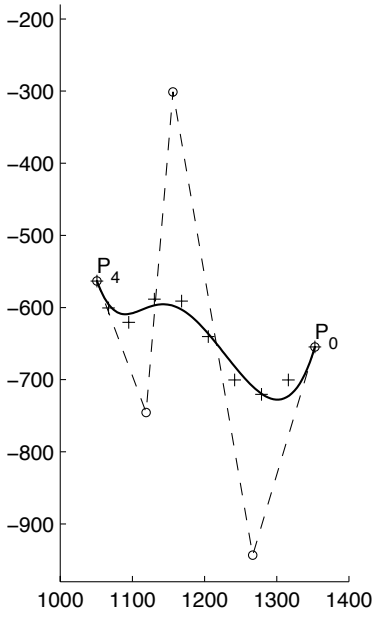

Figure 4: Zhang first approximant (Rational Bézier)

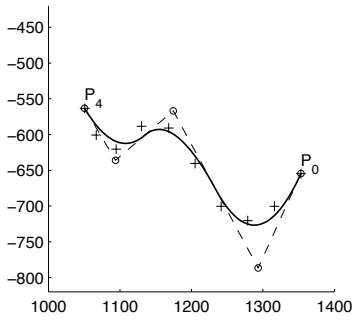

Figure 5: Zhang first approximant (NURBS)

For the rational Bézier curve, we aimed at dividing by at least 1000 the initial norm of both gradients $\|\nabla^{\mathbf{P}}\|$ and $\|\nabla^{\mathbf{w}}\|$ before stopping the process. So according to table 3, $\varepsilon_{\nabla^{\mathbf{P}}}$ and $\varepsilon_{\nabla^{\mathbf{w}}}$ have been respectively set to 0.1 and 1. For the NURBS curve, we expected a very accurate fitting trying to divide by 10000 the norm of both gradients. In this case this corresponds to fix $\varepsilon_{\nabla^{\mathbf{P}}}$ and $\varepsilon_{\nabla^{\mathbf{w}}}$ to 0.01 and 0.1 respectively. The maximum number of iterations has been set to 10000 and the approximation accuracy ε_d has been set to 10^{-6}. Tables 4 and 5 show the approximation errors and the corresponding iteration number and computing time when the process stops (all tests have been achieved on a 2 Gh PC computer). Tables 6 and 7 give the resulting weights. Figures 6 and 7 show the resulting rational Bezier curve and the resulting NURBS curve first initialized from the Zhang parametrization.

Init.	E_m	E_S	iteration	time (s)
Centripetal	0.286	0.491	9175	< 2
Chordal	0.284	0.487	9123	< 2
Foley-N.	0.301	0.519	8277	< 2
Zhang	0.321	0.558	9145	< 2

Table 4: Resulting errors (rational Bézier curve)

Init.	E_m	E_S	iteration	time (s)
Centripetal	0.006	0.010	2903	< 1
Chordal	0.006	0.010	3187	< 1
Foley-N.	0.006	0.010	2845	< 1
Zhang	0.006	0.010	2751	< 1

Table 5: Resulting errors (NURBS curve)

Init.	w_0	w_1	w_2	w_3	w_4
Centripetal	.53	1. 28	1.00	1.38	.38
Chordal	.54	1. 28	.99	1.39	.38
Foley-N.	.52	1. 31	.98	1.36	.45
Zhang	.55	1. 36	.87	1.31	.61

Table 6: Resulting weights (rational Bézier curve)

Init.	w_0	w_1	w_2	w_3	w_4
Centripetal	.76	1.14	1.15	1.04	.84
Chordal	.73	1.14	1.21	1.14	.67
Foley-N.	.77	1.14	1.14	1.02	.88
Zhang	.78	1.15	1.11	0.98	.93

Table 7: Resulting weights (NURBS curve)

One can notice that the initial approximation errors given by the linear least square method are drastically improved. For the rational Bézier curve, E_m is roughly divided by 30 while E_s is roughly divided by 40. For the NURBS curves E_m and E_s are roughly divided by 2000. To measure the influence of handling weights on the resulting approximation errors, we fitted these curves using the iterative method we proposed in [CDB05], which does not handle weight values. Note that according to the weights set to 1, these curves are respectively Bézier and B-Spline curves. For the Bézier curve, the best results we can obtain are $E_m = 1.40$ and $E_s = 2.74$. For the B-Spline curve $E_m = 0.38$ and $E_s = 0.84$. This states that for the rational Bézier curve, handling weights allows to reduce

by roughly 5 the errors obtained without optimizing the objective function with respect to weights. For the NURBS curve, errors are roughly divided by 50.

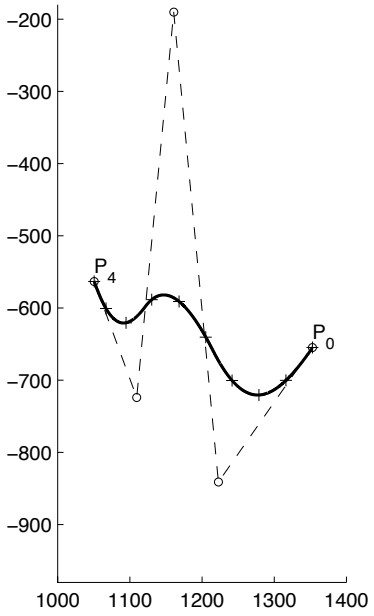

Figure 6: Resulting Rational Bézier

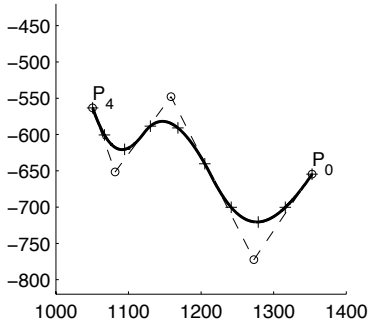

Figure 7: Resulting NURBS

Example 2

In this second example, we fit a clamped cubic NURBS curve to the set of 201 data points given in figure 8. This set is used to emphasize the interest of the improved Hoschek method (IH) proposed in [SD03]. This allows us to make a rapid comparison of our approach with this iterative method providing very convincing results. Note that this method is for B-Spline fitting and does not handle weights. As in [SD03], the curve we fit is defined on the parametric range [a=0,b=1] and is controlled by 19 poles. Intermediate knots are uniformly spaced.

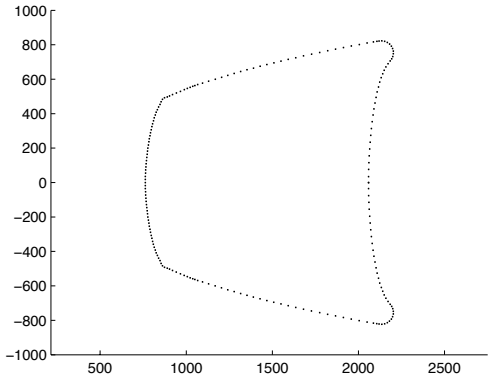

Figure 8: The 201 data points

To start with the same configuration for both methods, weights are initialized to 1 and the first approximation curve (given in figure 9) is obtained by a linear least-square minimization with a centripetal parametrization.

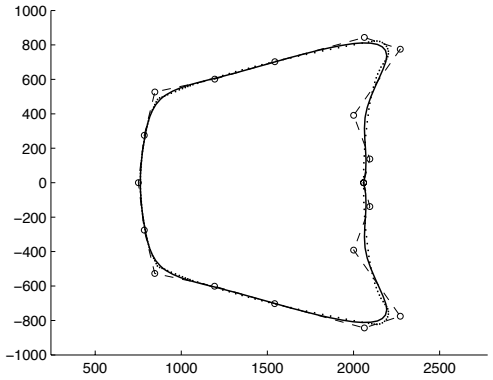

Figure 9: The first approximant

In order to compare both methods we initialized our fitting accuracy with the best mean error which can be obtained with the improved Hoschek method. On a 2Gh PC computer, the best results we can obtain with this method are $E_m = 1.88$ and $E_s = 6.29$ in 4000 iterations (300 s). Setting ε_d to 1.88, our process stops after 1953 iterations (18 seconds). The corresponding maximum error is 5.59. This again emphasizes the importance of handling weights in this problem. Moreover, as illustrated in the graph figure 11, where E_m is collected over 5000 iterations, our method leads to a significant improvement of this value. Indeed, after 5000 iterations approximation errors are $E_m = 1.29$ and $E_s = 4.43$. The corresponding approximation curve is given in figure 10. One can also emphasize the smooth and strict convergence of E_m to-

wards the minimum, and its very fast decrease during the first iterations.

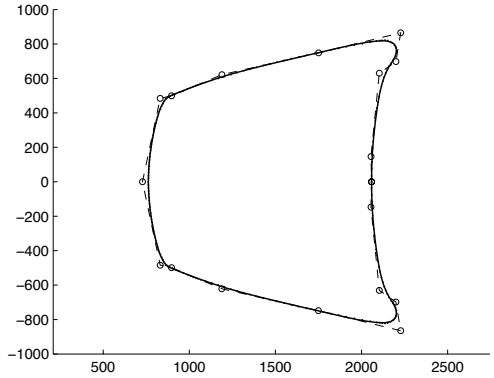

Figure 10: The approximant after 5000 iterations

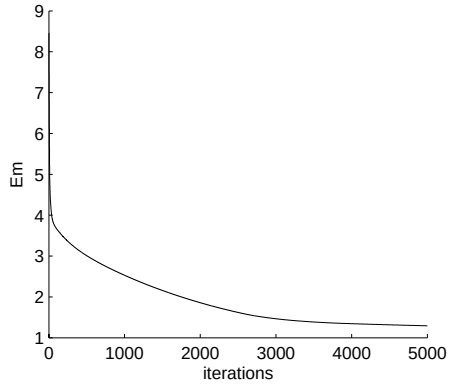

Figure 11: E_m over 5000 iterations

5. CONCLUSION

We have proposed an iterative method to fit a rational pole curve on a set of data points in the least-square sense. Based on a relaxation approach, our algorithm aims at alternatively minimizing a sum of squared Euclidean norms with respect to three types of unknowns: the control points, the node values, and the weights. The method uses a projection step to optimize the objective function with respect to node values and two robust gradient based techniques to optimize the objective function with respect to control points and weight values. The positive error function $d(\mathbf{P},\mathbf{t},\mathbf{w})$ monotonically reduces through iterations ensuring the convergence of the process. This algorithm, suitable for all types of rational pole curve, is robust and efficient. We are currently extending it to surfaces. Experimental results emphasize the drastic influence of handling weights in this problem.

6. REFERENCES

[AB01] M. Alhanaty and M. Bercovier. Curve and surface fitting and design by optimal methods. *Computer Aided Design*, 33(2):167–182, 2001.

[AW95] G. Arfken and H. J. Weber. *Mathematical Methods for Physicists, 4th ed.* Orlando Academic Press, 1995.

[CDB05] J. C. Chambelland, M. Daniel, and J. M. Brun. A robust iterative method devoted to pole curve fitting. In *CAD/GRAPHICS 2005 conference (Hong-Kong, Chine, December 7-10, 2005) proceedings, ISBN 0-7695-2473-7*, pages 22–27. IEEE Computer Society, 2005.

[Dan96] M. Daniel. *Data Fitting with B-splines Curves*. In Modelling and Graphics in Science and Technology, J. Teixeira et J. Rix Eds., Springer Verlag, pp 91-104, 1996.

[Far01] G. Farin. *Curves and Surfaces for CAGD, a Pratical Guide, 5th ed.* Morgan Kaufmann, 2001.

[Hos88] J. Hoschek. Intrinsic parametrization for approximation. *Comput. Aided Geom. Design*, 5(1):27–31, 1988.

[HW05] S. M. Hu and J. Wallner. A second order algorithm for orthogonal projection onto curves and surfaces. *Comput. Aided Geom. Design*, 22(3):251–260, 2005.

[Lee89] E. T. Y Lee. Choosing nodes in parametric curve interpolation. *Computer Aided Design*, 21(6):363–370, 1989.

[LS86] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting: An Introduction*. Academic Press, 1986.

[MH03] Y. L. Ma and W. T. Hewitt. Point inversion and projection for NURBS curve and surface: Control polygon approach. *Comput. Aided Geom. Design*, 20(2):79–99, 2003.

[Mor97] M. E. Mortenson. *Geometric Modeling, 2nd edition*. John Wiley and Sons, 1997.

[PFTV02] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C++, 2nd ed.* Cambridge University Press, 2002.

[PT96] L. A. Piegl and W. Tiller. *The NURBS Book, 2nd ed.* Springer, 1996.

[SD03] E. Saux and M. Daniel. An improved Hoschek intrinsic parameterization. *Comput. Aided Geom. Design*, 20(8-9):513–521, 2003.

[SKH98] T. Speer, M. Kuppe, and J. Hoschek. Global reparametrization for curve approximation. *Comput. Aided Geom. Design*, 15(9):869–877, 1998.

[ZCM98] G. Zhang, F. Cheng, and K. T. Miura. A method for determining knots in parametric curve interpolation. *Comput. Aided Geom. Design*, 15(4):399–416, 1998.