A Parallel Multistage Method for Surface–Surface Intersection

Heiko Bürger and Robert Schaback

Abstract: A global divide–and–conquer method and a local marching method are combined into a massively parallel algorithm for surface/surface intersection. Special attention is given to the comparison of bounding boxes and to the safe and efficient calculation of all components of the intersection. The computational complexity of the algorithm is analyzed, and a series of examples is provided for illustration of the theoretical results.

AMS classifications: 65D17, 65Y05, 65Y25, 68U05, 68U07

Keywords: Marching algorithms, Divide–and–conquer methods, bounding boxes, uniqueness.

1 Introduction

Let two smooth and regular surfaces

$$F : \Omega_F \to I\!R^3, \qquad G : \Omega_G \to I\!R^3 \tag{1.1}$$

on polygonal domains Ω_F, Ω_G in $I\!R^2$ be given. The intersection $S := F(\Omega_F) \cap G(\Omega_G)$ may be empty or a set with a rather complicated structure. As outlined already in an early overview article [Pratt/Geisow '86], the calculation of S requires to solve two different subproblems:

- Determine all connected components of S and make sure that no part of S is overlooked;
- Calculate each connected component of S efficiently and accurately.

The first subproblem is of *global* nature, while the second is *local*. Consequently, there are two different kinds of approaches to solve these problems:

- The *global* problem is typically solved by "divide–and–conquer" strategies making use of successive subdivision and bounding boxes ([Houghton et.al. '85], [Dokken et.al. '85]).
- The *local* problem is solved by "marching" algorithms that use predictor–corrector strategies of Newton–Raphson type to follow intersection curves (see e.g.: [Barnhill et.al. '87], [Barnhill/Kersey '90], [Müllenheim '90]).

For both approaches there still is a number of questions to be answered. We pick out a few:

- Which type of bounding box is preferable (and why)?

[0]File: /usr/nam/rschaba/tex/fertig/cut/paper.tex. Date of TEX run : September 15, 1992.
Status : Ready for submission.

- Which types of algorithms are more effective than others?
- Can marching methods be made safer without excess computational cost?
- What is the optimal combination of global divide–and–conquer methods with local marching algorithms?
- How far can surface–surface intersection (SSI) algorithms be parallelized?

This paper, based on [Schaback '89b] and [Bürger '92], gives partial answers to these questions via an analysis of a multi–stage method that combines a general divide–and conquer algorithm with a safeguarded marching algorithm in such a way that

- the tradeoff between safety and speed can be controlled by the user and
- a maximum degree of parallelism is introduced, without introducing too much synchronization or communication overhead.

2 Divide–and–Conquer Algorithm

This section describes a global algorithm of divide–and conquer type that allows to handle SSI problems for surface pairs (1.1). For all surface patches $F : \Omega_F \to \mathbb{R}^3$ considered here, we assume that there is a set $B(F)$, called a "bounding box", with

$$F(\Omega_F) \subseteq B(F),$$

and that pairs of such boxes are easy to test for being disjoint or not. Furthermore we assume each surface to allow subdivision into smaller surfaces and corresponding bounding boxes as

$$\begin{aligned} \Omega_F &= \bigcup_{i=1}^{m} \Omega_{F_i}, & F_i : \Omega_{F_i} &\to \mathbb{R}^3, \\ F_i &= F\big|_{\Omega_{F_i}}, & F_i(\Omega_{F_i}) &\subseteq B(F_i) \end{aligned}$$

for $1 \le i \le m = m(F)$. If subdivision is applied repeatedly, we require it to take only a finite number of steps to let all subsurfaces ("patches") F_i satisfy the criterion

$$\text{diam } B(F_i) < \varepsilon$$

for any prescribed $\varepsilon > 0$. For each type of surface patch the user wants to allow, we thus need subroutines for the following tasks:

- Generation of a bounding box $B(F)$ for F
- Calculation of its diameter
- A sufficient criterion for two bounding boxes $B(F)$ and $B(G)$ to be disjoint
- A subdivision algorithm.

For efficiency reasons we do not require a necessary *and* sufficient criterion for boxes $B(F)$ and $B(G)$ to be disjoint, but for safety reasons we cannot do without a test that safely implies $B(F) \cap B(G) = \emptyset$. Whenever such a test gives no affirmative answer, we write $B(F) \cap B(G) \not\equiv \emptyset$ in the sequel.

The following algorithm uses the data structure of a "bag" or "bucket" without any implied order of storage or access. The only operations are

- test for emptiness of the bag
- taking an element out of a nonempty bag
- putting an element into a bag,

and these operations should be safely executable by parallel processes without any further synchronization or communication.

Algorithm 2.1. (divide–and–conquer)
Let $\mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{R}$ be bags of patch pairs (F, G) with $B(F) \cap B(G) \not\equiv \emptyset$, and let $\varepsilon > 0$ be given.
Start:
$\mathcal{L}_1 := \{(F, G)\}$ for the given surfaces F and G, while $\mathcal{R} := \mathcal{L}_2 := \emptyset$.
Loop:
While $\mathcal{L}_1 \neq \emptyset$
let any number of parallel processes do:

Take a pair (F, G) from $\mathcal{L}_1$.

if diam $B(F) \leq \varepsilon$ and diam $B(G) \leq \varepsilon$
then put (F, G) into $\mathcal{R}$ and repeat loop.

if diam $B(F) > \varepsilon$
then subdivide F into $F_1, \ldots, F_m$.

if diam $B(G) > \varepsilon$
then subdivide G into $G_1, \ldots, G_n$.

For all pairs (F_i, G_j) of generated subpatches do:

if $B(F_i) \cap B(G_j) = \emptyset$
then discard (F_i, G_j)
else

if there is a marching algorithm working on $\mathcal{L}_2$,
and if points $x \in \Omega_F,\ y \in \Omega_G$ with
$\|F(x) - G(y)\| < \varepsilon$ can be found,
then put (F_i, G_j) into $\mathcal{L}_2$
else put (F_i, G_j) into $\mathcal{L}_1$.

In this form, the algorithmus clearly is finite under the above hypotheses on the subdivision strategy and the bounding boxes. It gradually subdivides all the patch pairs in $\mathcal{L}_1$ and passes them on to $\mathcal{R}$ or $\mathcal{L}_2$ if they cannot be discarded.

The link to the local marching algorithm to be described later is an additional routine that tries to find an approximation of an intersection point of two surface patches F_i and G_j. This routine need not be sophisticated or fail–safe, because it just hands the pair (F_i, G_j) and the candidate for an intersection point over to the marching algorithm via the bag $\mathcal{L}_2$. The marching algorithm may pass the pair (F_i, G_j) back to the divide–and conquer–algorithm via $\mathcal{L}_1$ if it cannot properly proceed from the suggested approximate intersection point, and the divide–and–conquer method will have another try after subdivision. Details of this will be given in the context of the multistage algorithm following later.

3 Bounding boxes

Constructing bounding boxes $B(F)$ for a surface $F : \Omega_F \to I\!R^3$ is easy if a representation

$$F(u) = \sum_{i \in I} b_i \beta_i(u), \qquad u \in \Omega_F$$

of $F(\Omega_F)$ via a control net $\{b_i\}_{i \in I} \subset I\!R^3$ and a partition of unity

$$\begin{aligned} \beta_i : \Omega_F &\to I\!R, \quad i \in I \\ \beta_i(u) &\geq 0, \quad i \in I,\ u \in \Omega_F \\ \textstyle\sum_{i \in I} \beta_i(u) &= 1, \quad u \in \Omega_F \end{aligned}$$

is used, because $F(\Omega_F)$ will always be in the convex hull of the control net, irrespective of the explicit form of the partition of unity (see e.g.: [Farin '90], [Hoschek/Lasser '89]).

Coordinate boxes $B(F)$ will then be defined as the smallest L_∞ ball that contains the control net. These boxes are easy to calculate by taking minima/maxima of coordinates of control points. However, when applying subdivision, they do not adjust flexibly to the shape of the surface. If h is the maximum sidelength of a coordinate box, the enclosed surface patch will be approximated up to no more than $\mathcal{O}(h)$.

Oriented parallelepiped boxes are defined as

$$B(F) = \{x \in I\!R^3 \mid \delta_j^- \leq x^T \eta_j \leq \delta_j^+,\ 1 \leq j \leq 3\}$$

for three linearly independent vectors η_j and three pairs of scalars $\delta_j^- < \delta_j^+$, $1 \leq j \leq 3$. The vectors η_j can be normalized to $\|\eta_j\|_2 = 1$ and will then be the surface normals of the parallelepiped. For a variety of surface patch types one can make use of the flexibility of these boxes to yield better approximations of the surface. In general, one chooses η_1 and η_2 to be roughly tangential to the surface patch, while η_3 is approximately normal. Then the real numbers δ_j^- , δ_j^+ can be calculated as

$$\delta_j^- = \min_{i \in I} b_i^T \eta_j, \qquad \delta_j^+ = \max_{i \in I} b_i^T \eta_j,\ 1 \leq j \leq 3$$

to make sure that the control net $\{b_i\}_{i \in I}$ of F and the image $F(\Omega_F)$ of the surface are both contained in $B(F)$. Since subdivision converges quadratically for a wide class of surface patch

types (see [Schaback '89a] and [Schaback '92] for short proofs of this fact using polar forms), we will have

$$\delta_3^+ \ - \delta_3^- \ = \mathcal{O}(h^2)$$

for

$$h := \max(\delta_1^+ \ - \delta_1^- \ , \delta_2^+ \ - \delta_2^- \ )$$

tending to zero during successive subdivision. This allows a comparison of boxing strategies:

Theorem 3.1 *1) If the given smooth and regular surfaces F and G have a distance*

$$dist\ (F,G) = \inf_{\substack{u\in\Omega_F \\ v\in\Omega_G}} \|F(u)-G(v)\| = \delta > 0$$

and if oriented parallelepiped boxes are used, the divide–and–conquer algorithm requires work $\mathcal{O}(\delta^{-2})$ for $\delta \to 0$ in the worst case, while coordinate boxes require $\mathcal{O}(\delta^{-4})$.

2) If F and G have a nonempty intersection S, then the algorithm (with $\mathcal{L}_2$ ignored) calculates a covering of S by bounding boxes of diameter at most ε.

3) Regular pieces of transversal intersection curves are approximated with precision $\mathcal{O}(\varepsilon^2)$ for $\varepsilon \to 0$ if oriented parallelepiped boxes are used, while coordinate boxes provide an approximation of precision $\mathcal{O}(\varepsilon)$.

Proof: The second assertion is obvious. To prove the first, we assume both F and G to be subdivided into patches F_i, $1 \le i \le M$ and G_j, $1 \le j \le N$ such that $B(F_i) \cap B(G_j) = \emptyset$ for all pairs (F_i, G_j). If h is the maximum diameter of the bounding boxes, their numbers M and N are bounded by

$$c \cdot M \cdot h^2 \ \le \text{area}\ (\Omega_F)$$

$$c \cdot N \cdot h^2 \ \le \text{area}\ (\Omega_G)$$

with a constant c, and these worst–case inequalities cannot be improved. The overall amount of work is proportional to the number of patch pairs to be handled, which is in turn proportional to $M \cdot N$ and may reach $\mathcal{O}(h^{-4})$ in the worst situation (consider two parallel planes at distance δ, represented as Bernstein–Bézier tensor products).

So far there is no difference between the boxing strategies. But for a given surface distance $\delta > 0$ the coordinate bounding boxes require $h \sim \delta$ to become disjoint, while oriented parallelepiped boxed only need $h^2 \sim \delta$. This proves the first assertion. The algorithm stops when all diameters of remaining bounding boxes are not larger than ε. But then the coordinate boxes approximate the intersection only with precision ε, while oriented parallelepiped boxes provide precision ε^2 in the direction of the surface normals. If local surface normals are not parallel, this will approximate the intersection along regular transversal curves up to precision ε^2, while degenerations still are handled with precision ε. Here, a *transversal* point $z = F(x) = G(y)$ of the intersection set S of surfaces F anf G is understood as a point where the normals $\eta_F(x)$ and $\eta_G(y)$ to F and G exist and are not parallel. Each transversal point of S is an interior point of a piece of a regular transversal intersection curve. Degenerations of S can only occur in non-transversal points. □

The above analysis is independent of the work W required for testing a single box pair for disjointness, because W will only enter as a fixed factor. For small values of ε and δ it may be

Example 1		Coordinate boxes			Parallelepipeds		
Iterations/Routines		$\mathcal{L}_1$	$\mathcal{R}$	disjoint	$\mathcal{L}_1$	$\mathcal{R}$	disjoint
0	Subdivision and Boxing	24	–	112	23	–	143
1	Subdivision and Boxing	24	–	360	20	–	348
	Test of box diameter	24	0	–	20	0	–
2	Subdivision and Boxing	48	–	336	29	–	291
	Test of box diameter	48	0	–	29	0	–
3	Subdivision and Boxing	71	–	697	30	–	434
	Test of box diameter	71	0	–	30	0	–
4	Subdivision and Boxing	187	–	949	43	–	437
	Test of box diameter	187	0	–	43	0	–
5	Subdivision and Boxing	384	–	2608	39	–	649
	Test of box diameter	384	0	–	39	0	–
6	Subdivision and Boxing	757	–	5387	15	–	609
	Test of box diameter	757	0	–	15	0	–
7	Subdivision and Boxing	1420	–	10692	0	–	240
	Test of box diameter	1420	0	–	0	0	–
8	Subdivision and Boxing	2573	–	20147			
	Test of box diameter	2573	0	–			
9	Subdivision and Boxing	4597	–	36571			
	Test of box diameter	4597	0	–			
10	Subdivision and Boxing	6807	–	66745			
	Test of box diameter	6642	165	–			
11	Subdivision and Boxing	3435	–	102837			
	Test of box diameter	1743	1857	–			
12	Subdivision and Boxing	0	–	27888			
	Test of box diameter	0	0	–			

Table 1 Numbers of patch pairs treated by the divide–and–conquer method

better to use coordinate bounding boxes, but for increased precision requirements the oriented parallelepiped boxes will always pay off.

Table 1 shows an example with $\varepsilon = 0.0001$ and two disjoint surfaces with distance $\delta = 0.000932 \approx 10\varepsilon$, which were given in tensor product Bernstein–Bézier representation. Since there was no parallel machine available, we implemented the loop in the algorithm iteratively, performing a complete sweep over the bag $\mathcal{L}_1$ for each iteration and generating a new instance of $\mathcal{L}_1$ as input for the next iteration. Iteration zero consisted of two subdivision steps without any testing. The column $\mathcal{L}_1$ lists the number of elements in the bag $\mathcal{L}_1$ after termination of each iteration cycle. Similarly, the column $\mathcal{R}$ lists the number of elements in the bag $\mathcal{R}$, while the column labeled "disjoint" contains the number of discarded patch pairs due to the sufficient test for disjointness. The results show how the number of non-discarded oriented parallelepiped boxes stays far below the number of coordinate boxes. In all examples that we considered, the number of coordinate boxes always was larger than the square of the number of oriented paral-

lelepiped boxes (see the proof of Theorem 3.1). Some of the coordinate box pairs even get too small and are moved into $\mathcal{R}$ before they are safely regarded as disjoint. This effect, however, would disappear for smaller values of ε in this example, but it will appear again for cases of disjoint surfaces with $\delta \approx 10\varepsilon$.

4 Marching algorithm

This section describes a curve–following algorithm for surface–surface intersection which is developed from a method in [Diener/Schaback '90] for tracing trajectories in high–dimensional spaces. The basic idea is to use a single parameter α serving as an adaptive stepsize, while the algorithm stays within a fixed tolerance ϵ near the actual curve. Furthermore, we monitor existence and uniqueness of the curve to be followed, and we keep the algorithm efficient, passing all possible degenerations back to the global divide–and–conquer strategy.

Our analysis will be confined to a single pair F, G of surface patches (1.1) and we ignore patch boundaries for a while. Assume two points $F(u_0)$ and $G(v_0)$ to be known where $(u_0, v_0) \in \Omega_F \times \Omega_G$ and

$$\|F(u_0) - G(v_0)\| < \varepsilon.$$

Here ε is chosen so small that we can consider

$$z_0 = \frac{1}{2}\,(F(u_0) + G(v_0)) \tag{4.1}$$

to be a sufficiently good approximation for a point of the intersection of F and G. Recall that these data will in our approach be provided by the global divide–and–conquer algorithm, which does a very early search for an intersection point by a quick–and–dirty method to be explained later. We shall now perform a marching step from (u_0, v_0) to some (u^*, v^*) with

$$\|F(u^*) - G(v^*)\| < \varepsilon,$$

and if this step is actually possible, we will know that there indeed is a unique nondegenerate curve piece of $F(\Omega_F) \cap G(\Omega_G)$ that approximately connects z_0 and $z^* = (F(u^*) + G(v^*))/2$.

We first form (4.1) and the vector

$$r = \eta_F(u_0) \times \eta_G(v_0)$$

from the two surface normals $\eta_F(u_0)$ and $\eta_G(u_0)$. If the norm of $\eta_F(u_0) \times \eta_G(v_0)$ is smaller than a certain small tolerance $\eta > 0$, we do not proceed further. We now formulate Newton's method for calculating an intersection point of the two curves that lies in the subset

$$F(\Omega_F) \cap G(\Omega_G) \cap E(z_0, r, \alpha)$$

of the affine hyperplane

$$E(z_0, r, \alpha) = z_0 + \alpha r + (\mathrm{span}(r))^\perp$$

which is the space $(\mathrm{span}(r))^\perp$ moved along the ray $z_0 + \alpha r$ by a fixed stepsize parameter α (see figure 1). This yields the 4×4 system

$$H(u_0, v_0, r, \alpha; u, v) = \begin{pmatrix} F(u) - G(v) \\ \frac{1}{2}\, r^T(F(u) + G(v)) - \alpha - r^T z_0 \end{pmatrix} = 0 \tag{4.2}$$

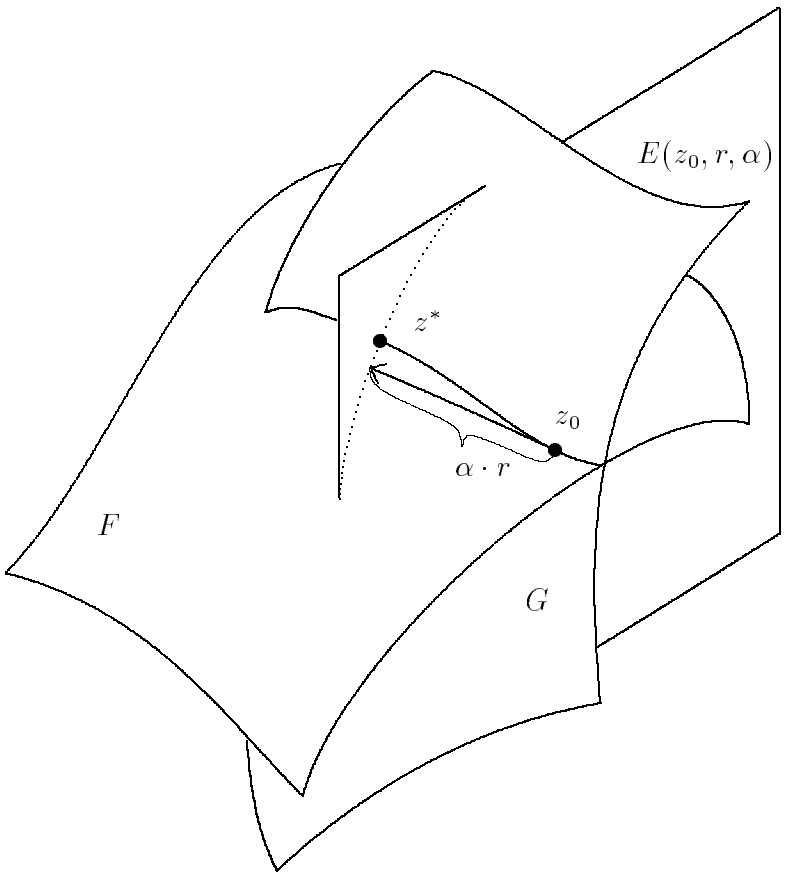


Figure 1 A step of the marching method

for $(u, v) \in \Omega_F \times \Omega_G$ with the Jacobian

$$H'(r, u, v) = \begin{pmatrix} \nabla F(u) & \nabla G(v) \\ \frac{1}{2}\, r^T \nabla F(u) & \frac{1}{2}\, r^T \nabla G(v) \end{pmatrix}$$

being independent of z_0 and α. At the starting point (u_0, v_0) we find

$$\det H'(r, u_0, v_0) = \|\eta_F(u_0) \times \eta_G(v_0)\| \geq \eta > 0 \tag{4.3}$$

after some calculation. Solving the system

$$H'(r, u_0, v_0) \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

for $x, y \in I\!R^2$ yields nonzero solutions x and y of

$$r = \nabla F(u_0)x \;=\; \nabla G(v_0)y \tag{4.4}$$

which we use to define starting points

$$u_\alpha = u_0 + \alpha x, \qquad v_\alpha = v_0 + \alpha y$$

for Newton's method along the ray $(u_0, v_0) + \alpha(x, y)$. From (4.2) and (4.4) we get

$$H(u_0, v_0, r, \alpha; u_\alpha, v_\alpha) = H(u_0, v_0, r, 0; u_0, v_0) + \mathcal{O}(\alpha^2)$$

for $\alpha \to 0$, and we assume

$$\|H(u_0, v_0, r, \alpha; u_\alpha, v_\alpha) - H(u_0, v_0, r, 0; u_0, v_0)\| \leq \gamma_S \cdot \alpha^2 \tag{4.5}$$

for all α from some interval $[\alpha_-, \alpha_+]$ with $\alpha_- \leq 0 \leq \alpha_+$ determined by domain boundaries such that $(u_\alpha, v_\alpha) \in \Omega_F \times \Omega_G$. Here γ_S is a bound on the second derivative of $H(u_0, v_0, r, \alpha; u_\alpha, v_\alpha)$ with respect to α on $[\alpha_-, \alpha_+]$. Now assume

$$\|H'(r; u_\alpha, v_\alpha)\|^{-1} \leq \beta \qquad \text{for all } \alpha \in [\alpha_-, \alpha_+]$$

and

$$\|H'(r; u, v) - H'(r; \tilde{u}, \tilde{v})\| \leq \gamma_N \|(u, v) - (\tilde{u}, \tilde{v})\|$$

for all $(u, v), (\tilde{u}, \tilde{v}) \in \Omega_F \times \Omega_G$. Then, by Kantorovich's theorem [Ortega/Rheinboldt '70], Newton's method when started in (u_α, v_α) will converge for

$$(\|H(u_0, v_0, r, \alpha, 0; u_0 v_0\| + \gamma_S \alpha^2)\beta^2 \gamma_N \leq \frac{1}{2} \, . \tag{4.6}$$

If

$$\varepsilon_0 \beta^2 \gamma_N \leq \frac{1}{4} \tag{4.7}$$

is satisfied for

$$\varepsilon_0 := \|H(u_0, v_0, r, 0; u_0, v_0)\|,$$

we can employ a stepsize $\alpha_1 \in [\alpha_-, \alpha_+]$ with

$$\alpha_1^2 \leq \frac{1}{4\beta^2 \gamma_S \gamma_N} \leq \frac{1}{\gamma_S} \left(\frac{1}{2\beta^2 \gamma_N} - \varepsilon_0 \right),$$

the sign σ of α_1 determining the search direction. Thus we choose α_1 such that

$$\alpha_1 = \left\{ \begin{array}{ll} \min \left(\alpha_+, \frac{1}{2\beta\sqrt{\gamma_S \gamma_N}} \right\}, & \text{if } \sigma > 0 \\ -\max \left(|\alpha_-|, \frac{1}{2\beta\sqrt{\gamma_S \gamma_N}} \right), & \text{if } \sigma < 0 \end{array} \right\} \tag{4.8}$$

and for all α between 0 and α_1 there will be a unique solution (u_α^*, v_α^*) of (4.2). The intersection curve $\alpha \mapsto (F(u_\alpha^*), G(v_\alpha^*))$ can be stably calculated for α between 0 and α_1, but it suffices to store an iterate $(\tilde{u}_{\alpha_1}, \tilde{v}_{\alpha_1})$ with

$$\|F(\tilde{u}_{\alpha_1}) - G(\tilde{v}_{\alpha_1})\| =: \varepsilon_1 \leq \frac{1}{4\beta^2 \gamma_N}$$

as the result of the marching step from (u_0, v_0) using stepsize α_1. The curve is unique in the balls around (u_α, v_α) with radius

$$\|(u_\alpha, v_\alpha) - (u, v)\| \leq \frac{1}{\beta \gamma_N} \left(1 + \sqrt{1 - 2\delta_\alpha}\right) \geq \frac{1}{\beta \gamma_N} \tag{4.9}$$

where

$$\begin{aligned} \delta_\alpha &= \gamma_N \cdot \beta^2 \cdot \|H(u_0, v_0, r, \alpha; u_\alpha, v_\alpha)\| \\ &\leq \gamma_N \cdot \beta^2 \cdot (\varepsilon_0 + \alpha^2 \gamma_S) \leq \frac{1}{2} \, , \end{aligned}$$

by (4.6), and the curve satisfies

$$\begin{aligned} \|(u_\alpha, v_\alpha) - (u_\alpha^*, v_\alpha^*)\| &\leq \frac{1}{\beta\gamma_N}\,(1 - \sqrt{1 - 2\delta_\alpha}) \\ &\leq \frac{1}{\beta\gamma_N}\,(1 - \sqrt{1 - 2\gamma_N\beta^2(\varepsilon_0 + \alpha^2\gamma_S)}) \end{aligned}$$

for all α between 0 and α_1.

In this form the algorithm performs a marching step only if it is sure that it follows a regular curve piece that is at some distance from any other part of the intersection set. Note that we treat uniqueness in the domain $\Omega_F \times \Omega_G$, thus avoiding problems with multiple intersections at different parameters. So far our approach is theoretically sound, but in practice the constants γ_N, γ_S and β are not available. We just use local estimates of

$$\|H'(r; u, v)^{-1}\| \qquad \text{for } \beta$$

and

$$2\,\frac{\|H(\ldots;\tilde{u},\tilde{v}) - H(\ldots;u,v) - H'(\ldots;u,v)((\tilde{u},\tilde{v}) - (u,v))\|}{\|(\tilde{u},\tilde{v}) - (u,v)\|^2} \qquad \text{for } \gamma_N$$

within Newton's iteration. The constant γ_S is estimated right after stepping from (u_0, v_0) to (u_α, v_α), using (4.5) appropriately.

When applying these local estimates we tacitly assume that they are valid in a sufficiently large neighbourhood of our current points, after being multiplied by 5 for safety. Smaller factors proved to be too risky.

Furthermore, we double our estimate of γ_N whenever Newton's method fails to produce a "solution" within five iterations. After a finite number of these updates, our γ_N value must be realistic, because there is a fixed upper bound on γ_N whenever F and G are in C^2. Of course, raising γ_N may cause the algorithm to stop because (4.7) fails, but this case is considered as a degeneration and is left to the global divide–and–conquer strategy. Similarly, we refuse to proceed if β, γ_N or γ_S exceed certain large a–priori bounds, which are dependent on the machine precision and the accuracy of the linear equation solver.

Theorem 4.1 *If the constants β, γ_N, and γ_S of the marching method are estimated realistically, the method will follow a nondegenerate intersection curve of length L with accuracy ε by a computational effort of order $L|\log|\log\varepsilon||$ for large L and small ε. This compares favorably with the work of order $L\varepsilon^{-1/2}$ required by the divide–and–conquer method. Finally, the marching algorithm makes sure that there is no other solution branch within a known distance to the calculated solution.*

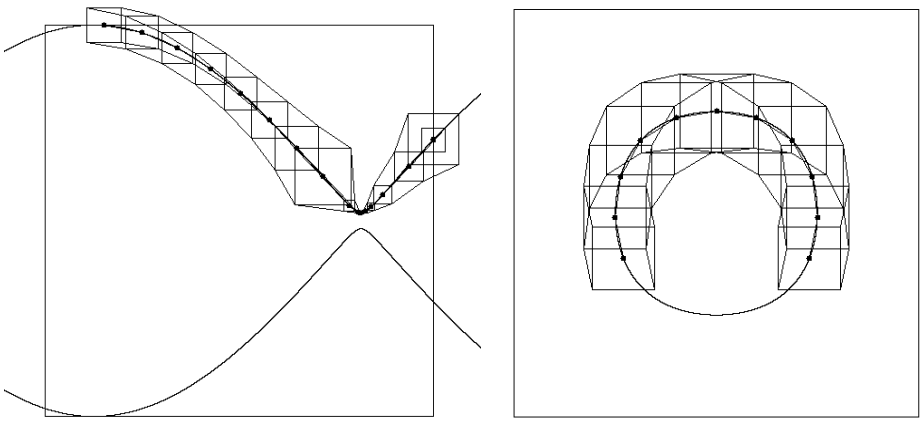
Proof: The stepsize α of the method cannot go to zero since β, γ_N and γ_S are bounded from above. Thus the number of curve–following steps is at most proportional to L, while the number of Newton steps is proportional to $|\log|\log\varepsilon||$ due to quadratic convergence. This proves the complexity statement. The constant in this $\mathcal{O}(L \cdot |\log|\log\varepsilon||)$ bound depends on the constants η, β, γ_S, and γ_N controlling the non–degeneracy of the intersection curve. The second assertion is obvious from the Newton–Kantorovich theorem.

To yield an inclusion of an intersection curve of length L with precision ε the divide–and–conquer method finally needs at least N patches of maximum diameter h, where $N \cdot h \approx L$ and $h^2 \approx \varepsilon$. Thus $N \simeq L \cdot \varepsilon^{-1/2}$ patches are necessary for a curve on a single surface. If disjoint patch pairs are disposed immediately, there will be approximately $const. \cdot N \approx L \cdot \varepsilon^{-1/2}$ patch pairs along a nondegenerate curve of length L, and the final step of the divide–and–conquer method will be of complexity at least $\mathcal{O}(L \cdot \varepsilon^{-1/2})$ for large L and small ε. □

Theorem 4.1 confirms the observation that curve–following methods are much more effective than divide–and–conquer methods.

The marching method will automatically stop in the vicinity of degenerations like bifurcation points. When applied in the interior of patches, a criterion for prevention of loops is required which can easily be implemented by stopping whenever $r_{new}^T r_{old} \leq 0$ (see [Sederberg/Meyers '88] for more sophisticated criteria). Since we normally start the algorithm on (u, v) where u or v is on a patch boundary, choosing the proper sign σ in (4.8), we only require to stop it at patch boundaries, which is easy. Our marching method cannot cycle within a patch when coming in from the boundary, because it must then necessarily run into a degeneration and stop.

The output consists of a sequence of points $(u_i, v_i) \in \Omega_F \times \Omega_G$, $1 \leq i \leq k$, where the first and last pair have at least one component on a domain boundary. Furthermore, we save the local uniqueness radii $\frac{1}{\beta\gamma_N}$ from (4.9) for later use; note that these do not depend on ε and get larger if β or γ_N gets smaller. Since γ_N is dependent on the second derivatives of patches, it will be small when patches are "flat" in the sense of a "flatness test", e.g. after a large number of subdivision steps. Thus flat patches will lead to large stepsizes and large uniqueness radii, improving the performance of the marching method. To illustrate the behavior of the marching algorithm we picked a hazardous case with two intersection curves that come very close. Figure 2 shows the sequence of L_∞ uniqueness balls derived from (4.9) in $\Omega_F = [0,1]^2$, centered around the points calculated by the marching algorithm. Note how the algorithm flexibly adjusts the stepsize and the uniqueness radius (4.9) and how it avoids overshooting to the other curve. For cases which are less risky, the uniqueness radius attains reasonably large values (see Figure 3). This will significantly reduce the amount of work necessary to exclude the existence of additional solution branches, as will be seen below.



Figures 2 and 3 Uniqueness boxes along intersection curve in Ω_F.

5 Multistage algorithm

We now combine the divide–and–conquer algorithm of section 2 with the marching method of the previous section. Recall that the divide–and–conquer algorithm, when linked to a local method, will test each non–discarded patch pair very early for a possible intersection point. We implemented this in a very primitive way by trying to solve $F(u) = G(v)$ on each patch boundary curve via Newton's method started in the corners of the patch. We stopped Newton's method when it left the patch boundary or when it ran into numerical difficulties. Furthermore, we simply neglected the possibility of interior intersection points and of all kinds of singularities, because we could rely on later subdivision steps and the overall safety of the pure divide–and–conquer method. See [Müllenheim '91] for a much more refined approach to the construction of starting points.

Algorithm 5.1.(Local part of the multistage algorithm)
Data: Bags $\mathcal{L}_1, \mathcal{L}_2$, and $\mathcal{R}$ as in the divide–and–conquer method.
Another bag $\mathcal{P}$ for results.
Loop: While $\mathcal{L}_2 \neq \emptyset$ let any number of parallel processes do:

Take (F, G) from $\mathcal{L}_2$.

if diam $B(F) \leq \varepsilon$ and diam $B(G) \leq \varepsilon$
then transfer pair to $\mathcal{R}$ and repeat loop.

Subdivide F and/or G into subpatches F_i, G_j
as in the divide–and–conquer algorithm.
For all pairs (F_i, G_j) of subpatches do :

if $B(F_i) \cap B(G_j) = \emptyset$
then discard (F_i, G_j)
else

if no traceable curve is found in (F_i, G_j),
then put (F_i, G_j) into $\mathcal{L}_1$
else output curve data to $\mathcal{P}$ and

if neither $B(F_i)$ nor $B(G_j)$ lies
in a uniqueness box for this curve,
then put (F_i, G_j) into $\mathcal{L}_2$
else discard (F_i, G_j).

The overall data flow of the complete multistage algorithm is visualized in Figure 4, where one can imagine any number of parallel processes executing the global and the local part. Synchronization is only required at the access to the bags, and there are many possibilities to handle this, depending on the storage and network model of the parallel architecture.

Post–processing the resulting curve pieces is easy, because we store lists of intersection points starting and ending at domain boundaries plus the radii of the uniqueness balls in the domains at these points. This information allows to assemble the curves from their pieces in a very reliable way. Much more difficult is the treatment of singularities; we are confronted with a nonempty bag $\mathcal{R}$ of possibly non–disjoint, but rather small patch pairs that do not contain