GPU-Based Parallel Solver via Kantorovich Theorem for The Nonlinear Bernstein Polynomial Systems

Feifei Wei, Jieqing Feng*, Hongwei Lin

State Key Lab of CAD&CG, Zhejiang University, 310058, China

Abstract

This paper proposes a parallel solver for the nonlinear systems in Bernstein form based on subdivision and the Newton-Raphson method, where the Kantorovich theorem is employed to identify the existence of a unique root and guarantee convergence of the Newton-Raphson iterations. Since the Kantorovich theorem accommodates a singular Jacobian at the root, the proposed algorithm performs well in a multiple root case. Moreover, the solver is designed and implemented in parallel on Graphics Processing Unit(GPU) with SIMD architecture; thus, efficiency for solving a large number of systems is improved greatly, an observation validated by our experimental results.

Keywords: nonlinear system, Kantorovich theorem, parallel computing, tensor, tangent root, normal cone

1. Introduction

Root finding of a nonlinear system in terms of B-spline or Bernstein polynomials is a fundamental problem in various geometric modeling applications [5]. Analytical solutions only exist for univariate polynomials of degree no more than 4. When the degree of the polynomial or the number of constraints increases, an efficient and robust solver is considered a difficult problem. Many approaches have been proposed to address this problem, such as Descartes rules [2, 13], interval arithmetic [14], and resultant theory [12] *etc.* However, the subdivision-based approach is more attractive for geometric modeling applications due to their geometric significance.

The geometric approach fully exploits the inherent convex hull property and the numerical stability of Bernstein polynomials or B-spline basis functions. The subdivision method proposed by Lane and Riesenfeld [9] is a pioneering work that can solve a univariate Bernstein polynomial equation robustly. The Bézier clipping method proposed by Nishita *et. al.* [16] is an improved subdivision method and is applied to ray-tracing rational parametric surface patches. The Projected Polyhedron algorithm [21] developed by Sherbrooke and Patrikalakis is a generalization of the Bézier clipping method for the multivariate case, and has been applied to solving many nonlinear problems, such as surface and surface intersection, offset, medial axis, and other shape interrogation problems [17]. Instead of bisecting the domain directly, the clipping approaches clip the domain more elaborately according to the convex hull of control points, exploiting the advantages of the Bernstein polynomials. Thus, besides polynomial subdivision cost, there are additional computational costs of the convex hull and intersection in clipping approaches. For the case of a single root, each step of the projected polyhedron algorithm can purge away a larger no-root interval than that of the subdivision method. However, for the case of many roots, a large number of inefficient clipping steps at an early stage will pay off benefits at a later stage [5]. Recently, Mourrain and Pavone [15] proposed a preconditioned improvement that can reduce the steps for both subdivision-based and reduction-based (*i.e.* clipping) methods.

Rather than solving a nonlinear Bernstein polynomial system directly, if all of the different roots could be isolated via polynomial subdivision or domain clipping, the roots could be solved more efficiently by using other numerical methods, *e.g.*, the Newton-Raphson method, where the center of a reduced sub-domain containing an isolated root can be adopted as an initial guess. The concept of the normal cone was first proposed by Sederberg and Meyers [20] to identify the existence of loops in a surface and surface intersection. Elber and Kim [5] employed the normal cone to deduce a subdivision termination criterion that can isolate all of different roots of a nonlinear Bernstein polynomial system. Then the quadratically convergent Newton-Raphson method is adopted to approximate each isolated root. Furthermore, Hanniel and Elber [6] develop a computationally tractable approach, namely dual representation, to check whether this criterion is met or not. However, this subdivision termination criterion can not guarantee convergence of the Newton-Raphson iterations with the specified initial guess, which is the center of the isolated sub-domain. According to the normal cone test, an isolated sub-domain contains a single root at most. If the multiple root is a tangent case, the subdivisions will

*Corresponding author, jqfeng@cad.zju.edu.cn

be performed until sub-domain size reaches the tolerance.

In this paper we adopt Kantorovich theorem to address the above convergent problem. Given an initial guess in a domain, in which the Jacobian of system should be Lipschitz continuous, the Kantorovich theorem can determine whether the Newton-Raphson iteration is well-defined. If conditions are satisfied in the Kantorovich theorem, there will be two concentric regions surrounding the initial guess: the large one is the region in which unique zero exists; the smaller one contains all of the Newton-Raphson iteration sequences, in which they will converge to the unique zero. Furthermore, the Kantorovich theorem does not assume nonsingularity of Jacobian matrix $J(\mathbf{x}_*)$ at zero. This is helpful for solving the multiple root case, since we can improve the efficiency of root finding by terminating the subdivision earlier than the normal cone based method.

With the rapid development of GPGPU (General Purpose computing on Graphics Processing Units), graphics hardware is becoming a new attractive parallel computing platform. The proposed subdivision-based nonlinear system solver based on Kantorovich theorem is tailored for SIMD architecture of contemporary GPUs. Thus, the significant performance speedup for a large number of nonlinear systems can be gained.

There are three major contributions in this paper. Firstly, by using the Kantorovich theorem, we can not only identify the existence of unique root, but also guarantee the convergence of the Newton-Raphson iteration with a suitable initial guess. Secondly, the multiple root of tangential case can be solved more efficiently. Thirdly, the proposed nonlinear solver is designed for the SIMD architecture of GPUs. Thus, we can gain better performance than the corresponding single-threaded solver on CPU, especially for a large number of nonlinear systems.

The paper is organized as follows. In Section 2, tensor representation and operations of Bernstein polynomials are introduced briefly. In Section 3, the subdivision-based solver based on Kantorovich theorem is given first, then its parallel version designed for the SIMD architecture is introduced, and the process of multiple root case is described in detail. In Section 4, numerical examples and discussions are given. Finally, we provide conclusions.

2. Tensor Preliminaries

Consider a nonlinear system as follows:

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} : \mathbb{R}^n \to \mathbb{R}^n \tag{1}$$

where $\mathbf{F} = (F_1(\mathbf{x}), F_2(\mathbf{x}), \cdots, F_n(\mathbf{x}))^T$. Its roots are real points $\{\mathbf{x}_*\}$ in $\mathbb{R}^n$, such that $F_i(\mathbf{x}_*) = 0$, for $i = 1, \cdots, n$. In this paper, it is assumed that each nonlinear constraint is a Bernstein polynomial and the number of variables equals to that of constraints. An under-determined system can be reduced to the above case by exhaustively sampling some variables according to subdivision tolerance.

The tensor representation can facilitate arithmetic operations related to Bernstein polynomials on SIMD architecture GPU [11]. A tensor is a higher dimensional analog of a matrix, where the number of indices is the rank of tensor. For example, a planar algebraic curve can be represented as a rank 2 tensor (or a matrix), and an algebraic surface in $\mathbb{R}^3$ can be represented as a rank 3 tensor.

There are three operations associated with a rank n tensor $\mathfrak{F}_{e_1 e_2 \cdots e_n}$ of multivariate constraint, *i.e.*, contraction, transformation, and norm estimation. They are defined as follows:

$$\begin{aligned} &\mathbf{F}(\mathbf{x}) = \mathbf{x}_1^{e_1}\mathbf{x}_2^{e_2}\cdots\mathbf{x}_n^{e_n}\mathfrak{F}_{e_1 e_2 \cdots e_n} \\ &\mathbf{T}(F) = \tilde{\mathfrak{F}}_{f_1 f_2 \cdots f_n} = \mathbf{T}_{f_1}^{e_1}\mathbf{T}_{f_2}^{e_2}\cdots\mathbf{T}_{f_n}^{e_n}\mathfrak{F}_{e_1 e_2 \cdots e_n} \\ &\mathbf{Norm}(F) = \|\mathfrak{F}_{e_1 e_2 \cdots e_n}\| \end{aligned} \tag{2}$$

where tensors($\mathfrak{F}$ and $\tilde{\mathfrak{F}}$) are represented using Einstein index notation e_is and f_is, $\mathbf{x}_i^{e_i}$ is the vector of d degree Bernstein polynomial $\{B_j^d(x_i)\}_{j=0}^d$. Tensor contraction corresponds to evaluation of a multivariate constraint in Equation (1). Tensor transformation corresponds to a subdivision operation, which transforms one tensor on a given domain to a new one on its sub-domain. Both tensor contraction and transformation are alternatives of de Casteljau's algorithm for Bernstein polynomial. Using tensors is more suitable for GPU implementation. The two operations are similar with matrix-vector and matrix-matrix multiplications, respectively. Norm estimation gives a measurement of tensor magnitude, which is useful in the Kantorovich theorem. The norm estimation will be explained in detail in Section 3.2.

3. Kantorovich Solver

3.1. Normal cones test

For each Bernstein polynomial in Equation (1), if coefficient signs of $F_i(\mathbf{x})$ are all positive or negative, there will be no root in its domain according to convex hull property of Bernstein polynomial. The corresponding domain can be discarded. It is called the **C**ontrol **C**oefficients **S**igns test, abbreviated as **CCS** test here. It can be described as:

Algorithm 1 CCS test

1: **Input**: $\{F_i(\mathbf{x})\}_{i=1}^n$, // Bernstein polynomials;
2: For each $F_i(\mathbf{x})$
   $c_i^{min}, c_i^{max} \Leftarrow$ min & max coefficients of $F_i(\mathbf{x})$;
3: **if** $\exists i$ such that $c_i^{min} c_i^{max} > 0$ **then**
4: **return False**; //no root exist
5: **else**
6: **return True**
7: **end if**

The normal cone of a surface can be regarded as a "bounding box" of surface normal. If all the normal cones of $\{F_i(\mathbf{x})\}_{i=1}^n$ have no intersection in a domain, the domain will contain at most one connected component of

the solution set, *i.e.*, one root [20]. It is called the **N**ormal **C**ones test, abbreviated as **NC** test here. The **NC** test is equivalent to the singularity test of the Jacobian matrix of Equation (1). Otherwise, the domain should be subdivided recursively until all of single roots are isolated or the domain size reaches a prescribed threshold. Then the quadratically convergent Newton-Raphson method can be employed to approximate each single root.

In most cases, the Newton-Raphson iteration will converge to the isolated root efficiently if the center of the subdomain is chosen as the initial guess. However, this initial guess, which lies in a sub-domain with non-singularity Jacobian, cannot always guarantee the convergence of the Newton-Raphson iteration. Figure 1 shows a counterexample, where two planar curves intersect at two points. Two circular regions indicate the convergent regions, in which any initial guesses can produce convergent iteration to the intersection points, *i.e.*, the roots. The dash curve is loci of singular Jacobian, on which the normal cone test will fail. The rectangular region is a termination sub-domain that passes the **NC** test. There is no intersection between the dash curve and the rectangular region. Though there is a unique root in this domain, the center $\mathbf{x}_0$ of rectangular domain is not a convergent initial guess. This counterexample shows that the subdivision termination criteria via **NC** test is not a sufficient condition of the Newton-Raphson iteration convergence. Although experimental results show that adoptions of both **NC** and **CCS** tests could purge away a lot of potential poor initial guesses, a rigorous proof of convergent conditions of the Newton-Raphson iteration by using the specified initial guess is still absent [5]. Another problem arises from the multiple root case. Since in a sub-domain containing a multiple root, the **NC** test will always fail. The method will keep on subdividing the domain till the size of subdomain is less than the prescribed threshold.

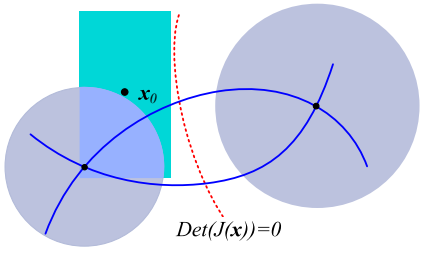

Figure 1: The initial guess $\mathbf{x}_0$ obtained via **CCS** test and **NC** test cannot always guarantee the convergence of the Newton-Raphson iteration.

3.2. Kantorovich theorem

Besides root-isolations via **NC** test or **CCS** test, there are also other works to study the local convergence of the Newton-Raphson iteration. Among them, Kantorovich theorem is an elegant and powerful one, in that it makes no assumption about the existence of a zero and nonsingularity of its Jacobian matrix. For the convenience, we introduce the Kantorovich theorem described in [3].

KANTOROVICH THEOREM Let $r > 0$, $\mathbf{x}_0 \in \mathbb{R}^n$, $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ and assume that $\mathbf{F}$ is continuously differentiable in an open neighborhood $N(\mathbf{x}_0, r)$ of radius r around $\mathbf{x}_0$. For a given vector norm and the induced operator norm, $J \in Lip_\gamma(N(\mathbf{x}_0, r))$ and $J(\mathbf{x}_0)$ is nonsingular, and there exist constants $\beta, \eta \geq 0$ such that:

$$\|J(\mathbf{x}_0)^{-1}\| \leq \beta, \|J(\mathbf{x}_0)^{-1}\mathbf{F}(\mathbf{x}_0)\| \leq \eta.$$

Define $\alpha = \beta\gamma\eta$. If $\alpha \leq \frac{1}{2}$ and $r \geq r_0 \equiv \frac{1-\sqrt{1-2\alpha}}{\beta\gamma}$, then sequence $\mathbf{x}_k$ defined as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - J(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{x}_k), k = 0, 1, \cdots$$

is well defined, and converges to $\mathbf{x}_*$. It is a unique zero of $\mathbf{F}$ in the closure of $N(\mathbf{x}_0, r_0)$. If $\alpha < \frac{1}{2}$, then $\mathbf{x}_*$ is the unique zero of $\mathbf{F}$ in $N(\mathbf{x}_0, r_1)$, where $r_1 \equiv min[r, \frac{1+\sqrt{1-2\alpha}}{\beta\gamma}]$ and:

$$\|\mathbf{x}_k - \mathbf{x}_*\| \leq (2\alpha)^{2^k}\frac{\eta}{\alpha}, k = 0, 1, \cdots .$$

To meet the conditions of Kantorovich theorem, we should estimate the norm of the inverse Jacobian matrix at initial guess β, first iteration step length η and Jacobian Lipschitz continuous constant γ in its neighborhood for the nonlinear system (1). The constants β and η can be trivially computed if a vector norm and its induced operator norm of the matrix are defined. The Jacobian Lipschitz continuous constant γ indicates a variation rate of Jacobian matrix under the induced norm. Thus, it can be estimated via second-order partial derivatives of the system, *i.e.*, Hessian. For simplicity, we call the conditions in Kantorovich theorem **K**C in the rest of paper.

For the system (1), its Jacobian is a $n * n$ rank 2 function tensor, and its Hessian is a $n * n * n$ rank 3 function tensor. Each element of the Hessian function tensor is a second partial derivative function of F_i, which is also a Bernstein polynomial. Due to convex hull property, the maximum absolute value (a scalar) of control coefficients can be adopted as a norm of the second derivative function. As a result, we can reduce norm estimation of a function tensor to a problem of a scalar tensor. However, the norm of a scalar tensor is still a problem, which can not be evaluated efficiently yet.

Qi[18] shows that a super-symmetric tensor has similar properties as a symmetric matrix, such as its trace, eigenvalues, *etc.*, which are invariant under coordinate transformations. Lim[10] adopts a constrained variational approach to calculate the eigenvalues, eigenvectors, singular values, and singular vectors of a tensor. Both of two approaches convert the eigenvalue or singular value computation problem to the root-finding problem of a non-linear system. It is still expensive and sometimes not feasible. In our setting, the Hessian H_i of each F_i in system (1) is a $n * n$ matrix, which can be evaluated conveniently if the matrix norm is defined, and we can define the Hessian norm of system (1) as the maximum norm of vector

$\{H_1, H_2, \cdots, H_n\}$

$$\|H\| = max\{\|H_i\|_\infty, 1 \le i \le n\} \tag{3}$$

as an approximation of Jacobian Lipschitz constant γ in the Kantorovich theorem. Owing to convex hull property of Bernstein polynomials and simple norm computations of matrix and vector, the above estimations require less temporal and spatial computing resources than those in [18] and [10], but they are conservative to some extent.

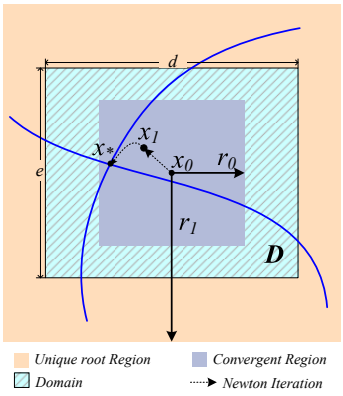

Figure 2: Illustration of root distribution according to Kantorovich theorem. $N(\mathbf{x}_0, r_0)$ and $N(\mathbf{x}_0, r_1)$ are both define in term of infinity norm. Note that the convergent regions in Figure 1 are defined in terms of 2-norm.

There are two concentric neighborhoods $N(\mathbf{x}_0, r_0)$ and $N(\mathbf{x}_0, r_1)$ in the Kantorovich theorem. The outer one $N(\mathbf{x}_0, r_1)$ is the region in which there is a unique root, while the inner one $N(\mathbf{x}_0, r_0)$ is the convergent region of subsequent the Newton-Raphson iterations, in which $\mathbf{x}_0$ is adopted as initial guess. Figure 2 shows an example. Two planar algebraic curves intersect at one point $\mathbf{x}_*$. The neighborhoods $N(\mathbf{x}_0, r_0)$, $N(\mathbf{x}_0, r_1)$ and the sub-domain $\mathbf{D}$ are depicted with different colors. If the long edge d of the sub-domain $\mathbf{D}$ satisfies $\frac{d}{2} \le \frac{1+\sqrt{1-2\alpha}}{\beta\gamma} = r_1$, then the sub-domain $\mathbf{D}$ is in the neighborhood $N(\mathbf{x}_0, r_1)$ completely. Thus there is a unique root in $\mathbf{D}$. In other words, we can use $\mathbf{D}$ to isolate all the zeros as in [5]. Furthermore, if $r_0 \le \frac{e}{2}$, *i.e.*, a half of the short edge length of $\mathbf{D}$, the neighborhood $N(\mathbf{x}_0, r_0)$ is in $\mathbf{D}$. Thus, the Newton-Raphson iterations with initial guess $\mathbf{x}_0$ would converge to the unique root $\mathbf{x}_*$, the intersection point in the $N(\mathbf{x}_0, r_0)$.

Otherwise, if $\frac{d}{2} > r_1$, *i.e.* $\mathbf{D}$ is not in the neighborhood $N(\mathbf{x}_0, r_1)$, there may be more than one root in D; if $\frac{e}{2} < r_0$, *i.e.* $N(\mathbf{x}_0, r_0)$ is not in the sub-domain $\mathbf{D}$, the Newton-Raphson iteration with initial guess $\mathbf{x}_0$ may not be convergent in the sub-domain $\mathbf{D}$. In these cases, we should subdivide the sub-domain $\mathbf{D}$ further so that we can delimit the unique root and convergent region. Traditional bisection method subdivides a domain at the middle point of long edge. To facilitate the subdivisions in parallel, we bisect a domain along its all edges simultaneously. Because the constraints are defined on $[0, 1]^n$, 2^n equilateral sub-domains will be obtained after one subdivision. The pseudo codes of **KC** test and the recursive root-finding algorithms are given in Algorithms 2 and 3, respectively. Note that if the **KC** test is passed, its output is adopted as the initial guess for the subsequent Newton-Raphson iteration to find the optimal roots of the nonlinear system.

Algorithm 2 KC test

Input: a nonlinear system $\mathbb{P} =< \{F_i\}_i^n, \mathbf{D} >$, n functions of n-variable defined on a domain $\mathbf{D} = \{(u_j^{min}, u_j^{max})\}_{j=1}^n$ as in Equation 1;
For each function F_i: Compute all second partial derivative functions $\{\frac{\partial^2 F_i}{\partial x_s \partial x_t}\}$. Let $C_{st}^{(i)}$ be the maximum absolute value of control coefficients of function $\frac{\partial^2 F_i}{\partial x_s \partial x_t}$, the Hessian tensor of F_i ($n \times n$ matrix) is defined as $H_i = \{C_{st}^{(i)}\}_{s,t=0}^n$;
$\gamma = \max(\|H_i\|_\infty)$, $i = 0, \cdots, n$;
$d \Leftarrow$ long edge of domain $\mathbf{D}$;
$e \Leftarrow$ short edge of domain $\mathbf{D}$;
$\mathbf{x}_0 = (\frac{u_1^{min}+u_1^{max}}{2}, \cdots, \frac{u_{n-1}^{min}+u_{n-1}^{max}}{2})$, center of $\mathbf{D}$
$\beta = \|\mathbf{J}^{-1}(\mathbf{x}_0)\|$, $\eta = \|\mathbf{J}(\mathbf{x}_0)^{-1} \cdot \mathbf{F}(\mathbf{x}_0)\|$, $\alpha = \beta\gamma\eta$;
$r_0 = (\frac{1-\sqrt{1-2\alpha}}{\beta\gamma})$, $r_1 = (\frac{1+\sqrt{1-2\alpha}}{\beta\gamma})$;
if $(\alpha < \frac{1}{2})\&\&(r_1 \ge \frac{d}{2})\&\&(r_0 \le \frac{e}{2})$ **then**
  return True
  // $\mathbf{x}_0$ can guarantee convergent iterations
else
  return False
end if

Compared with the algorithm in [6], Algorithm 3 is more space and time consuming because of the adoption of the Hessian. For the system with n constraints and n variables, both storage and subdivision costs of the Hessians are $O(n^3)$, while those of Jacobian in [6] are $O(n^2)$. Because only maximum estimation of each entry of Hessian is necessary in our algorithm, the additional memory for storage of Hessian is negligible. Meanwhile the costs of normal cone construction in [1] and their intersection computation via dual hyperplanes representation in [6] can be avoided. In each subdivision step, the **NC** test takes just about the same time as the **KC** test, according to the examples in Section 4.

3.3. Kantorovich solver on SIMD architecture

Driven by high performance and high quality 3D graphics applications, the programmable GPU has evolved into a highly parallel, multithreaded, manycore processor with tremendous computational horsepower and a very high memory bandwidth. A modern GPU usually contains hundreds of low cost stream processors. It is becoming an attractive platform for general-purpose computing.

However, most of state of art GPU programming languages or application programming interfaces, such as HLSL, GLSL, OpenCL, DirectX Compute *etc.*, do not support recursive functions. Even though CUDA v3.0 augments the support of recursion feature recently, it is still far

Algorithm 3 Kantorovich root-finding **KanSub**$(\mathbb{P}, \tau)$

1: **Input:** a nonlinear system $\mathbb{P}$ as in Algorithm **KC** test; τ, tolerance of subdivision resolution
2: **if** $\max(u_j^{min} - u_j^{max}) < \tau$ **then**
3: **return** $\{(\frac{u_1^{min}+u_1^{max}}{2}, \cdots, \frac{u_{n-1}^{min}+u_{n-1}^{max}}{2})\}$; // As a root under subdivision threshold τ
4: **else**
5: **if** (**CCS**) **then**
6: **if** (**KC**) **then**
7: **return** $\{(\frac{u_1^{min}+u_1^{max}}{2}, \cdots, \frac{u_{n-1}^{min}+u_{n-1}^{max}}{2})\}$ // As initial guess of the Newton-Raphson iteration
8: **else**
9: subdivide $\{F_i\}$ into a set of new $\{F_i^j\}, j = 1, \cdots, 2^n$ at the middle of each direction.
10: **return** $\bigcup_{j=1}^{2^n}$ **KanSub**$(< \{F_i^j\}_{i=1}^n, \mathbf{D}^j >, \tau)$
11: **end if**
12: **end if**
13: **end if**

Algorithm 4 Kantorovich root-finding **in Parallel**

1: **Input:** a set of nonlinear systems $\{\mathbb{P}\}$; τ, tolerance of subdivision resolution
2: **while** $\{\mathbb{P}\} \neq \emptyset$ **do**
3: **if** $\max(u_j^{min} - u_j^{max}) < \tau$ **for each** $\mathbb{P}$ **then**
4: **return** $\{(\frac{u_1^{min}+u_1^{max}}{2}, \cdots, \frac{u_{n-1}^{min}+u_{n-1}^{max}}{2})\}$ // As a root under subdivision threshold τ
5: **else**
6: subdivide $\{F_i\}$ into T sub-functions **for each** $\mathbb{P}$, label ones that pass **CCS** test;
7: **scan** sub-function been labeled, obtain new $\{\mathbb{P}\}$;
8: **if** $\{\mathbb{P}\} = \emptyset$ **then**
9: **return**
10: **end if**
11: construct H_i and perform **KC** test **for each** sub-function F_i, accept subdomain centers pass the **KC** test, label failed ones;
12: **scan** subdomains been labeled, obtain new $\{\mathbb{P}\}$;
13: **if** $\{\mathbb{P}\} = \emptyset$ **then**
14: **return**
15: **end if**
16: **end if**
17: **end while**

from being practical for complex geometric applications. As a result, for inherently recursive algorithms such as subdivision-based Kantorovich solver, we implement its iterative version via Breadth-first search tailed for SIMD architecture. This is described in Algorithm 4. Each iterative step contains two phases: firstly, the constraints $F_i(\mathbf{x})$ are subdivided and the sub-domains containing no zeros are discarded via **CCS** test. Secondly, the Kantorovich theorem is employed to identify valid initial guesses in the sub-domains that passed the **CCS** test, which are sent to the subsequent Newton-Raphson iterations. The sub-domains that fail the **KC** test will be scanned and gathered together for further subdivision. The subdivisions and **CCS** and **KC** tests are performed until there is no feasible sub-domain.

To exploit parallelism of GPUs, each F_i of n variables is subdivided uniformly into $T = m^n$ ones, instead of bisection (2^n) in Algorithm 3. Obviously, a more dense subdivision could result in a smaller subdivision depth. However, limited resources in GPU will pay back the benefits of increased parallelism. According to capacity of state of art GPU, T can be set as a constant 64 for two and three dimensional cases, *i.e.*, $m = 8, 4$ for $n = 2, 3$ respectively. If tolerance of sub-domain size τ is set to $1.0e-6$ for single precision floating-point arithmetic in GPU, the maximum subdivision depth will be 7 and 10 for $n = 2$ and 3 respectively in worst case in our algorithm, while the maximum depth are 20 in bisection approaches.

The most time-consuming step in our algorithm is the subdivision of the constraints. The classical de Casteljau's algorithm is more suitable for bisection, while the blossoming algorithm [19] can be applied to an arbitrary subdivision with more loops. Although the blossoming algorithm is numerically stable, it will introduce many deep loop instructions, which will lead to a significant performance drop of instruction throughputs on GPU.

As an alternative approach, a multivariate Bernstein polynomial is represented in tensor form. The control coefficients tensor is associated with two operations, *i.e.* contraction and transformation, as described in Section 2. The contraction operation corresponds to an evaluation of a Bernstein polynomial, which is implemented as a matrix-vector multiplication. The transformation operation corresponds to a subdivision of a Bernstein polynomial, which can be understood as a "matrix-matrix" multiplication. The control coefficients of a tensor on a sub-domain can be obtained by sequential tensor transformations of the entire domain along each direction. If we subdivide a Bernstein polynomial of degree d into m ones uniformly, the k-th transformation tensor $\mathbf{T}_{f_i}^{e_i}$ in Equation (2) can be obtained via the following formula:

$$\mathbf{T}_{f_i}^{e_i} = \left\{ \sum_{0 \le a,b \le j}^{a+b=j} B_a^{d-i}\left(\frac{k}{m}\right) * B_b^i\left(\frac{k+1}{m}\right) | 0 \le i,j \le d \right\}$$

where $B(x)$ is Bernstein basis function. If the system in (1) is uniformly subdivided into m^n ones, we will compute $n * m$ transformation tensors totally. For 2D case, we will compute $2 * 8$ rank 2 transformation tensors for $T = 8^2 = 64$ in our setting. Since these transformation tensors are shared by all T polynomials on the sub-domains for each iteration step, we can replace the expensive blossoming computation with pre-computation and fast runtime texture fetches on GPUs.

The **scan** operations of lines 7 and 12 in Algorithm 4

are to collect the sub-domains that pass **CCS** and **KC** tests. Implementing a sequential scan operation in single thread on a CPU is trivial. Harris *et. al.* [8] proposed a parallel version of scan operation on GPU that has same complexity with that of CPU one. The parallel versions of other operations such as **reduce** and **compact** have been proposed and exposed in [7]. The **scan** operation allows us to compact the intermediate sub-domains generated in each subdivision step. Otherwise, the number of sub-domains will increase exponentially. As the result of the **scan** operation, the number of sub-domains to be checked via **KC** test in each step will approach the number of roots, which saves considerable storage space.

3.4. Tangent Root (A Case of Multiple root)

If a domain contains a multiple root, the nonlinear system will always fail the **NC** test. Thus, the subdivision-based methods like [5] will keep on subdividing the domain until the size of sub-domain is less than the threshold. In this case, a large number of subdivisions will seriously affect the efficiency of algorithm, even if there is a good initial guess in the sub-domain. Algorithm 3 and 4 would behave the same way, when the center of domain, *i.e.* the selected initial guess, can not pass the **KC** test. However, the Kantorovich theorem considers both the domain and the initial guess simultaneously, rather than the domain separately, it is still possible for us to find a good guess from different alternatives.

Here, we propose a solution to this problem. The main idea is to consider as many as possible initial guesses other than just the center in the **KC** test, so that we can choose those well-defined guesses. The local distribution of the root around these initial guesses can give us more information to purge away useless regions according to the Kantorovich theorem. First, we sample $T(= m^n)$ points uniformly in the current domain, which are also the centers of T sub-domains in next subdivision step. For each sub-domain in the next subdivision step, if its unique root region $N(\mathbf{x}_0, r_1)$, obtained via **KC** test, does not contain the sub-domain, the sub-domain needs to be subdivided further; otherwise, there is exactly one root in the sub-domain; furthermore, if $N(\mathbf{x}_0, r_1)$ covers the neighboring sub-domains completely, the covered neighboring sub-domains can be purged away since there is just one root in $N(\mathbf{x}_0, r_1)$. Finally, we obtained a set of unique root sub-domains $N(\mathbf{x}_0, r_1)$. Then we choose $\mathbf{x}_0$ as an initial guess in each sub-domains so that the Newton-Raphson iterations are convergent in it. This is trivial work.

We implement this scheme by slightly modifying the Algorithms 4. Since we perform T **KC** tests rather than just the center in each step, it is necessary to minimize the costs of extra **KC** tests by reusing γ. Firstly, we store the estimations of γ in k-th step, $k = 0, 1, \cdots$. Then we calculate β and η just after the subdivision of F_i and **CCS** test in 6th line of $(k+1)$-th step of Algorithm 4. As a result, the **KC** test is performed at that moment while adopting the γ on the entire domain stored in previous step. This additional **KC** test makes the choice of an initial guess more flexible, since we have T candidates at most. Another benefit of moving the evaluations of β and η forward results from the reduction of estimation costs of Hessian in 11th line of Algorithm 4, since norm computation of H_is is costly and the registers available for the parallel threads are limited. Simultaneous evaluations of β and η with γ would affect the parallelism of GPUs. Alternatively, the storage and transfer of β, η and γ are trivial. Therefore, although this modification creates a little overhead, we believe it is worthwhile.

Algorithm 5 Additional **KC** test for multiple root. This test is called just after the 6th line of Algorithm 4. Note the difference between $\gamma_{\mathbb{P}}$ and γ in Algorithm 2.

1: **Input:** a set of systems $\{\mathbb{P}\}$ and their Hessian norms $\{\gamma_{\mathbb{P}}\}$;
2: **for** each system $\mathbb{P}$ **do**
3: **for** each subdomain that pass **CCS** test **do**
4: Calculate β, η, x_0, d as **KC** test ;
5: $\alpha = \gamma_{\mathbb{P}}\beta\eta$;
6: $r_1 = (\frac{1+\sqrt{1-2\alpha}}{\beta\gamma_{\mathbb{P}}})$;
7: **if** $(\alpha < \frac{1}{2})\&\&(r_1 \geq \frac{d}{2})$ **then**
8: Discard neighbors completely covered by $N(\mathbf{x}_0, r_1)$, **return** unique-root region $N(\mathbf{x}_0, r_1)$;
9: **end if**
10: **end for**
11: **end for**

4. Implementation and Discussion

We have implemented the proposed algorithm on a PC with an Intel Core 2 Quad 2.83GHz CPU and an NVIDIA GTX280 GPU. The parallel solver on GPU is written in CUDA v2.5, a general-purpose C language interface for general purpose computing. For state of art GPU, the single precision floating-point arithmetic is more efficient than the double one, while for CPU, their efficiencies are the same. Thus, we only implement a single precision floating-point solver on GPU. In our implementations, the sub-domain size tolerance is $1.0e-6$ for both CPU and GPU ones, which is small enough for root isolation. The convergent tolerances of the Newton-Raphson iteration are $1.0e-15$ and $1.0e-6$ for CPU and GPU, respectively.

The first example is to compute intersections of two bicubic planar algebraic curves in terms of Bernstein poly-

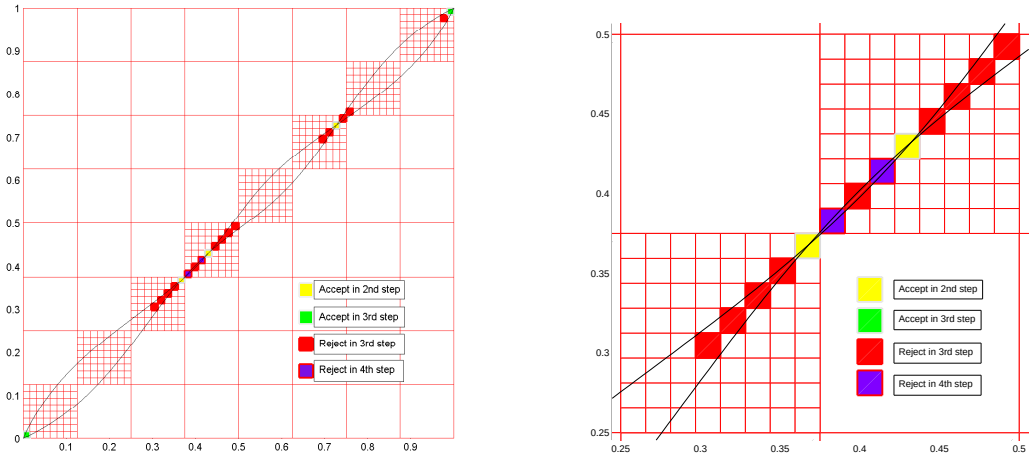

Figure 3: (a): the subdivision tree of root-finding of two implicit bi-cubics from [6]. (b) shows a room-in over the region near the center of (a).

nomials, which is presented in [6]. They are:

$$F_1(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} P_{ij}B_i(x)B_j(y) = 0$$

$$[P_{ij}] = \begin{pmatrix} 0 & 2.2 & 1.1 & 1.1 \\ -1 & -1 & -1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & -1 & -2 & 0 \end{pmatrix} \tag{4}$$

and $F_2(x,y) = F_1(y,x)$. There are five single roots in the unit square $[0,1] \times [0,1]$. As described in Section 3.3, a domain is subdivided into 64 sub-domains uniformly for 2D case. The first two subdivisions of unit square and its zoom-in are illustrated in Figure 3(a) and Figure 3(b), respectively. After the second subdivision, the yellow tiny square pass the **KC** test and their centers can be adopted as convergent initial guesses. The smaller squares in the green tiny squares are accepted as convergent regions in the third subdivision, while the red ones are rejected at the same time. The purple ones are rejected via **CCS** test after the fourth subdivision.

The system is solved on CPU and GPU in 2.63 ms and 0.47 ms with Algorithm 3 and 4, respectively. As a comparison, it takes the IRIT [4] 0.94 ms CPU time to solve it, in which the algorithm [6] has been integrated. The runtime difference between two CPU solvers can be explained as in Section 3.2. According to the experimental result, the **NC** test takes just about the same time as the **KC** test. Because the Kantorovich theorem provides a more strict conditions about domain and initial guess than those of the **NC** test, the proposed Algorithm 3 will takes about one more depth subdivision than the **NC** test approach. Thus, we can draw that the **NC** test approach is superior to our CPU implementation for the single root case.

However, for the example in Equation (4), we noticed that most of the GPU stream-processors are on idle due to the shortage of input, especially in the Hessian computations in the **KC** test, which is the most time consuming procedure. If the number of nonlinear systems increases, much more speed up via parallel processing can be gained. Figure 4 shows two synthetic examples of a large number of nonlinear systems. Figure 4(a) shows the intersections between level sets of two algebraic curves, and Figure 4(b) shows the intersections of two bi-cubic algebraic curves in term of B-spline basis, where the B-spline functions are converted into piecewise Bernstein polynomials via knot insertion algorithm. Both of the examples consist of a large number of nonlinear systems. Figure 5 shows the scalability and speedup of our parallel solver on GPU for a different number of nonlinear systems in Figure 4(b). $T2$ is the number of parallel threads working in batch in the 11th line of Algorithm 4. The speedup times are almost linear with regard to the number of working threads, until the peak value is arrived at $T2 = 32, 64$ for 512, 1024 nonlinear systems, respectively. After that, the speedup time remains unchanged since all of stream-processors work at their full capacity. It is the same for different numbers of subdivision T in 6th line in Algorithm 4. However, the tuning of number of parallel threads is subject to the computing architecture of CUDA. The multi-processor units create, schedule, and execute threads in groups of 32. Thus, we should set the number of threads per block as times of warp size. Another restriction in our parallel solver is available register number in one thread block since the register is limited resource in state of art GPU. We use the difference of F_i to construct Hessian function, which hardly increases the allocation of the register. As a result, for $T2 = T = 64$, our GPU allows one bi-14 degree or tri-quartic constraints for each thread at most.

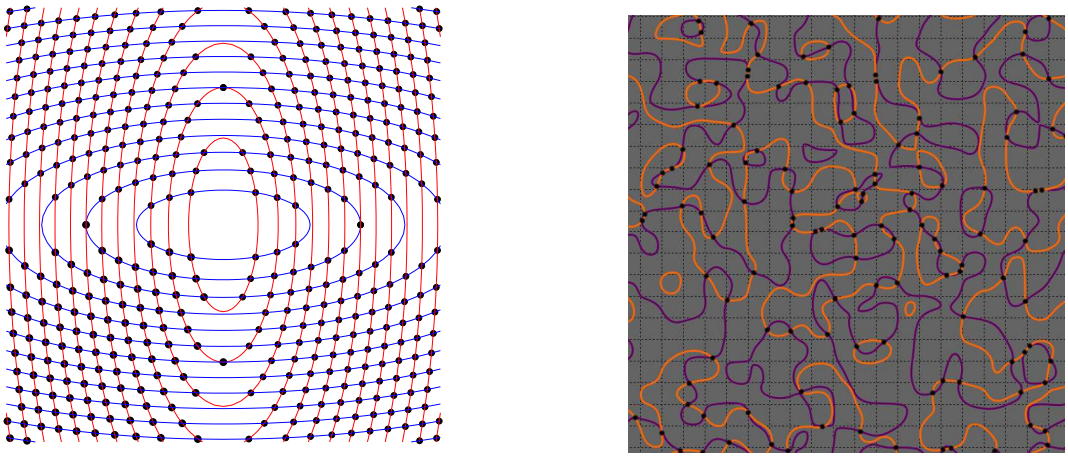

Figure 4: (a) shows the intersections of the iso-curve set of two algebraic curves. (b) shows the intersections of two bi-cubic algebraic curves in term of B-spline basis.

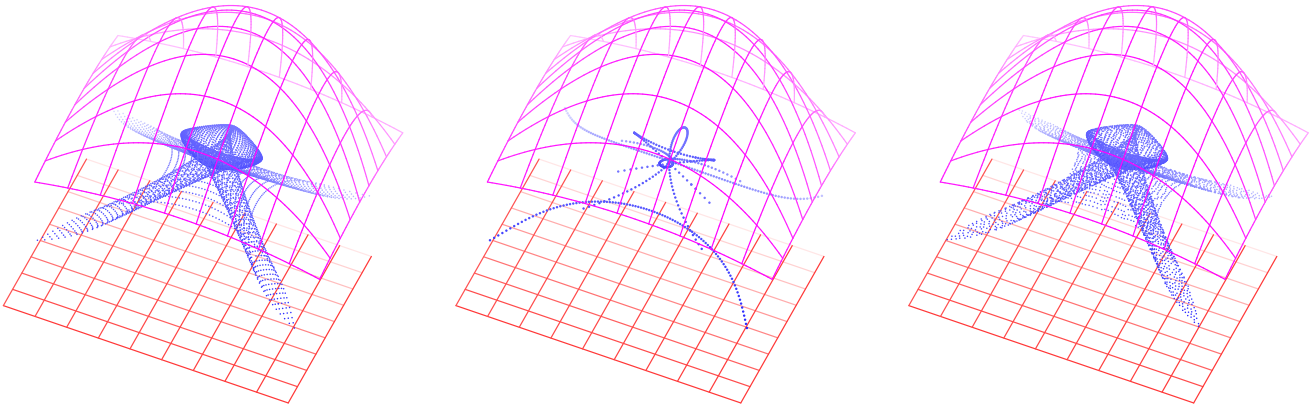

Figure 6: Bisector surface between an biquadratic surface $S1(u, v)$ and a planar $S2(s, t)$. It is a under-determined problem with 4 variables and solution space of 2-manifold. (a) and (b) enumerate u, v with resolution of 80×80 and 5×100, respectively. (c) is the result of IRIT under same subdivision resolution as (a). (a) takes 54.26 ms, while (c) takes 24859 ms.

The proposed algorithm is also suitable to solve under-determined systems. Before solving the system, the system is converted into a well-defined system by exhaustively sampling the additional variables according to given subdivision tolerance. Figure 6 shows the bisector surface [5] between a biquadratic surface $S1(u, v)$ and a planar $S2(s, t)$. Since the bisector surface between two surfaces can be reduced to an four variables (u, v, s, t) under-determined system of two constraints. The degrees of two constraints are $(7, 7, 3, 2)$ and $(7, 7, 2, 3)$, respectively. To solve the system efficiently, two variables of higher degree, *i.e.*, u and v, are sampled uniformly in their domains. Here the sampling solution is 80×80. In this way, the system is converted into 80×80 well-defined systems. Each system contains two constraints of two variables, whose degrees are $(3, 2)$ and $(2, 3)$, respectively. It takes our parallel solver total 54.26 ms GPU and CPU times to solve the well-defined systems, while it takes IRIT 24859 ms CPU times to solve it. The results are shown in Figures 6(a) and 6(c). Figure 6(b) shows another solution under sampling solution 5×100. It can draw that our parallel solver can gain much better speed-up than the solver in IRIT. We believe that by using the same scheme of variable reduction, IRIT could achieve better performance. Still, our algorithm provides a significant speedup. As we know, there are many under-determined systems in geometric modeling applications [5] and their solution space may be 1, 2 or 3-manifold. The proposed parallel solver can exploit the advantages of SIMD architecture of GPU or multicore CPU, which admits high throughput of lightweight computing.

Figure 7 show a double root case, where two planar algebraic curves, *i.e.* circles $g = 0$ and $f = 0$, are tangent at one point. There are three sub-domains, the hatched grid, which can pass the **CCS** test in first subdivision step. The center of middle sub-domain is more close to the double root than those of other two sub-domains. Against

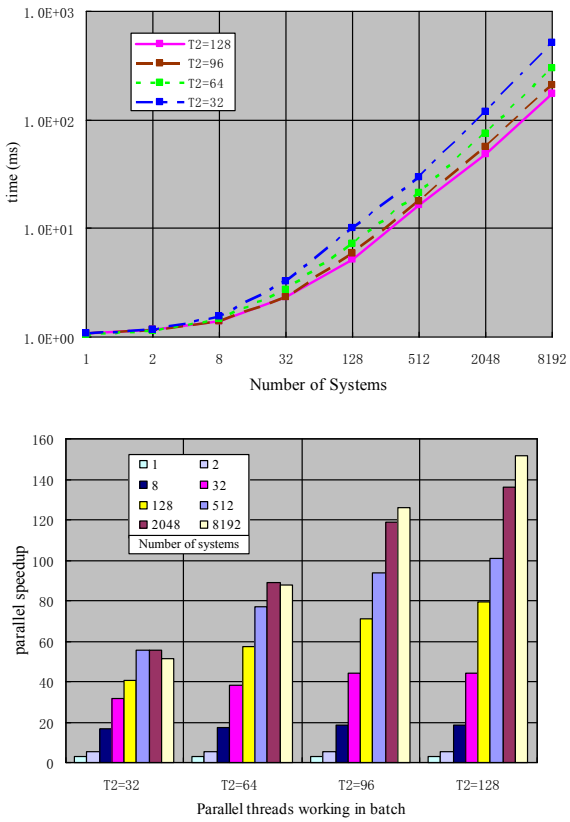

Figure 5: The run time and speedups comparison with IRIT for different numbers of nonlinear systems.

our expectations, the middle initial guess can not meet the convergent conditions of Kantorovich theorem, while the other ones can pass the **KC** test. The convergent region and unique root region are depicted as concentric circles. However, if we apply the Algorithm 4 to this example straightforwardly, the subsequent iterations will reject the top and bottom sub-domains. It will subdivide the middle one till the tolerance resolution is achieved. By using the revised scheme in Algorithm 5, the top and bottom sub-domains will be accepted and the middle one will be purged away, since the middle one lies in the unique root regions of other two sub-domains. In this example, our solver will terminates just after the additional **KC** test of Algorithm 5 in the first subdivision step, and obtain two convergent initial guesses.

5. Conclusion

In this paper, we propose a subdivision-based nonlinear Bernstein polynomial system parallel solver that is suitable for the SIMD architecture of state of art GPU. The solver is based on the Kantorovich theorem, which gives the conditions to determine a convergent initial guess for the Newton-Raphson iteration. By exploiting the parallelism of GPU, our algorithm can achieve over 100 times speedup for a large number of systems, compared with

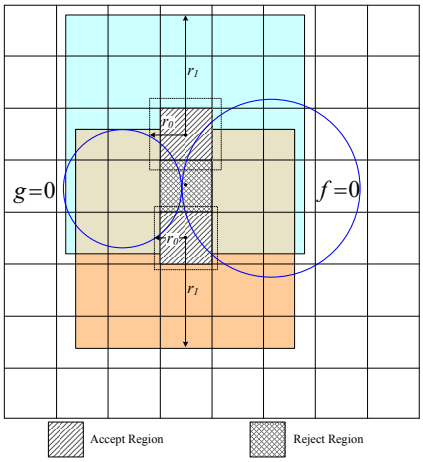

Figure 7: Local root distribution estimation of three initial guess. r_0,r_1 are same as in Figure 2. Note that the convergent region may not contain the unique root for using norms.

the CPU solver. The proposed solver can also dealt with under-determined system and multiple root case. This work can also be adapted to other SIMD architecture processes, such as multicore CPU.

Currently the proposed parallel solver is designed for GPU, which has no flexible memory management. Thus, the scalability is subject to hardware specification. Stream processors in the contemporary GPU are designed for processing single precision floating point arithmetic, while double precision floating point one is just added to address scientific and high-performance computing applications lately. The performance of double precision arithmetic is still much slower than the single ones. However, we believe that rapid development of GPU can overcome the restriction.

The Newton-Raphson iteration adopts Jacobian matrix to approximate the zeros. We have shown that the Kantorovich theorem can provide a convergent initial guess even for the singular Jacobian case, *i.e.* a tangent root case. However it is not hold for more complex multiple root case, as shown in Figure 8: a self-intersection curve intersects with a ellipse at the self-intersection point. The self-intersection point is a singular point, i.e., $\mathbf{F}$, $\mathbf{F}_x$ and $\mathbf{F}_y$ all vanish. We implemented the Newton-Raphson iterations by adopting initial guesses near it. The color maps of the convergent steps are shown in Figure 8, in which the red and brown colors indicate the convergent guesses with convergent iterations, while the blue color shows the divergence ones. We also notice that when the Newton-Raphson steps are more than 10, the iterations could go beyond the sub-domain, where the initial guesses belong. However, the condition of $\alpha < 0.5$ in Kantorovich theorem cannot be satisfied for all the initial guesses even for convergent ones. The example illustrates that the actually convergent region of initial guess for the Newton-Raphson iteration may be much greater than the one estimated by Kantorovich theorem. This may be the price that we pay

for conservative norm estimation in our algorithm.

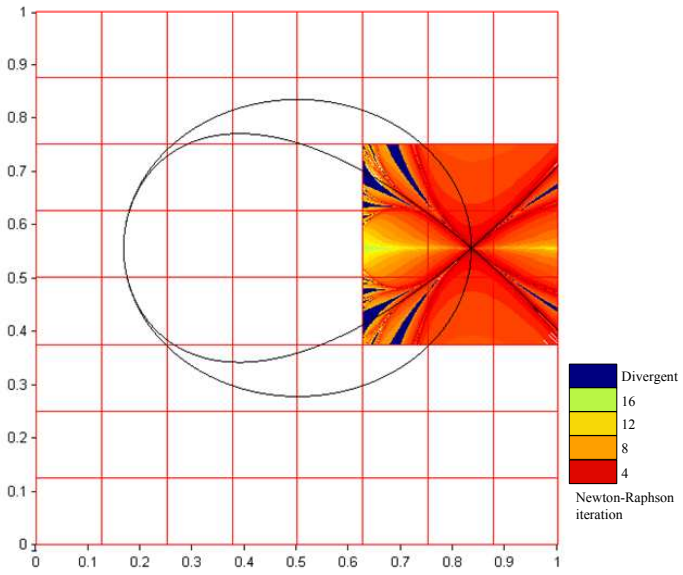

Figure 8: The points in colored girds can not satisfy the Kantorovich theorem. However, there are many convergent regions by adopting the Newton-Raphson method, except the blue-colored ones. The color maps show the number of the Newton-Raphson steps.

6. Acknowledgements

The authors are grateful to Gershon Elber who provided the open source of the IRIT. We also thank the anonymous referee for useful comments and improvements.

References

[1] G. Barequet, G. Elber, Optimal bounding cones of vectors in three dimensions, Information Processing Letters 93 (2005) 83 – 89.

[2] G.E. Collins, R. Loos, Real zeroes of polynomials, Computer algebra: symbolic and algebraic computation (2nd ed.) (1983) 83–94.

[3] J. Dennis, R.B. Schnabel, Numerical methods for unconstrained optimization and nonlinear equations, Prentice-Hall Englewood Cliffs, N.J., 1983.

[4] G. Elber, Irit homepage (2010). `http://www.cs.technion.ac.il/~irit/`.

[5] G. Elber, M.S. Kim, Geometric constraint solver using multivariate rational spline functions, in: SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications, ACM, New York, NY, USA, 2001, pp. 1–10.

[6] I. Hanniel, G. Elber, Subdivision termination criteria in subdivision multivariate solvers using dual hyperplanes representations, Comput. Aided Des. 39 (2007) 369–378.

[7] M. Harris, J. Owens, S. Sengupta, Y. Zhang, A. Davidson, Cudpp homepage (2007). `http://www.gpgpu.org/developer/cudpp/`.

[8] M. Harris, S. Sengupta, J.D. Owens, Parallel prefix sum (scan) with cuda, in: H. Nguyen (Ed.), GPU Gems 3, Addison Wesley, 2007.

[9] J.M. Lane, R.F. Riesenfeld, Bounds on a polynomial, BIT Numerical Mathematics 21 (1981) 112–117.

[10] L.H. Lim, Singular values and eigenvalues of tensors: A variational approach, PROCEEDINGS OF THE IEEE INTERNATIONAL WORKSHOP ON COMPUTATIONAL 1 (2005) 129.

[11] C. Loop, J. Blinn, Real-time gpu rendering of piecewise algebraic surfaces, in: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, ACM, New York, NY, USA, 2006, pp. 664–670.

[12] D. Manocha, J. Demmel, Algorithms for intersecting parametric and algebraic curves i: simple intersections, ACM Trans. Graph. 13 (1994) 73–100.

[13] K. Mehlhorn, M. Sagraloff, Isolating real roots of real polynomials., in: J. Johnson, H. Park, E. Kaltofen (Eds.), ISSAC, ACM, 2009, pp. 247–254.

[14] R.E. Moore, F. Bierbaum, Methods and Applications of Interval Analysis (SIAM Studies in Applied and Numerical Mathematics) (Siam Studies in Applied Mathematics, 2.), Soc for Industrial & Applied Math, 1979.

[15] B. Mourrain, J.P. Pavone, Subdivision methods for solving polynomial equations, J. Symb. Comput. 44 (2009) 292–306.

[16] T. Nishita, T.W. Sederberg, M. Kakimoto, Ray tracing trimmed rational surface patches, in: SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, 1990, pp. 337–345.

[17] N.M. Patrikalakis, T. Maekawa, Shape Interrogation for Computer Aided Design and Manufacturing, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[18] L. Qi, Eigenvalues and invariants of tensors, Journal of Mathematical Analysis and Applications 325 (2007) 1363 – 1377.

[19] L. Ramshaw, Blossoms are polar forms, Comput. Aided Geom. Des. 6 (1989) 323–358.

[20] T.W. Sederberg, R.J. Meyers, Loop detection in surface patch intersections, Comput. Aided Geom. Des. 5 (1988) 161–171.

[21] E.C. Sherbrooke, N.M. Patrikalakis, Computation of the solutions of nonlinear polynomial systems, Computer Aided Geometric Design 10 (1993) 379 – 405.