PREPRINT 438

NOVEMBER 2015

igpm | RWTH AACHEN UNIVERSITY

Fast Approximation Methods for Fitting Surfaces to Unorganized Point Clouds

Karl-Heinz Brakhage

Institut für Geometrie und Praktische Mathematik
Templergraben 55, 52062 Aachen, Germany

Karl-Heinz Brakhage
Institut für Geometrie und Praktische Mathematik, RWTH Aachen,
Templergraben 55, 52056 Aachen, Germany
brakhage@igpm.rwth-aachen.de

Fast Approximation Methods for Fitting Surfaces to Unorganized Point Clouds

Karl-Heinz Brakhage
Institut für Geometrie und Praktische Mathematik, RWTH Aachen,
Templergraben 55, 52056 Aachen, Germany
brakhage@igpm.rwth-aachen.de

Abstract

We present and analyze a novel fast method for scattered data approximation with curves / surfaces which have a representation as a linear combination of smooth basis functions associated with the control points. Our technique can be applied to standard Bézier and B-spline curves / surfaces as well as for subdivision schemes. The approach can be formulated in such way that for the iteration we have a standard least squares problem in each step. A regularization term that expresses the fairness of the intermediate and / or final result can be added. Adaptivity is easily integrated in our concept. Furthermore our approach is well suited for reparameterization occurring in grid generation.

Keywords: Splines, Multivariate Approximation, Fairing, Numerical Analysis, Numerical Linear Algebra

Introduction

The problem of computing a smooth curve or surface representation of a target shape given by a set of unorganized data points has many applications in engineering and CAD. For a long period surface fitting methods use the distance between a point on the fitting surface and the corresponding foot point on the target surface or vice versa for minimizing the objective function. These methods are called point distance minimization (PDM). The parameters for the projected points have to be adjusted in an outer loop. Later Pottmann et al. ([6, 7]) introduced an approach based on the minimization of a quadratic approximant of the squared distance function. Their aim was to avoid the parameterization problem and to construct algorithms of second order convergence. Methods based on this idea are called SDM, standing for squared distance minimization. Unfortunately in general the second order Taylor approximant does not lead to symmetric positive definite system matrices and for this reason the existence and uniqueness of the minimum cannot be guaranteed. Thus the second order Taylor approximation was modified to ensure positive definiteness. But this modification destroys the second order and thus the claimed quadratic convergence of those methods. Nevertheless such approaches need less iterations. On the other side the main drawback is a large computational overhead (see [3]), e.g. due to principal curvature computations. Furthermore the parameterization is not really totally avoided.

We use a different approach. This sidesteps the curvature computation. Furthermore our method can still be written in the form of a standard least squares problem which is not the case for SDM. Thus we can use the normal equation and iterative solvers for them as well as orthogonal transformations that have a better condition number than the normal equations. To reduce the number of parameter corrections (the outer loop) we minimize a combination of the PDM and the distances of the data points to the linear approximation of the target surface at the projection point. Note that the later one coincides with the squared distance function for points on the surface. We will demonstrate the idea of our method for the approximation with Beziers. It turned out that our method has super linear convergence with only a small computational overhead for surface normals.

The rest of this paper is organized as follows. First we give some basic notations and properties of Bézier and B-spline surfaces. Next we analyze the behavior of the standard PDM iteration and that of our new one. Details on the used approximation strategies and algorithms for splines will be given. Finally a summary and an evaluation regarding computation time and accuracy of the algorithms usually applied for these purposes will be given.

Basics on Splines and Subdivision

Bézier curves of order n are given by $n+1$ control or Bézier points $\mathbf{p}_i \in \mathbf{R}^m,\ i = 0 \dots n$ (here $m = 2,\ 3$), that build the so-called control polygon, for $u \in [0, 1]$ by

$$\mathbf{x}(u) = \sum_{i=0}^{n} B_i^n(u)\, \mathbf{p}_i = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i}\, \mathbf{p}_i\ . \tag{1}$$

The $B_i^n(u)$ are called Bernstein polynomials. Derivatives of Bézier curves of order n are Bézier curves of order $n-1$.

To keep the technical descriptions as simple as possible, we will not touch the additional constraints coming from continuity conditions. Let us only state here that for B-splines of order four or higher the first and second derivative are continuous. With the exception of extraordinary vertices the same is true for Catmull Clark and Loop subdivision.

One main advantage for interpolation and approximation with curves like (1) is the linearity in the control points $\mathbf{p}_i$. B-spline curves have these property, too. They are given by

$$\mathbf{x}(u) = \sum_{i=0}^{n} N_{i,p,U}(u)\, \mathbf{p}_i \tag{2}$$

where $N_{i,p,U}(u)$ is the i-th normalized B-spline function of order p (degree $p-1$) corresponding to the generally non-uniform knot vector $U = (u_0, u_1, \dots, u_{n+p})$. We usually assume that U is clamped and the internal knots are single knots, i.e., $u_0 = \dots = u_{p-1} < u_p < \dots < u_n < u_{n+1} = \dots = u_{n+p}$. For the sake of simplicity we write $n_{i,p}$ instead of $n_{i,p,U}$. Furthermore, without loss of generality in this paper we assume U to be scaled in such a way that $u_0 = 0$ and $u_{n+p} = 1$. Surfaces are represented by tensor products of the form

$$\mathbf{x}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)\, N_{j,q}(v)\, \mathbf{p}_{ij} \text{ or } \mathbf{x}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u)\, B_j^m(v)\, \mathbf{p}_{ij} \tag{3}$$

for B-spline and Bézier surfaces, respectively. Again these representations are linear in the control points. The same applies for stationary subdivision curves and surfaces. The algorithms presented below can be used for all these curve and surface classes. We rewrite all the above representations in the form

$$\mathbf{x}(u,v) = \sum_{j=1}^{N} B_j(u,v)\,\mathbf{p}_j \tag{4}$$

with general basis functions $B_j(u,v)$. In [10] and [11] it was shown that for Loop and Catmull Clark subdivision the corresponding $B_j(u,v)$ can be evaluated at arbitrary points.

Further information on Bézier- and B-splines can be found in standard literature on CAGD (Computer Aided Geometric Design), e.g. [4]. For more details on subdivision surfaces see [5].

Approximation of Curves and Surfaces

To be more precise with our least squares formulation we have to introduce some technical notations. For an optimal approximation of a given surface $\mathbf{y}(s,t),\ (s,t) \in [s_{min}, s_{max}] \times [t_{min}, t_{max}] =: D \subset \mathbf{R}^2$ by a parametric surface we have to determine the control points $\mathbf{p}_j$ associated to (4) in such a way that

$$\max_{(s,t)\in D} \min_{(u,v)\in[0,1]^2} \|\mathbf{y}(s,t) - \mathbf{x}(u,v)\|_2 \tag{5}$$

is minimized. Since in practice these problems are too complex to solve we switch to discrete approximation problems. For Béziers the whole domain D is normally a-priori splitted into a couple of subdomains. In each (sub-)domain a set of approximation points $\mathbf{y}_i = \mathbf{y}(s_i, t_i)$ is chosen. In this paper we assume that an initial simple mesh corresponding to the correct topology of the target surface and allowing the computation of the (u_i, v_i) is already given. We further use the error estimator

$$\delta = \max_i \|\mathbf{y}_i - \mathbf{x}\,(u_i, v_i))\|_2\ . \tag{6}$$

If the error is too large, we (recursively) subdivide the parameter domain. For subdivision surfaces this is a normal subdivision step. For B-splines we use knot insertion and for Béziers we split each subdomain into four equal parts. The corresponding parameter values (u_i, v_i) are recomputed for the new domains. This step is not necessary for B-splines.

The basic approximation principle, explained for surfaces, is as follows. Let $\mathbf{y}_i,\ i \in \{1, \ldots, M\} =: I$ be given data points or samples on a given target surface. We want to compute a good approximating parametric fitting surface with a representation of the form (4). For the B-spline case we further assume that the knot vectors U and V are already determined with the constraints of the previous section. In a first step we have to find (at least approximately) the nearest points $\mathbf{x}_i = \mathbf{x}(u_i, v_i) \approx \mathbf{y}_i$ on the fitting surface. A good estimation of the parameter values (u_i, v_i) might be a difficult task. Furthermore it is well known that the parameterization problem is a fundamental one for the whole approximation process and the final result. Therefore, parameter correction procedures

have to be used to improve the quality of the final approximation. Unfortunately the decoupling of the overall fitting procedure into the two independent steps parameter correction and solving a linear least squares problem with fixed parameters converges very slowly. On the other hand the overhead due to the curvature computation and the set up of a more complex SDM error function leads to computational inefficiency of SDM. A comparison in [3] shows that the time to attain comparable results used by SDM on iterative optimization is about 30% to 50% more than PDM. The goal of our development is the avoidance of the curvature computation in unity with the construction of a cheap error function that accelerates the parameter correction.

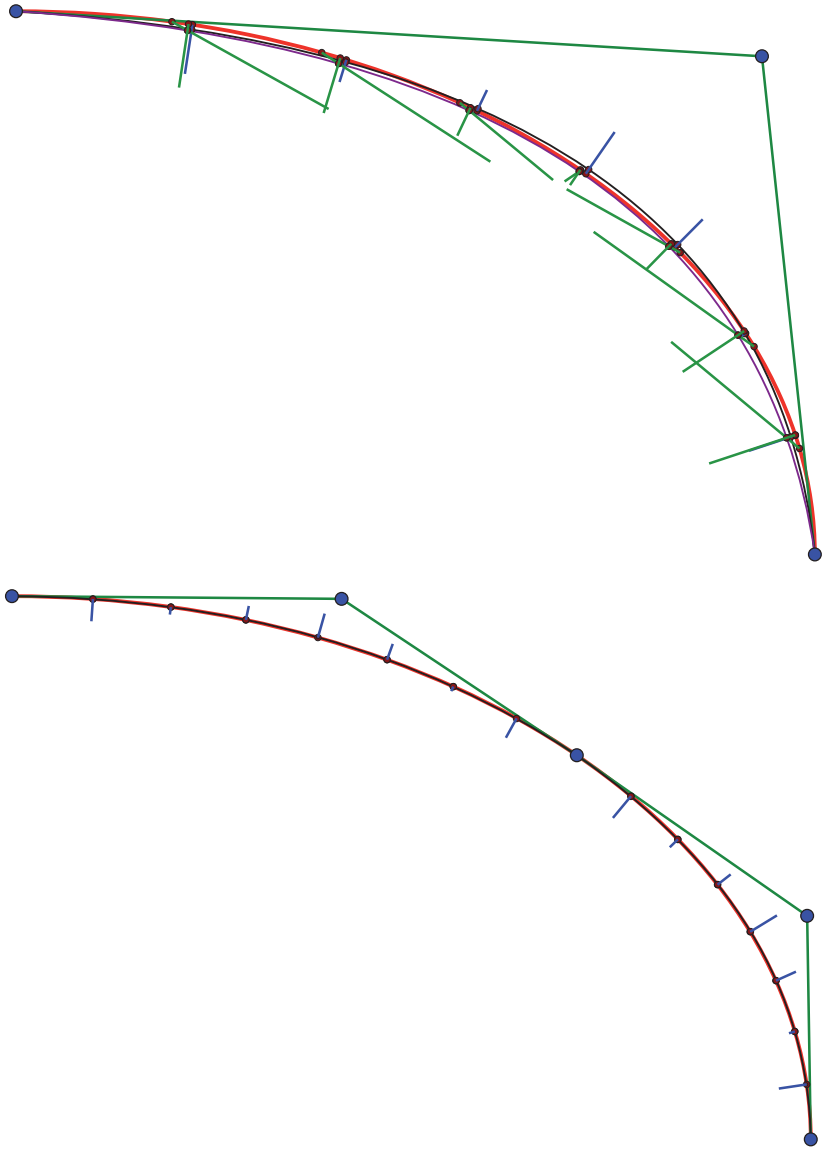

Figure 1: Recursive approximation with quadratic Bézier curves. Upper plot: Initial approximation and parameter correction. Lower plot: First subdivision step.

To get a better understanding of the key ingredients of our methods that lead to convergence acceleration, we first analyze the behavior with planar curves. Figure 1 shows the approximation of one quarter of an ellipse with quadratic Béziers. We force interpolation at the beginning and the end of the interval. Thus $\mathbf{p}_0 = \mathbf{y}(t_{min})$, $\mathbf{p}_2 = \mathbf{y}(t_{max})$ and the only free control point is $\mathbf{p}_1$. Only one level of subdivision is shown (lower plot). This simple example already shows the power of spline approximation and the influence of parameter corrections. For quadratic splines we expect the convergence rate $\mathcal{O}(h^3)$. Of course this can only be achieved if the underlying curve is in $C^3[t_{min}, t_{max}]$. Thus for every recursion the error will be (asymptotically) reduced by a factor of 8.

We chose $\Delta t = (t_{max} - t_{min})/8$ and started with $\mathbf{y}_i = \mathbf{y}(t_{min} + i\,\Delta t)$ for $i \in \{1, 2, \ldots, 7\}$. As initial guess for the u_i we used $u_i = i/8$. In the upper row we see the initial approximation with different (scaled) error vectors. Those without parameter correction are not orthogonal to the target curve. Even more, without parameter correction the error vectors become more and more tangential to the curves in higher recursion levels. For a measurement of the real curve to curve distance at these points the error vectors should be orthogonal to them. This can be achieved with parameter corrections. The error vectors after parameter correction are shown too. Finally the result after some iterations is shown. To make the changes of the parameter values visible we projected onto the target curve. All error vectors are scaled with a factor of ten. In the lower part the result after one subdivision is shown. Here the error vectors are scaled with 80 to make the effect of error reduction visible. Since the error vectors scaled in such a way are shorter for the subdivided curve the errors have reduced by factors larger than 8.

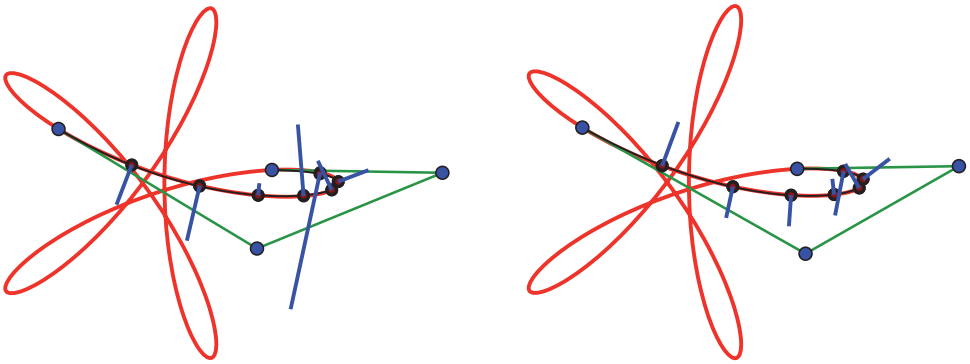

Figure 2: Approximation with cubic Bézier curves – PDM 100 iterations and our method 3 iterations. The error vectors are scaled with a factor of 100.

Next we want to study the influence of good parameter corrections. Again we start with an illustrative example. In Figure 2 we have used a cubic Bézier with interpolation at the boundaries for the approximation of a trochoid. Only one approximation interval is shown with error estimates. Again as reference an approximation with desired relative error of 10^{-6} is shown. Very few control points lead to a good approximation of the trochoid curve. For cubic splines we expect the convergence rate $\mathcal{O}(h^4)$. Here the curve for approximation has to be in $C^4[t_{min}, t_{max}]$. Furthermore we see that we need much less iterations with an optimized parameter correction strategy.

For a better understanding of fruitful concepts for parameter correction we take a look on the related analysis. Pottmann et al. ([6, 7]) introduced a concept of using

local quadratic approximants of the squared distance function to a surface for approximation strategies. At each regular point $\mathbf{y}(s,t)$ of a smooth target surface we have a local right-handed Cartesian system whose first two vectors $\mathbf{t}_k(s,t)$, $k = 1, 2$ are the principal curvature directions belonging to the signed principal curvature $\kappa_k(s,t)$. The third one $\mathbf{n}(s,t)$ is orthogonal to them. Furthermore these vectors are normed. The thereby defined frame Σ is called *principle frame*. If we define $\rho_k(s,t) = 1/\kappa_k(s,t)$ we can write the second order taylor approximation for a point $\mathbf{x}_i$ with coordinates $(0,0,d)$ in Σ (i.e. $\mathbf{y}$ is the closest point to $\mathbf{x}_i$ on the target surface) as

$$F_d(\mathbf{x}) = \frac{d}{d-\rho_1}\left(\mathbf{t}_1^T(\mathbf{x}-\mathbf{y})\right)^2 + \frac{d}{d-\rho_2}\left(\mathbf{t}_2^T(\mathbf{x}-\mathbf{y})\right)^2 + \left(\mathbf{n}^T(\mathbf{x}-\mathbf{y})\right)^2 . \tag{7}$$

Unfortunately there are points $\mathbf{x}_i$ for which F_d is **not** a positive definite quadratic form. This leads to a system matrix that is not symmetric positive definite such that we can not guarantee the existence and uniqueness of the minimum. For this reason (7) is modified to

$$F_d^+(\mathbf{x}) = \frac{|d|}{|d|+|\rho_1|}\left(\mathbf{t}_1^T(\mathbf{x}-\mathbf{y})\right)^2 + \frac{|d|}{|d|+|\rho_2|}\left(\mathbf{t}_2^T(\mathbf{x}-\mathbf{y})\right)^2 + \left(\mathbf{n}^T(\mathbf{x}-\mathbf{y})\right)^2 . \tag{8}$$

Now we end up with a positive definite system matrix but the second order of our approximant is destroyed and thus the claimed quadratic convergence of that method. Nevertheless using (8) for minimization needs much less iterations. For this reason we use a scaled combination of the standard minimization of $(\mathbf{x}-\mathbf{y})^2$ and $(\mathbf{n}^T(\mathbf{x}-\mathbf{y}))^2$. The later one coincides with $F_d^+(\mathbf{x})$ if $d = 0$, which is the case if $\mathbf{x}_i$ lies on the target surface. From this we conclude that it is a good choice to make the scaling dependent of d.

$$F_d^{new}(\mathbf{x}) = (\mathbf{x}-\mathbf{y})^2 + \lambda(d)\left(\mathbf{n}^T(\mathbf{x}-\mathbf{y})\right)^2 . \tag{9}$$

For using (9) in our minimization concept we only need a normal $\mathbf{n}_i$ for each sample point $\mathbf{y}_i$. Since the normal of the fitting surface $\mathbf{x}(u,v)$ approximates the normal of the target surface, we can even use the normal of $\mathbf{x}(u,v)$. The high cost for the curvature computation is avoided, too.

For the sample points $\mathbf{y}_i, i = 1, 2, \ldots M$ with associated parameter values (u_i, v_i) of the fitting surface we can use the following setup. According to (4) the $\mathbf{x}_i$ are given as $\mathbf{x}_i = \mathbf{x}(u_i, v_i)$. We collect the control points $\mathbf{p}_j$ in a vector (of 3d vectors) $\mathbf{p}$, the $\mathbf{y}_i$ in a vector $\mathbf{y}$ and all coefficients $a_{ij} := B_j(u_i, v_i)$ in a matrix $A \in \mathbf{R}^{M\times N}$. With this notation PDM is simply

$$\mathbf{p}^\star = \operatorname*{argmin}_{\mathbf{p}} \|A\,\mathbf{p} - \mathbf{y}\|_2 . \tag{10}$$

For the solution of (10) we can use orthogonal transformations or the normal equation

$$A^T A\,\mathbf{p}^\star = A^T\,\mathbf{y} . \tag{11}$$

Using $\mathbf{x}_i = \sum_j a_{ij}\,\mathbf{p}_j$ we can write the distance to the tangent plane as

$$\|\mathbf{n}_i^T\,(\mathbf{x}_i - \mathbf{y}_i)\|_2 = \left\|\mathbf{n}_i^T\,\left(\sum_j a_{ij}\mathbf{p}_j - \mathbf{y}_i\right)\right\|_2 = \left\|\sum_j a_{ij}\,\mathbf{n}_i^T\,\mathbf{p}_j - \mathbf{n}_i^T\,\mathbf{y}_i\right\|_2 . \tag{12}$$

Notice that (12) leads to a minimization over the control points $\mathbf{p}_j$. To use matrix notations we have to separate the x, y and z components. We define $N_x = \text{diag}\{\mathbf{n}_{i,x}\}$, $N_y = \text{diag}\{\mathbf{n}_{i,y}\}, N_z = \text{diag}\{\mathbf{n}_{i,z}\}$ and collect the terms $\mathbf{n}_i^T\,\mathbf{y}_i$ in a vector $\mathbf{d}$. Now we can write this part as

$$\|N_x\,A\,\mathbf{p}_x + N_y\,A\,\mathbf{p}_y + N_z\,A\,\mathbf{p}_z - \mathbf{d}\|_2 \to \min. \tag{13}$$

For our final minimization problem according to (9) we use (10) and a scaled portion of (13). Here we have to split $\mathbf{p}$ into its x-components $\mathbf{p}_x$, y-components $\mathbf{p}_y$ and z-components $\mathbf{p}_z$. The same way we split $\mathbf{y}$ into $\mathbf{y}_x$, $\mathbf{y}_y$ and $\mathbf{y}_z$. With

$$\hat{A} = \begin{pmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & A \\ \lambda\,N_x\,A & \lambda\,N_y\,A & \lambda\,N_z\,A \end{pmatrix},\ \hat{\mathbf{p}} = \begin{pmatrix} \mathbf{p}_x \\ \mathbf{p}_y \\ \mathbf{p}_z \end{pmatrix} \text{ and } \hat{\mathbf{y}} = \begin{pmatrix} \mathbf{y}_x \\ \mathbf{y}_y \\ \mathbf{y}_z \\ \lambda\,\mathbf{d} \end{pmatrix} \tag{14}$$

our minimization problem now reads

$$\hat{\mathbf{p}}^\star = \underset{\hat{\mathbf{p}}}{\text{argmin}}\,\|\hat{A}\,\hat{\mathbf{p}} - \hat{\mathbf{y}}\|_2. \tag{15}$$

Thus again we can use orthogonal transformations, the normal equations or iterative methods. Especially for recursive approximation of surfaces the iterative methods are much faster. The parameter $\lambda \geq 0$ is chosen depending on the error estimator. For small errors we use a large λ.

As iterative solver we normally use a modification of the Conjugate Gradient method for linear Least Squares. CGNR (see [9]) can directly be applied to (15). But we have developed a more efficient variant for this special case. The only costly part of CG applied to $A\,x = b$ is the matrix-vector product of the system matrix A with an intermediate direction in each step. The normal equation for (15) is

$$\hat{A}^T\,\hat{A}\,\hat{\mathbf{p}}^\star = \hat{A}^T\,\hat{\mathbf{y}}. \tag{16}$$

For the standard CG method applied to (11) we have to compute one product of the form $\mathbf{z}_k = A^T\,A\,\mathbf{d}_k$ in each step. $\mathbf{d}_k \in \mathbf{R}^{3N}$ is a N-vector of 3d-vectors, $A \in \mathbf{R}^{M\times N}$. Thus $A\,\mathbf{d}_k \in \mathbf{R}^{3M}$ and $\mathbf{z}_k \in \mathbf{R}^{3N}$. As in CGNR we avoid the computation of $\hat{A}^T\,\hat{A}$. We rewrite $\hat{A}^T\,\hat{A}$ as

$$\begin{aligned} \hat{A}^T\,\hat{A} &= \begin{pmatrix} A^T\,A + A^T\,\lambda^2\,N_x^2\,A & A^T\,\lambda^2\,N_x\,N_y\,A & A^T\,\lambda^2\,N_x\,N_z\,A \\ A^T\,\lambda^2\,N_x\,N_y\,A & A^T\,A + A^T\,\lambda^2\,N_y^2\,A & A^T\,\lambda^2\,N_y\,N_z\,A \\ A^T\,\lambda^2\,N_x\,N_z\,A & A^T\,\lambda^2\,N_y\,N_z\,A & A^T\,A + A^T\,\lambda^2\,N_z^2\,A \end{pmatrix} \\ &=: A^T \otimes \begin{pmatrix} I + \lambda^2\,N_x^2 & \lambda^2\,N_x\,N_y & \lambda^2\,N_x\,N_z \\ \lambda^2\,N_x\,N_y & I + \lambda^2\,N_y^2 & \lambda^2\,N_y\,N_z \\ \lambda^2\,N_x\,N_z & \lambda^2\,N_y\,N_z & I + \lambda^2\,N_z^2 \end{pmatrix} \otimes A =: A^T \otimes B \otimes A \end{aligned} \tag{17}$$

B is a 3×3 block matrix. Each of the 9 blocks is a diagonal matrix. The multiplication $\otimes$ is implicitly defined in (17) as follows. $A^T \otimes B$ means that each block of B is multiplied with A^T from the left. $B \otimes A$ means that each block of B is multiplied with A from the right. Let $\mathbf{d}_k \in \mathbf{R}^{3N}$. The first product is the same as above: $\hat{\mathbf{z}}_k = A\,\mathbf{d}_k$. Next we apply B: $\overline{\mathbf{z}}_k = B\,\hat{\otimes}\,\hat{\mathbf{z}}_k$. $\hat{\otimes}$ describes an update of the elements in $\hat{\mathbf{z}}_k$. Note that due to

the structure of B only three reals are added to each component in $\hat{\mathbf{z}}_k$. The last step is a *normal* multiplication: $\mathbf{z}_k = A^T\,\overline{\mathbf{z}}_k \in \mathbf{R}^{3N}$. Let us summarize these steps:

$$\hat{\mathbf{z}}_k = A\,\mathbf{d}_k\ ,\ \ \overline{\mathbf{z}}_k = B\,\hat{\otimes}\,\hat{\mathbf{z}}_k\ ,\ \ \mathbf{z}_k = A^T\,\overline{\mathbf{z}}_k \tag{18}$$

For the right hand side we use

$$\hat{A}^T\,\hat{y} = \begin{pmatrix} A^T\,\mathbf{y}_x + A^T\,\lambda^2\,N_x\,\mathbf{d} \\ A^T\,\mathbf{y}_y + A^T\,\lambda^2\,N_y\,\mathbf{d} \\ A^T\,\mathbf{y}_z + A^T\,\lambda^2\,N_z\,\mathbf{d} \end{pmatrix} = A^T \otimes \begin{pmatrix} \mathbf{y}_x + \lambda^2\,N_x\,\mathbf{d} \\ \mathbf{y}_y + \lambda^2\,N_y\,\mathbf{d} \\ \mathbf{y}_z + \lambda^2\,N_z\,\mathbf{d} \end{pmatrix} =: A^T \otimes \hat{\mathbf{b}}\,. \tag{19}$$

Similar to $\overline{\mathbf{z}}_k = B\,\hat{\otimes}\,\hat{\mathbf{z}}_k$ above $\hat{\mathbf{b}}$ is an update of $\mathbf{y}$. Here only one real is added to each component. We rewrite this as

$$\mathbf{b} = A^T \left(\mathbf{y} \,\oplus\, \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix} \otimes\, \lambda^2\,\mathbf{d} \right)\ . \tag{20}$$

With (20) (has to be computed only once) and (18) we have described an efficient form for the costly parts of the CG method. The rest of the computations in CG are only two inner products and three vector additions in each step. Note that we can make use of the sparsity of the matrix A in (18).

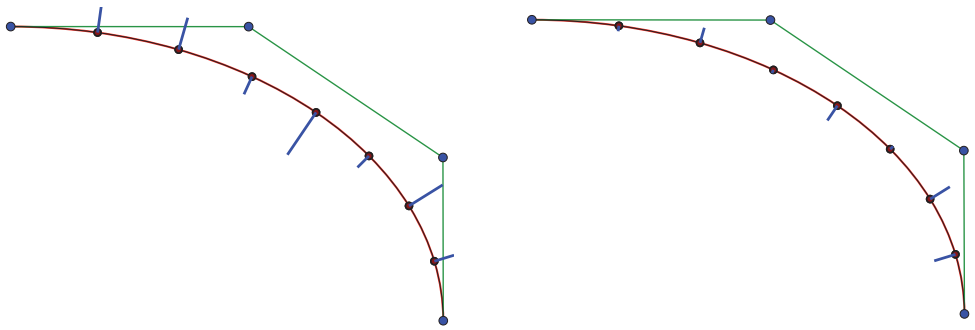

Figure 3: Convergence studies: 30 iterations with PDM (left) versus 3 iterations of our new method (right).

To analyze the coupling effect of the parameter correction with the solution of the linear system of equations we again use a simple curve approximation. Figure 3 shows the approximation of one quarter of an ellipse with cubic Béziers. We have chosen $a = 25$ and $b = 17$ for the principal axis of the ellipse. Again we force interpolation at the beginning and the end of the interval – here including the tangent – and the same initial distribution of the sample points as in Figure 3. The left plot shows the result after 30 iterations with PDM. The error vectors are scaled with a factor of 1000. This is compared with the result of our method after only three iterations. The main reason for the improvement in our method can not be seen from these plots. In Figure 4 we have plotted the signed distance of the parameter values in k-th iteration to those of the *best* approximation. Due to the coupling in our method we reach the optimal parameter values very quickly. We show

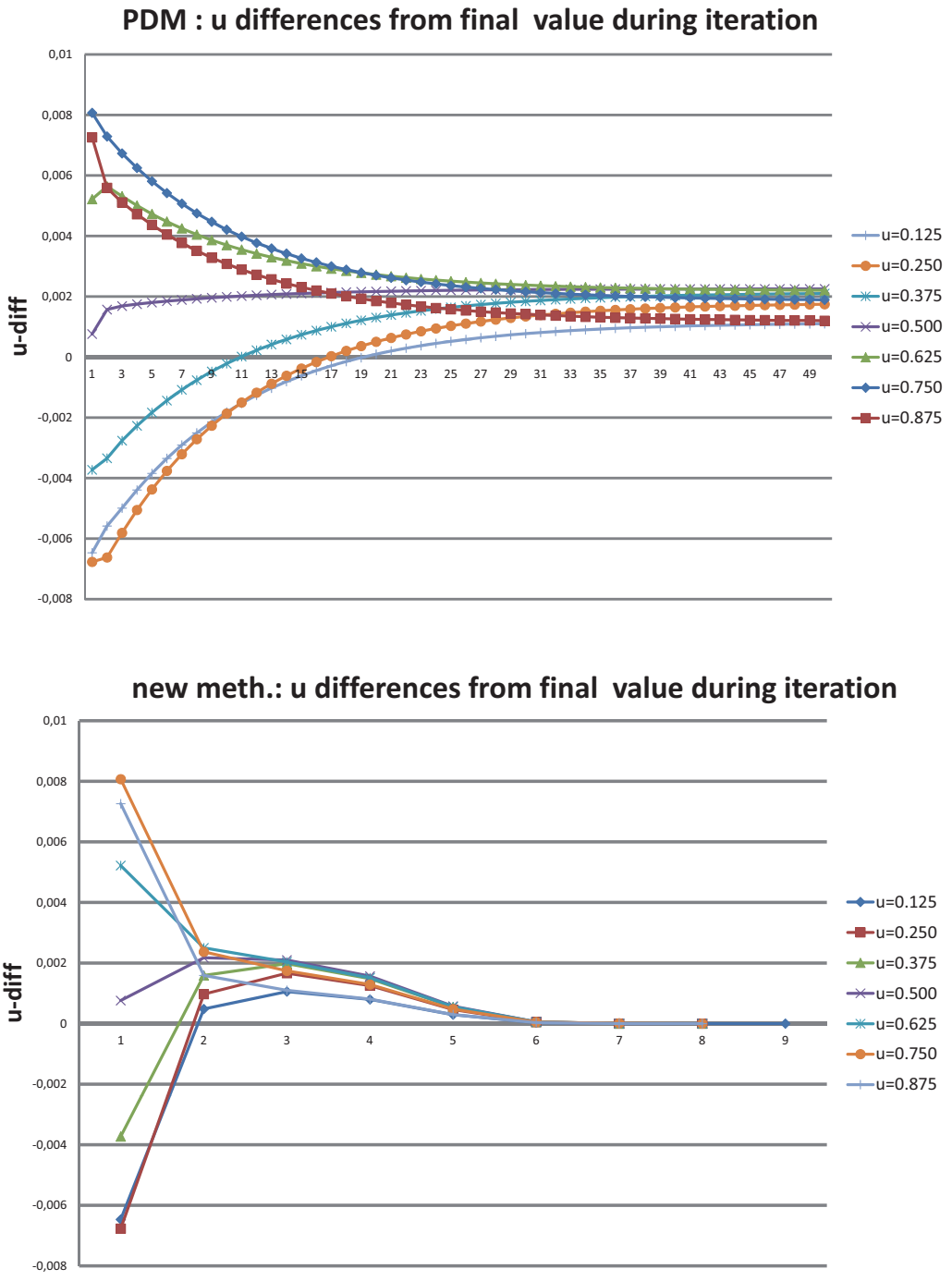

Figure 4: Differences of the parameter values from their optimal place: Upper plot PDM, lower plot our method.

the initial values and 8 iterations for our method and about fifty iterations of PDM. Here PDM needs more than a million iterations to get the optimal parameter values with a precision that leads to a solution that coincides with the optimal solution up to 10 digits.

The table below shows the error estimator according to (6) for the iteration. The errors are shown as multiple of 10^{-3}. Our method shows a quadratic convergence. The number of significant digits is doubled in each step. Due to influence of the machine precision this stops at iteration 5. With iteration 6 we have reached the minimum up to 10 significant digits. PDM has 9 significant digits after one million iterations.

	PDM method	our method
#itera	max error $[10^{-3}]$	max error $[10^{-3}]$
1	51.0291992263	**1**.2156378925
2	45.5340955079	**1.1**050648499
3	40.8463545168	**1.093**2771389
4	36.6556730174	**1.0932474**969
5	32.8881440210	**1.0932475**456
6	29.4980591588	**1.0932475452**
30	2.95707434807	1.0932475454
10^4	**1.**1713527195	1.0932475453
10^5	**1.093247**9122	1.0932475453
10^6	**1.0932475**458	1.0932475453

Conclusion and Future Research

We have presented and analyzed a novel and fast approximation approach for curves and surfaces. It can be used for the construction of smooth surfaces from point clouds as well as for the reparameterization of given surfaces. The methods are not restricted to be applied to Bézier or B-spline surfaces. They can be used for subdivision surfaces as well. In particular, the focus has been on an analytical understanding of the coupling of parameter correction with the linear Least squares approximation step and the transformation of the mathematical model to a form that can be used efficiently for fast solvers. In comparison to the standard PDM approach with decoupling of the linear solver and the parameter correction our method has a tremendous speed up. We need much fewer iterations without the drawback of the computational overhead of SDM.

Encapsulated Postscript (EPS), PDF and Scalable Vector Graphics (SVG) files have the entities quadratic an cubic Bézier curves. EPS and PDF files are for use in printed publications. SVGs can be used for websites. The major advantage of all three formats is the very small size of the files and the possibility to zoom without grid pattern effects. All the curves shown in this paper are piecewise quadratic or cubic Béziers. The reference curves are approximated with our approach by cubic Béziers to a final relative error of 10^{-6}.

There is still a large amount of work left for future research. We want to improve the stop criteria for the overall iteration and implement more flexible subdivision of the parameter domain. Up to know we only use uniform fragmentation for the surface case. For very large problems we want to use preconditioning of the CG method. Furthermore, we will test our algorithms with other surface classes. The integration of this methodology into

our algorithms on grid generation and reparameterization, for instance given in [1, 2, 8], will lead to a significant reduction of computation in those packages, too.

References

[1] K.-H. Brakhage & Ph. Lamby, Application of B-spline Techniques to the Modeling of Airplane Wings and Numerical Grid Generation, *CAGD (Elsevier)*, Volume 25(9), 738-750 (2008)

[2] K.-H. Brakhage, Grid generation and grid conversion by subdivision schemes. in *11^{th} International Conference on Numerical Grid Generation in Computational Field Simulations*, B. Soni et al., editor, Montral, Canada, May 24-28 2009.

[3] K. Cheng, W. Wang, H. Qin, K.-Y. Wong, H. Yang & Y. Liu, Fitting subdivision surfaces to unorganized point data using SDM, in *Pacific Conference on Computer Graphics and Applications, 2004*, 16-24.

[4] G. Farin. *Curves and Surfaces for CAGD. A Practical Guide*, The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, fifth edition, 2002.

[5] J. Peters & U. Reif, *Subdivision Surfaces*, Series: Geometry and Computing, Vol. 3, Springer, 2008.

[6] H. Pottmann & M. Hofer, Geometry of the squared distance function to curves and surfaces, In *VISUALIZATION AND MATHEMATICS III*, 223-244, 2003.

[7] H. Pottmann & S. Leopoldseder, A concept for parametric surface fitting which avoids the parametrization problem, *Computer Aided Geometric Design (20)*, 343-362, 2003.

[8] M. Rom, & K.-H. Brakhage, Reparametrization and Volume Mesh Generation for Computational Fluid Dynamics Using Modified Catmull-Clark Methods, in *Mathematical Methods for Curves and Surfaces, Volume 8177 of Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2014, pages 425-441.

[9] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, second edition, 2003.

[10] J. Stam, Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values, in *Proc. SIGGRAPH 98*, 395-404.

[11] J. Stam, Exact Evaluation of Loop Subdivision Surfaces, in *SIGGRAPH CDROM Proceedings 98*.