Doctoral theses at NTNU, 2012:99

Øystein Tråsdahl

High order methods for partial differential equations: geometry representation and coordinate transformations

Doctoral Thesis

Øystein Tråsdahl

Doctoral theses at NTNU, 2012:99

NTNU
Norwegian University of Science and Technology
Thesis for the degree of Philosophiae Doctor
Faculty of Information Technology
Mathematics and Electrical Engineering
Department of Mathematical Sciences

NTNU

ISBN 978-82-471-3475-7 (printed version)
ISBN 978-82-471-3476-4 (electronic version)
ISSN 1503-8181

Øystein Tråsdahl

High order methods for partial differential equations: geometry representation and coordinate transformations

Thesis for the degree of Philosophiae Doctor

Trondheim, April 2012

Norwegian University of Science and Technology
Faculty of Information Technology Mathematics
and Electrical Engineering
Department of Mathematical Sciences

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology
Mathematics and Electrical Engineering
Department of Mathematical Sciences

ISBN 978-82-471-3475-7 (printed version)
ISBN 978-82-471-3476-4 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2012:99

MILJØMERKET
241 TRYKKSAK 731

Printed by Skipnes Kommunikasjon as

PREFACE

This PhD thesis concludes my doctoral studies in numerical analysis at the Department of Mathematical Sciences at the Norwegian University of Science and Technology (NTNU).

First and foremost I want to thank my thesis advisor, Prof. Einar M. Rønquist, for all the time and effort he has put into this project. His insight has been invaluable and his enthusiasm a great encouragement and motivation for me. I would also like to thank his family for having me over in Boston in January this year during the hectic, but delightful week when this thesis was completed.

I am also grateful to Dr. Tormod Bjøntegaard for extensive collaboration, fruitful discussions, and co-authorship on some of the papers included in this thesis.

Thanks also to Dr. Jens L. Eftang, with whom I have shared my office for the majority of this time. Thanks for good advice, interesting discussions and help with the layout for this thesis.

I wish to thank my friends at NTNU for moral support and many a pleasant coffee break. Finally, a warm thanks to my family and Rike for always being there with support and encouragement.

Øystein Tråsdahl
Trondheim, March 2012

CONTENTS

Introduction 1

Paper 1
Spectral approximation of partial differential equations in highly distorted domains 31

Paper 2
High order polynomial interpolation of parameterized curves 55

Paper 3
High order interpolation of parametric curves and surfaces in $\mathbb{R}^3$ 67

Paper 4
High order numerical approximation of minimal surfaces 105

Paper 5
Adaptive spectral methods 135

INTRODUCTION

Solving partial differential equations (PDEs) using high order numerical methods is very attractive for problems where the solution and the given data exhibit a high degree of regularity; this is due to the rapid convergence rate that can be achieved for such problems. Such methods are also invaluable for problems where the solution is very sensitive to discretization errors, for example, when solving hydrodynamic stability problems. High order methods for PDEs have enjoyed a significant progress over the past few decades, starting with pure spectral methods in simple domains based on truncated Fourier series. The extension of these first methods to problems with non-periodic boundary conditions was later achieved by considering high order polynomial approximations [19, 8].

The combination of domain decomposition and spectral methods subsequently resulted in the development of the spectral element method [37, 32]. This method combines (some of) the geometric flexibility of low order finite element methods with the good convergence properties of spectral methods. The original spectral element method decomposes the global domain into quadrilateral or hexahedral elements, with each element considered as the image of a corresponding tensor-product reference domain (e.g., a d-dimensional cube in $\mathbb{R}^d$). The polynomial approximation is constructed using a tensor-product nodal representation based on the tensor-product Gauss-Lobatto Legendre points on the reference domain. In more than one space dimension, the multi-dimensional basis functions are all tensor-product extensions of the one-dimensional case, and only one-dimensional differentiation matrices are needed. Using the weak form of the PDE as a point of departure for the discretization, the associated integrals are also evaluated using the same GLL points. Convergence can be achieved by only increasing the polynomial approximation within each element, keeping the number of elements fixed; this will result in a convergence rate which depends on the regularity of the problem, with the possibility of achieving exponential convergence for analytic solution and data. However, it is also possible to achieve convergence by both increasing the polynomial degree within each element as well as increasing the number of elements, which is similar in philosophy to *hp* finite element methods [2, 41].

Over the past couple of decades the spectral element methodology has been extended to triangular and tetrahedral elements [29]. The motivation for this is to be able to consider even more complex domains and also to be able to use automatic mesh generators. Again, each individual element is constructed as the image of a reference triangle or reference tetrahedron. However, we remark that the approximation space on the d-dimensional cube is different from the approximation space on a simplex. On the cube, the polynomial approximation in each spatial direction is N, while on a triangle or on a tetrahedron, N denotes the total polynomial degree.

While the initial spectral element methods on simplices used modal basis functions, there has been a significant development over the last decade on constructing nodal basis functions. This has spurred interest in finding good interpolation points on simplices [20, 42, 36]. On triangles, the Fekete points have been considered close to optimal in the sense of minimizing the Lebesgue constant associated with polynomial interpolation. Another attractive feature with the Fekete points is that they reduce to the GLL points in the one-dimensional case, e.g., along an edge of a triangle. On tetrahedrons it is not clear what the best nodal points are. A complicating factor with simplices is that a single set of points may not give both good interpolation properties and good quadrature properties as the GLL points do in the tensor-product case.

When considering a computational domain with a piecewise smooth boundary, there is a common ingredient in all the spectral element methods discussed above: each element in physical space is considered as the image of a corresponding reference element (either a cube or a simplex). Independent of the choice of nodal points on the reference element, there is a need to construct an appropriate mapping between the reference element and each physical element. A common approach is to use an isoparametric approximation, which means that the geometry is represented using the same approximation space as for the other pertinent field variables. A key aspect of this construction is the quality of the polynomial approximation of the domain boundary, e.g., the approximation of each edge of a quadrilateral element or each edge of a triangular element. However, there is very limited discussion in the literature of how to best do this, and this topic will play a central role in this thesis. While the topic of interpolation of curves and surfaces has been given very little attention in the high order community, it is a central part of Computer Aided Geometric Design

(CAGD). Some of the work in this thesis will therefore try to exploit already existing results and adapt and extend these to the high order numerical solution of PDEs. In the two-dimensional case, high order polynomial approximation of edges is required, either edges of quadrilateral elements or the edges of triangles. One way to approximate a single edge is to view this as a function relative to an appropriately oriented coordinate system, and to apply classical interpolation. However, an edge can also be viewed as a parametric curve, with the possibility of reparametrization in order to more easily approximate the curve through polynomial approximation; this will also be a topic of this thesis work.

Once a good approximation of the boundary has been constructed, this approximation can be extended to the interior of the element (or global domain) through a chosen mapping. This mapping is often chosen as simple as possible, e.g., an affine mapping. However, there is an inherent freedom in high order methods to choose the mapping differently, and a wise choice can potentially give better representation of the various field variables (e.g., the solution and the geometry). This freedom is very often not exploited, but will be considered in this thesis. In particular, this freedom can give rise to adaptive methods. Methods for achieving adaptivity in the context of spectral methods have been proposed earlier [3, 35, 43], but the methods either lack robustness or generality or both.

Methods for adaptive mesh construction represent valuable and essential tools in the solution of moving boundary problems, i.e., time-dependent PDEs where the geometry itself changes with time and hence is part of the solution. An important example of such problems is free surface flows. A simpler example is mean curvature flow, in which case we do not have to solve the full governing fluid flow equations for each time step. Nonetheless, such problems give us the same challenge in terms of geometry representation: we have a computational domain which evolves and several degrees-of-freedom in how we let the computational mesh evolve with it. In particular, tangential movement of mesh points along the surface is allowed and may improve both the representation of the geometry and the numerical solution of other dependent field variables. However, mesh update algorithms have seen little development, and many authors resort to *ad hoc* methods such as imposing homogeneous Neumann or Dirichlet boundary conditions on the mesh movement [23, 6, 5, 25, 4]. Again, better control of the evolution and representation of the domain boundary will (implicitly) result in the construction of better mappings between the reference domain and each physical element.

This chapter is organized as follows: in Section 1, we review some standard results from classical interpolation of functions in one space dimension, and we discuss how interpolation of parametric curves in $\mathbb{R}^2$ can play an important part of this topic; this is illustrated by including a numerical example. We end Section 1 by discussing interpolation of curves and surfaces in $\mathbb{R}^3$. In Section 2, we discuss adaptive spectral methods in $\mathbb{R}^1$, while in Section 3, we discuss spectral methods for PDEs in $\mathbb{R}^2$ with particular focus on geometry representation in deformed quadrilateral domains. Finally, in Section 4, we discuss the role of geometry representation in evolving domains, with particular focus on high order methods and solving minimal surface problems.

1 High order interpolation

Consider polynomial interpolation of a one-dimensional function $u(x)$, $x \in \Omega = [a, b]$. With $N+1$ interpolation points we can construct a polynomial interpolant $I_N u(x)$ of degree N. The interpolation points can be chosen to be some type of Gauss points, e.g., the Gauss-Lobatto Legendre (GLL) points which will include the end points of the interval. The Gauss points are attractive to use since they give rapid convergence of the interpolation error $||u - I_N u||$ as the polynomial degree, N, increases, and the convergence rate increases with the regularity of u. The interpolation points are associated with the zeros of some orthogonal polynomials and are defined on the interval $\widehat{\Omega} = [-1, 1]$. If the interval is $[a, b]$, the simple linear coordinate transformation $x = \mathcal{F}(\xi)$, with

$$\mathcal{F}(\xi) = a + \frac{(b-a)}{2}(\xi + 1), \tag{1.1}$$

will transform $u(x)$ to $u(x) = u(\mathcal{F}(\xi)) = (u \circ \mathcal{F})(\xi) = \hat{u}(\xi)$, and standard GLL interpolation can be applied to the function $\hat{u}(\xi)$ to give the interpolant $I_N \hat{u}(\xi) \in \mathbb{P}_N(\widehat{\Omega})$, such that

$$I_N \hat{u}(\xi_i) = u(x_i), \qquad i = 0, \ldots, N, \tag{1.2}$$

where ξ_i is a GLL point and $x_i = \mathcal{F}(\xi_i)$. Here, $\mathbb{P}_N(\widehat{\Omega})$ is the space of polynomials of degree less than or equal to N defined over $\widehat{\Omega}$. Once we have constructed $I_N \hat{u}$, $I_N u$ is readily given as $I_N u = (I_N \hat{u}) \circ \mathcal{F}^{-1}$.

The quality of a set of interpolation points Ξ in $\widehat{\Omega}$ is usually measured through the Lebesgue constant $\Lambda_N(\Xi)$, for functions $\hat{u}$ defined over $\widehat{\Omega}$ [36],

$$||\hat{u} - I_N \hat{u}||_\infty \leq \big(\Lambda_N(\Xi) + 1\big)||\hat{u} - \hat{u}_N||_\infty, \tag{1.3}$$

where $\hat{u}_N$ is the best approximation of $\hat{u}$ by polynomials of degree less than or equal to N. A good set of interpolation points is therefore a set which has a small Lebesgue constant, something which is true for the GLL points. The norm in the above inequality is the $L^\infty(\widehat{\Omega})$ or the maximum norm.

If we assume that $u \in H^\sigma(\Omega)$, the interpolation error can be bounded as [8]

$$||u - I_N u||_{L^2(\Omega)} \leq cN^{-\sigma}||u||_{H^\sigma(\Omega)}. \tag{1.4}$$

In particular, if u is analytic ($\sigma \to \infty$), the error decays exponentially fast as N increases. A similar error estimate can be derived for the best approximation $u_N(x)$, where

$$||u - u_N||_{L^2(\Omega)} = \min_{v \in \mathbb{P}_N(\Omega)} ||u - v||_{L^2(\Omega)}. \tag{1.5}$$

We now make a few comments regarding these error estimates. First, they are generally derived for functions defined on $\widehat{\Omega}$. However, the simple linear coordinate transformation (1.1) implies that $I_N u(x)$ and $u_N(x)$ are also polynomials of degree N, and the error estimates also applies for the functions defined in the original coordinate system.

An interesting question now is how we potentially could use a different (non-affine) coordinate transformation (or mapping) $\mathcal{F} : \widehat{\Omega} \to \Omega$ to produce a different $\hat{u} = u \circ \mathcal{F}$ that (i) would be more suitable to polynomial interpolation in the GLL points; and (ii) produce a resulting interpolant $I_N u$ in the original coordinate system with a smaller interpolation error than the interpolant generated through the standard (affine) mapping. Such an interpolant $I_N u(x)$ would generally no longer be a polynomial in x, and the interpolation points $x_i = \mathcal{F}(\xi)$, $i = 0, \ldots, N$, would no longer correspond to a GLL distribution.

1.1 Interpolation of parametric curves in $\mathbb{R}^2$

Another way to look at the problem of interpolating a one-dimensional function $u(x)$ is by looking at the given function as a parametric curve in $\mathbb{R}^2$. A simple parametrization is the curve

$$(x, u(x)), \qquad x \in [a, b]. \tag{1.6}$$

However, we can also look at the reparametrized curve

$$(\mathcal{F}(\xi), \hat{u}(\xi)), \qquad \xi \in [-1, 1], \tag{1.7}$$

where $x = \mathcal{F}(\xi)$ and $\hat{u}(\xi) = u(\mathcal{F}(\xi))$. Of course, as ξ varies from -1 to 1, and x varies from a to b, these two parametrizations describe the same curve (or geometric object). However, for a nonlinear $\mathcal{F}$, the "speed" at which the curve is traversed is different from the linear case.

In general, a function $u(x)$ can thus be described as a vector-valued function

$$\boldsymbol{f}(\xi) = (x(\xi), y(\xi)), \qquad \xi \in [-1, 1], \tag{1.8}$$

which again can be reparametrized through a change of variable. Reparametrization represents an opportunity to improve the approximation properties of the interpolant. A natural question is then how to find the best reparametrization in the sense that the interpolation error, measured in a suitable norm, is minimized.

Interpolation of functions is an old and well-explored topic. Interpolation of parametric curves, on the other hand, is a field where research is still active, and where there are still unanswered questions. In one way, one can consider interpolation of a parametric curve in $\mathbb{R}^d$ as interpolation of a vector-valued function, and traditional methods can be applied. For example, one can interpolate each parametric function in the Gauss points. However, the particular feature that separates parametric curves from vector-valued functions is *reparametrization*: one can choose to reparametrize the given parametric curve before applying the interpolation procedure. In effect, we interpolate a different vector-valued function that describes the same geometric object. Reparametrization can be thought of as traversing the same curve at a different speed, and interpolating a reparametrized curve can be thought of as moving the original interpolation points along the exact curve. For example, one particular parametrization is the *arc length parametrization*, which traverses the curve with constant velocity (i.e., the Jacobian $J(\xi) = \sqrt{x_\xi^2 + y_\xi^2}$ is constant in ξ).

Let us now make a few additional remarks about parametric curves. A parametric curve is C^k*-continuous* if each parametric function ($x(\xi)$ or $y(\xi)$ in (1.8)) is a C^k function. Moreover, the curve is G^k*-continuous* if its arc length parametrization consists of only C^k functions. The concept of G^k-continuity is independent of the given parametrization. In $\mathbb{R}^2$, G^1 continuity means that the curve has no break points (i.e., a continuous tangent), and G^2 continuity means it has a continuous curvature.

Consider interpolation of the curve (1.8) in $\mathbb{R}^2$ in the Gauss-Lobatto Legendre (GLL) points. To this end we define the curve interpolation operator $\boldsymbol{I}_N$ by

$$\boldsymbol{I}_N \boldsymbol{f}(\xi) = (I_N x(\xi), I_N y(\xi)), \tag{1.9}$$

where I_N means interpolation of a function in the GLL points. Hence, the interpolant is a parametric curve, and each parametric function is a polynomial in $\mathbb{P}_N([-1,1])$. The polynomials can be represented by the nodal bases

$$\begin{aligned} I_N x(\xi) &= \sum_{i=0}^{N} x_i \ell_i(\xi), \\ I_N y(\xi) &= \sum_{i=0}^{N} y_i \ell_i(\xi), \end{aligned} \tag{1.10}$$

where ℓ_i is the i'th Lagrange interpolant through the $N+1$ GLL points $\xi_0, \dots, \xi_N$, such that $\ell_i(\xi_j) = \delta_{ij}$. This way interpolation is achieved simply by sampling the given curve in the points (x_i, y_i), $i = 0, \dots, N$. The polynomial interpolant is uniquely determined by the $2(N+1)$ coefficients x_i and y_i in (1.10). In the current context, only $(N-1)$ of these are free variables: once $x_0, \dots, x_N$ are determined, $y_0, \dots, y_N$ are determined by the requirement that $y_i = u(x_i)$. Also, since we use Lobatto type interpolation points it is natural to let $x_0 = a$ and $x_N = b$, i.e., we require that the end points be interpolation points. Hence, interpolation of parametric curves is a problem with $N-1$ *degrees-of-freedom.* Note how this differs from classical interpolation of functions: this freedom exists *after* we have chosen the GLL points to be the interpolation points in the reference variable ξ.

We now make some remarks on the resulting interpolation error. In the case of parametric curve interpolation, the interpolation error can get contributions from both parametric functions, so we somehow need to "balance" the regularity between the two parametric functions. This is what reparametrization is about: moving the variation and complexity from one parametric function to the other. However, it is not obvious how we can exploit this freedom to minimize the interpolation error in a controlled and automatic way.

The issue of curve interpolation has been given very little attention in the high order community. However, it is a central part of Computer Aided Geometric Design (CAGD) where it has been studied extensively [12, 34]. In [10] it was shown that under certain conditions, cubic polynomial curves in $\mathbb{R}^2$ can interpolate both function values, tangent directions *and* curvature at the end points, resulting in approximation order six. This is in contrast to classical interpolation of functions, which gives fourth order approximation. The price to pay for the increased accuracy is a system of non-linear equations that must be solved. The interpolation method was viewed as a generalization of Hermite interpolation based on geometric quantities and was therefore called *geometric Hermite interpolation.* In recent years we have seen a lot of work on geometric Hermite interpolation in the CAGD community; e.g., see [11, 15, 16, 28, 39, 40, 46]. A lot of interpolation methods have been proposed, and almost all are based on the same principle: using the degrees-of-freedom that are available to increase the number of interpolation points or to increase the number of derivatives matched at each interpolation point. Based on the number of degrees-of-freedom, the maximum attainable number of interpolation points for curves in $\mathbb{R}^2$ has been *conjectured* [24] to be $2N$, i.e., $N-1$ more interpolation points than classical interpolation of a function. The conjecture remains unproven, and no general interpolation method to achieve $2N$ interpolation points for general N have been proposed in the literature.

1.2 Numerical example: Interpolating the Runge function

We consider now the function famous for illustrating the Runge phenomenon [17],

$$u(x) = \frac{1}{1+16x^2}, \qquad x \in [-1,1]. \tag{1.11}$$

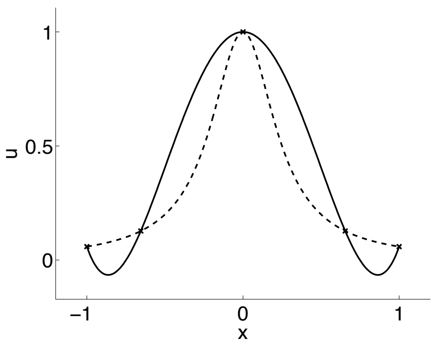
A standard high order interpolation at the GLL points yields a polynomial $I_N u(x)$, and a slight oscillatory behavior can be observed in Figure 1. Compared with other types of polynomial interpolations, the GLL points are considered to be close to optimal. For example, equidistant interpolation points will give such wild oscillations that the interpolants do not even converge when N increases.

Note that a standard interpolation method makes the approximation $I_N u$ a polynomial in x. In contrast, for the equal-tangent method discussed in Paper 1, we require the individual approximations $x_N = I_N x$ and $y_N = I_N y$ to be polynomials in the reference variable ξ, while $y_N(x_N)$ will not, in general, be a polynomial. This makes the equal-tangent method completely different from classical interpolation, and there is a potential of avoiding the well-known oscillations associated with classical interpolation.

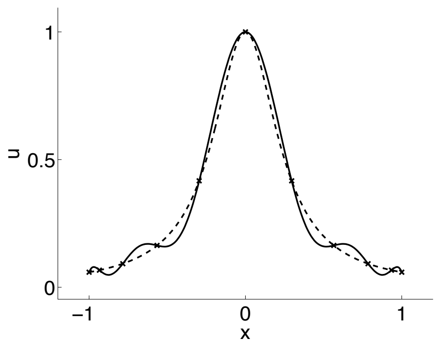
Figure 2 shows the interpolation error in the L^2-norm. The standard method gives exponential convergence, but with very low convergence rate. The equal-tangent method converges fast from the very beginning, reaching machine precision already at $N \approx 15$.

1.3 Interpolation of curves and surfaces in R^3

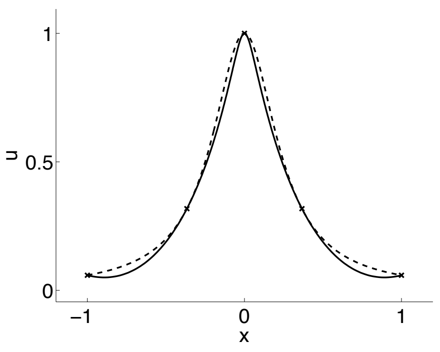
The results and conclusions related to the interpolation of parametric curves in $\mathbb{R}^2$ can be extended to parametric curves $(x(\xi), y(\xi), z(\xi))$ in $\mathbb{R}^3$; see Figure 3. However, note that a polynomial interpolant in $\mathbb{R}^3$ does not involve more degrees-of-freedom than in $\mathbb{R}^2$.



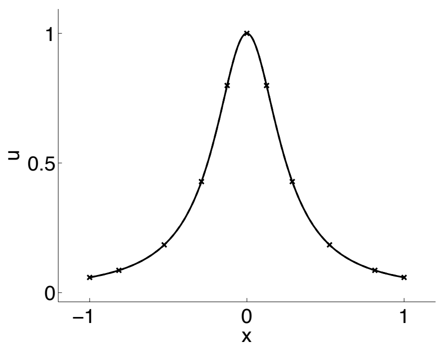
(a) $N = 4$



(b) $N = 10$



(c) $N = 4$



(d) $N = 10$

Figure 1: The exact curve (dashed) and the interpolant for $N = 4$ and $N = 10$, using the standard method (top) and the equal-tangent method (bottom) discussed in Paper 1. We see that the standard method results in oscillations which decrease as N increases, while no oscillations are observed with the equal-tangent method.

There are many similarities between interpolation of parametric curves and interpolation of parametric surfaces. In particular, there is a certain number of degrees-of-freedom associated with a reparametrization, and these can be used to improve the interpolant.

A parametric surface $\boldsymbol{f}$ in $\mathbb{R}^3$ can be described in a Cartesian coordinate

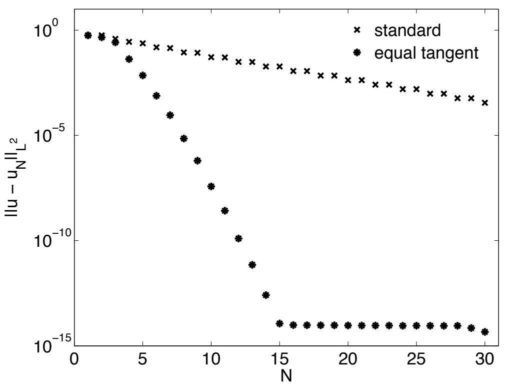


Figure 2: Interpolation error measured in the discrete L^2-norm. The equal-tangent method gives a much faster convergence than standard interpolation in the GLL points.

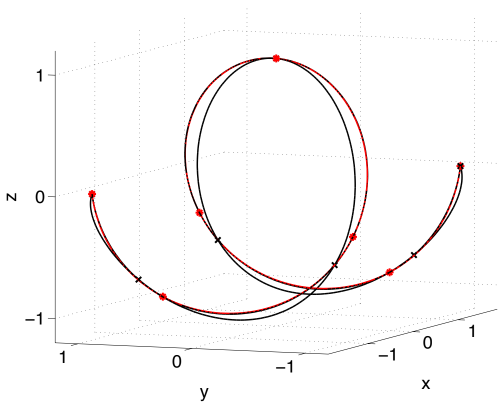


Figure 3: Interpolation of a space curve that is one and a half turn of a helix. Interpolating the natural helix parametrization yields a solution ($\times$) that is clearly distinguished from the exact curve. On the other hand, the equal-tangent method described in Paper 3 gives an interpolant ($*$) that cannot be distinguished from the exact curve.

system by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_1(\eta_1,\eta_2) \\ f_2(\eta_1,\eta_2) \\ f_3(\eta_1,\eta_2) \end{bmatrix} = \boldsymbol{f}(\eta_1,\eta_2), \qquad \eta_1,\eta_2 \in [-1,1]. \tag{1.12}$$

Hence, each of the parametric functions f_i, $i = 1,2,3$, are defined over a reference domain $\widehat{\Omega} = [-1,1] \times [-1,1] \subset \mathbb{R}^2$. A change of variable, realized by a bijective mapping $\varphi : \widehat{\Omega} \to \widehat{\Omega}$ yields a reparametrization

$$\boldsymbol{g}(\xi_1,\xi_2) = \boldsymbol{f}(\varphi(\xi_1,\xi_2)) = \boldsymbol{f}(\eta_1,\eta_2). \tag{1.13}$$

Note that ξ_1 and ξ_2 here represent two independent variables and not two GLL points.

The interpolant is a parametric surface $\boldsymbol{I}_N\boldsymbol{g}$ described by

$$\boldsymbol{I}_N\,\boldsymbol{g}(\xi_1,\xi_2) = \begin{bmatrix} I_N g_1(\xi_1,\xi_2) \\ I_N g_2(\xi_1,\xi_2) \\ I_N g_3(\xi_1,\xi_2) \end{bmatrix}, \qquad \xi_1,\xi_2 \in [-1,1], \tag{1.14}$$

where each component $I_N g_i$, $i = 1,2,3$, is a polynomial of degree less than or equal to N in the two independent variables in $\widehat{\Omega}$. Each component is conveniently represented by sums of tensor-product Lagrangian interpolants in the tensor-product GLL points, i.e.,

$$I_N g_i(\xi_1,\xi_2) = \sum_{m=0}^{N}\sum_{n=0}^{N} (g_i)_{mn}\,\ell_m(\xi_1)\ell_n(\xi_2), \qquad i = 1,2,3. \tag{1.15}$$

The rectilinear mesh that is made up by the interpolation points in $\widehat{\Omega}$ is mapped to a curvilinear mesh on Ω; see Figure 4.

The basis coefficients $(g_i)_{mn}$ are uniquely determined by the interpolant. Some of these are determined by the requirement that the nodes on the boundary of the tensor-product GLL mesh are mapped to the boundary $\partial\Omega$ of the exact surface. This gives $2(N-1)^2$ interior degrees-of-freedom plus $4(N-1)$ edge degrees-of-freedom, i. e., we are left with a total of $2N^2 - 2$ degrees-of-freedom in the parametric surface interpolation problem. This is to be compared with the $N-1$ degrees-of-freedom for the interpolation of parametric curves (e.g, in $\mathbb{R}^2$ or in $\mathbb{R}^3$).

The concept of geometric Hermite interpolation can also be applied in the context of parametric surfaces, but the problem is much harder due to the increased number of unknowns. Mørken [33] gives a detailed discussion of the optimal approximation order and constructs a quadratic Taylor approximant with

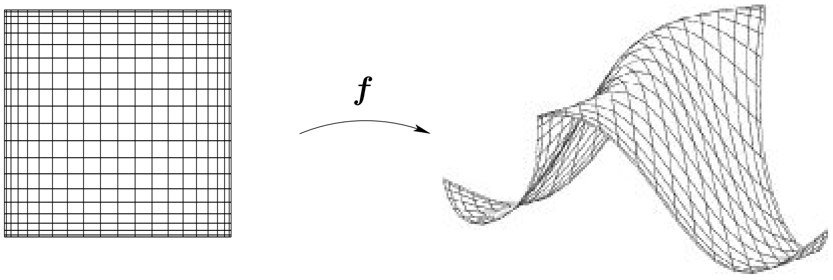


Figure 4: The surface is mapped from a reference domain $\widehat{\Omega} = [-1, 1] \times [-1, 1]$ by the parametrization $\boldsymbol{f}$ (which can be reparametrized). The physical interpolation points are the images of the tensor-product GLL points.

approximation order four. Lagrange interpolation of surfaces with quadratic polynomials is considered in [27]. Paper 3 discusses the problem of interpolation of curves and surfaces in $\mathbb{R}^3$ in more detail.

2 Adaptive spectral methods in R^1

Adaptivity is well developed in the context of low order finite element methods [1, 41, 44]. However, this is not the case in the context of spectral methods [7]. As a motivation for looking closer at this problem, consider the one-dimensional advection-diffusion equation

$$-\varepsilon u_{xx} + u_x = f, \qquad x \in \Omega, \tag{2.1}$$

accompanied by suitable boundary conditions. Here, ε is a (small) constant, f is a smooth, given function, and Ω is a bounded interval on the real axis. To be specific, we may consider the particular case with $\Omega = (0, 1)$, $\varepsilon = 0.01$, $f(x) = 1$, and homogeneous Dirichlet boundary conditions. This problem has the exact solution

$$u(x) = x - \frac{e^{x/\varepsilon} - 1}{e^{1/\varepsilon} - 1}, \tag{2.2}$$

which features a boundary layer with a width of order $\mathcal{O}(\varepsilon)$ near $x = 1$.

Given the exact solution $u(x)$ to this differential equation, we can construct the classical high order interpolant, or we can construct the parametric interpolant discussed earlier. The conclusion is that, for boundary-layer solutions like (2.2), a smooth, non-affine mapping $\mathcal{F}$ which moves the physical interpolation points towards the boundary layer will give a smaller interpolation error than the standard one.

In order to more clearly see how we may formulate an adaptive problem here, consider the weak formulation of the advection-diffusion problem: find $u \in X = H_0^1(\Omega)$ such that

$$a(u, v) = \ell(v), \qquad \forall v \in X, \tag{2.3}$$

where the bilinear form is given as

$$a(u, v) = \varepsilon \int_\Omega u_x v_x \, \mathrm{d}x + \int_\Omega u_x v \, \mathrm{d}x, \tag{2.4}$$

and the linear form is given as

$$\ell(v) = \int_\Omega f v \; \mathrm{d}x. \tag{2.5}$$

For the spectral discretization we first define the mapping $\mathcal{F} : \widehat{\Omega} \to \Omega$ and then define the discrete space

$$X_N = \{v \in X, v \circ \mathcal{F} \in \mathbb{P}_N(\widehat{\Omega})\}, \tag{2.6}$$

i.e., comprising functions that are polynomials over $\widehat{\Omega}$. We also let

$$J = J(\xi) = \frac{\mathrm{d}\mathcal{F}}{\mathrm{d}\xi}(\xi) \tag{2.7}$$

denote the Jacobian associated with the coordinate transformation $x = \mathcal{F}(\xi)$. The discrete problem then reads: find $u_N \in X_N$ such that

$$a(u_N, v) = \ell(v), \qquad \forall v \in X_N. \tag{2.8}$$

Over $\widehat{\Omega}$ (i.e, in the reference variable ξ) the bilinear and linear forms read

$$a(u_N, v) = \varepsilon \int_{\widehat{\Omega}} \frac{1}{J} \hat{u}_\xi \hat{v}_\xi \, \mathrm{d}\xi + \int_{\widehat{\Omega}} \hat{u}_\xi \hat{v} \, \mathrm{d}\xi, \tag{2.9}$$

and

$$\ell(v) = \int_{\widehat{\Omega}} \hat{f} \hat{v} \, J \; \mathrm{d}\xi. \tag{2.10}$$

In practice, the bilinear and linear forms are evaluated using GLL quadrature. Since the Jacobian will then only be evaluated at the GLL points, it certainly suffices to consider an isoparametric mapping, which we denote by $\mathcal{F} = \mathcal{F}_N(\xi)$, and which can be expressed explicitly as

$$x(\xi) = \mathcal{F}_N(\xi) = \sum_{i=0}^{N} x_i \ell_i(\xi), \tag{2.11}$$

where $x_i = \mathcal{F}_N(\xi_i)$, $i = 0, \ldots, N$ are the physical points corresponding to the GLL points, and with $x_0 = 0$ and $x_N = 1$ (the domain boundary in this case).

In principle, we can then formulate an adaptive spectral method as follows: find $\mathcal{F}_N \in \mathbb{P}_N(\widehat{\Omega})$ and $u_N^* \in X_N(\mathcal{F}_N)$ such that

$$||u - u_N^*||_{L^2(\Omega} = \min_{u_N \in X_N(\mathcal{F}_N)} ||u - u_N||_{L^2(\Omega)}, \tag{2.12}$$

where u_N satisfies

$$a(u_N, v) = \ell(v), \qquad \forall v \in X_N(\mathcal{F}_N). \tag{2.13}$$

We have here written X_N as $X_N(\mathcal{F}_N)$ to explicitly emphasize that the discrete space depends on the mapping used. The ideas presented here is discussed in more details in Paper 5 together with some initial numerical results.

3 Spectral methods in $\mathbb{R}^2$

The main motivation behind the work presented in this thesis has been to improve the numerical solution of partial differential equations in complex domains using high order methods. As a simple example, consider the numerical solution of the Poisson problem in a deformed quadrilateral domain Ω. A numerical solution based on high order polynomials necessitates an accurate representation of the geometry. This is typically achieved by first constructing an accurate representation of the boundary of the domain, $\partial\Omega$, and then constructing a mapping between the reference domain $\hat{\Omega} = (-1, 1)^2$ and Ω.

Despite the fact that spectral (element) methods have been used to solve PDEs in complex geometries for a long time [9, 13, 21, 29], few results exist in the literature for how to best construct a high order representation of a single curve in the plane. In the case of a deformed quadrilateral, we need to approximate four curves in the plane (the four edges of Ω) before we are able to construct the mapping between $\hat{\Omega}$ and Ω.

3.1 Geometry representation in deformed domains

If the PDE is defined in a deformed quadrilateral domain (i.e., $\Omega \neq \widehat{\Omega}$), it is mapped to the reference domain before it is solved. For methods that are based on a weak formulation of the problem, this means in practice one or more changes of variables in order to make all the integrals go from -1 to 1. The change of variable can be viewed as a realization of a *mapping* $\mathcal{F}$ from the reference domain $\widehat{\Omega}$ to the physical domain Ω. Figure 5 shows what this mapping can look like when the spatial dimension is $d = 2$. The mapping is required to be one-to-one, and it is often required that both $\mathcal{F}$ and $\mathcal{F}^{-1}$ are analytic, in which case $\mathcal{F}$ is a *diffeomorphism.*

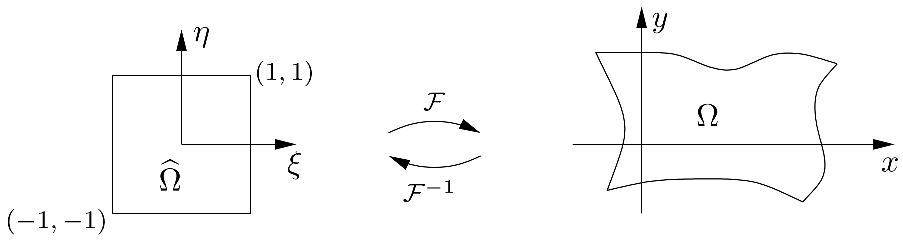


Figure 5: The mapping $\mathcal{F}$ between the reference domain $\widehat{\Omega}$ and the physical domain Ω in $\mathbb{R}^2$. The basis functions, numerical quadrature and differentiation are all defined on $\widehat{\Omega}$.

In high order methods it is common to take an *isoparametric approach*, i.e., to represent the geometry with polynomials of the same degree as the pertinent field variables. In this case the mapping is a d-dimensional tensor-product of polynomials of degree N, and it is denoted $\mathcal{F}_N$. It defines an *approximation* Ω_N of the physical domain. The isoparametric approach makes sense for several reasons. First, differentiation matrices that are already available for the field variables can be re-used. Second, if the geometry is deformed, errors in the representation of the geometry itself contributes to the overall discretization error. The isoparametric approach often ensures a balance in the work we put into approximating the field variables and approximating the geometry.

In $\mathbb{R}^2$, the numerical solution $\hat{u}_N$ and the isoparametric mapping $\mathcal{F}_N$ are both polynomials of degree N in each free variable in the reference domain $\widehat{\Omega} = [-1,1] \times [-1,1] \subset \mathbb{R}^2$. The mapping of the geometry is described by the two functions

$$\begin{aligned} x = \mathcal{F}_N^1(\xi,\eta) &= \sum_{i=0}^{N}\sum_{j=0}^{N} x_{ij}\ell_i(\xi)\ell_j(\eta), \\ y = \mathcal{F}_N^2(\xi,\eta) &= \sum_{i=0}^{N}\sum_{j=0}^{N} y_{ij}\ell_i(\xi)\ell_j(\eta), \end{aligned} \tag{3.1}$$

where ξ and η are the two free variables in $\widehat{\Omega}$. From this representation we note that:

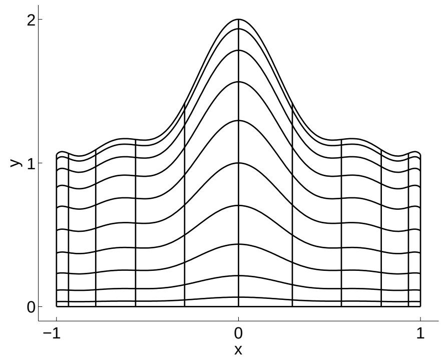
- The mapping is uniquely determined by the values of the $2(N+1)^2$ basis coefficients x_{ij} and y_{ij}, $0 \le i,j \le N$.
- Each point (x_{ij}, y_{ij}) is the image of the point (ξ_i, ξ_j) on a tensor-product GLL grid in $\widehat{\Omega}$.

Let us now discuss how the coefficients may be determined. If the exact mapping $\mathcal{F}$ from $\widehat{\Omega}$ to Ω is explicitly available, the points can be determined simply by evaluating $\mathcal{F}$ in the tensor-product GLL points, i.e., by interpolating $\mathcal{F}$ to construct the approximate mapping $\mathcal{F}_N$. The accuracy in the approximation of the geometry, and hence the overall discretization error in the numerical solution of the PDE, will then depend on $\mathcal{F}$.

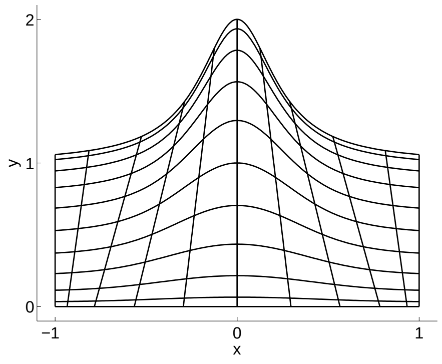
However, the exact mapping is often not available. Instead we have a description of each of the four edges (or curves) that make up the boundary $\partial\Omega$. In such cases the mapping $\mathcal{F}_N$ can be constructed by first determining the coefficients (x_{ij}, y_{ij}) that are associated with the boundary $\partial\Omega_N$, and then extend the mapping to the interior. Two common ways of achieving the latter is *transfinite interpolation* and *harmonic extension*. The former can be done using the Gordon-Hall algorithm [18], which creates $\mathcal{F}_N$ by multiplying the approximation of the boundary edges by weight functions and summing up. The latter can be done by solving an elliptic PDE with non-homogeneous Dirichlet boundary conditions. Both methods usually give smooth mappings when the boundary mappings are smooth, so the choice is often not vital for the overall performance of the high order method. However, harmonic extension can in some cases give "overspill" over the boundary in cases where the Gordon-Hall method does not [31].

The key to a good geometry representation is a good representation of the boundary. However, what a good representation means and how to achieve it turns out to be (at least partially) unsolved problems. Despite the fact that high order methods have been used to solve PDEs in complex geometries for a long time, few results exist in the literature for how to best construct a high order representation of the geometry.

Figure 6 shows two numerical approximations of a domain with three straight edges and a curved edge, described by the Runge function. The two mappings $\mathcal{F}_N$ are created using the Gordon-Hall algorithm, but with two different representations of the top edge. In the left plot, the x-coordinate of the grid points on the top edge are the GLL points, the same as on the bottom edge. However, this results in a poor approximation of the Runge function, and we see oscillations along the top boundary. In the right plot, a different set of interpolation points along the top edge is used, customized for this domain. The difference in the horizontal displacement of the interpolation points along the top and bottom edges results in non-linear vertical grid-lines when using the Gordon-Hall algorithm, despite the fact that both side edges are straight. However, this is no problem; the improvement in the representation of the curved boundary is more important, and the resulting approximation Ω_N of Ω is far better than in the left figure. The corresponding numerical solution u_N of a Poisson problem defined in Ω is also much improved.



(a) The standard method



(b) The equal-tangent method

Figure 6: Computational grids on a domain where the top boundary curve is described by the Runge function. The mappings $\mathcal{F}_N$, where $N = 10$, are created using the Gordon-Hall algorithm; only the representation of the boundary $\partial\Omega$ is different between the two.

The boundary $\partial\widehat{\Omega}$ of the reference domain will be mapped to the boundary $\partial\Omega$ of the physical domain. Each of the four curves that make up the boundary

$\partial\Omega_N$ of the computational domain corresponds to either $\xi = \pm 1$ or $\eta = \pm 1$. From (3.1) we see that the given boundary curve is represented as a *parametric curve* $(p_1(\xi), p_2(\xi))$ (and correspondingly for η), where each of the parametric functions is a polynomial of degree N. Since it is natural (though not necessary) to map boundary points on the tensor-product GLL grid to points on $\partial\Omega$, the coefficients corresponding to these points are often chosen to be coordinates on the boundary curve in the physical domain. Hence, the boundary curves of $\partial\Omega_N$ are polynomial parametric curves that interpolate a given exact curve. This interpolation problem is independent of the high order method used and the extension method used to represent the interior; we can therefore study it separately; high order interpolation of parametric curves in $\mathbb{R}^2$ will represent a key ingredient. In three space dimensions, accurate interpolation of curves and surfaces in $\mathbb{R}^3$ will be essential. Papers 1–3 discuss these issues in more detail.

4 Free surface problems

Consider a time-dependent surface (or a front) depicted in Figure 7. Assume that we know the front at time t^n. Assume also that a numerical approximation of the front is used, e.g., a high order polynomial approximation.

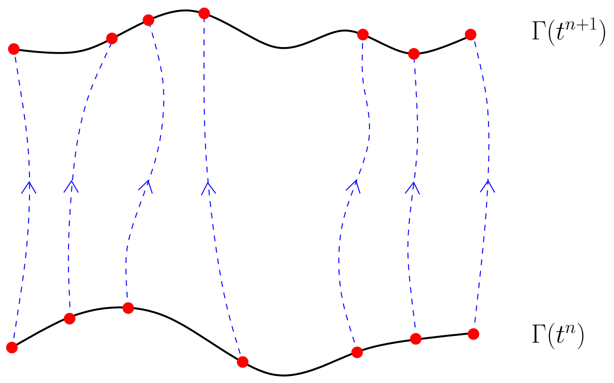


Figure 7: A front Γ at time t^n with an "optimal" point distribution. For example, the points can be the nodes along an edge of a deformed spectral element. The front is "immersed" in a velocity field and the particles follow the paths of the dashed lines between t^n and t^{n+1}.

At each point $\mathbf{x}$ along the surface there is an associated velocity field $\mathbf{u}$.

This velocity field can be explicitly known, e.g., as the solution of an underlying partial differential equation (for example, the solution of the Navier-Stokes equations in a free surface problem).

The velocity at a point along the surface represents the velocity of the corresponding "fluid particle". If we integrate the velocity of all the fluid particles along the surface, we obtain the position of the surface at a later time. This is what a pure Lagrangian description will give us; the motion of a particle is simply governed by the equation

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{u}(\mathbf{x}, t). \tag{4.1}$$

In a computational setting, we can limit the integration of (4.1) to the grid points, and then use the underlying surface parameterization to represent the entire surface at a later time; see Figure 7. A severe problem with this approach is obvious: we have no control over the distribution of the grid points at a later time t^{n+1}. This will again result in a loss of accuracy in the calculation of surface quantities, e.g., tangent and normal vectors, as well as the local curvature.

An advantage with the Arbitrary Lagrangian Eulerian (ALE) formulation is that it introduces a separate domain velocity $\mathbf{w}$ (also referred to as the grid velocity in the context of the discrete problem), which limits the deformation of the computational domain [25]. In an ALE-framework, the position of the interface is advanced according to

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{w}(\mathbf{x}, t) \tag{4.2}$$

instead of the pure Lagrangian approach (4.1). A continuum description dictates that $\mathbf{w} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n}$ (the kinematic condition), where $\mathbf{n}$ is the unit normal along the interface. However, no particular condition is required for the tangential component $\mathbf{w} \cdot \mathbf{t}$ of the domain velocity ($\mathbf{t}$ is the unit tangent vector). A common choice is to set $\mathbf{w} \cdot \mathbf{t} = 0$ along the surface, although this is often not an optimal choice; see Figure 8.

Let us also comment on the issue of temporal accuracy. Integration of (4.1) and (4.2) is commonly done using an explicit method; often an explicit multistep method (e.g., Adams-Bashforth) is preferred [22, 6]. If the velocity fields ($\mathbf{u}$ and $\mathbf{w}$) are sufficiently regular, we expect to achieve higher order temporal accuracy (second and third) in terms of the *location* of individual points along the front. As mentioned above, this approach may yield limited control over the *distribution* of the points along the front. If the point distribution is non-optimal, the resulting loss of spatial accuracy *will* affect the accuracy of surface

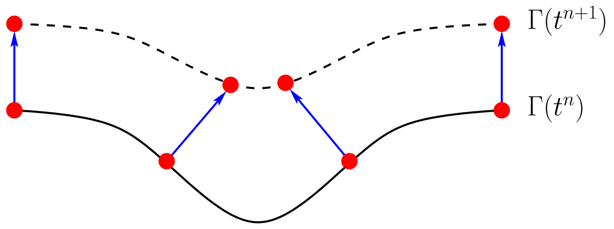


Figure 8: A front at time t^n with an "optimal" point distribution. This interface is advanced by honoring the kinematic condition, while imposing a zero tangential grid velocity. The resulting point distribution at a later time t^{n+1} is obviously no longer optimal.

quantities such as normal and tangent vectors, local curvature, and length/area, and this again may affect the accuracy of the interface tracking.

In order to develop a computational approach which will yield both high order temporal accuracy (e.g., second or third order), as well as good spatial accuracy in the calculation of surface quantities, it is crucial to solve the problem of automatically obtaining a good point distribution in a satisfactory way. This problem is particularly acute in the context of using high order methods. Despite the importance of this issue, very limited discussion or results appear to be available in the literature [4].

One of the obstacles in the study of moving geometries is the lack of (geometrically interesting) exact solutions. Another factor is the complexity and the cost of solving fluid flow problems in $\mathbb{R}^3$. Both of these problems are (partially) solved by considering *minimal surface* problems. These are problems where one represents only a surface in $\mathbb{R}^3$, not a three-dimensional body, and there exist some known interesting exact solutions. Finding minimal surfaces can be done in many ways; one of them is by numerical solution of a surface that evolves toward the area minimizer. In the context of time-dependent PDEs, minimal surfaces are then steady-state solutions. Even if the exact solutions in the prior time steps are still unknown, the known steady-state solution enables us to compare various mesh update strategies by how well the representation of the steady-state solution is. These and other issues are studied in Paper 4; we only give a brief introduction below.

4.1 Minimal surfaces

A minimal surface is a surface with the smallest possible area under certain constraints (boundary conditions, volume constraints etc.). These surfaces are mathematically intriguing, but still easily realized physically in the form of soap film [26, 30]. The study of minimal surfaces dates back to the eighteenth century, with the discovery of the catenoid (Euler, 1744) and the helicoid (Meusnier, 1776). Still, only a limited number of minimal surfaces are known, but new ones are still being found [14].

An important characteristic of minimal surfaces is that they have zero *mean curvature*, which is defined as the average of the two principal curvatures, i.e., $\kappa = \frac{1}{2}(\kappa_1 + \kappa_2)$. For surfaces that can be represented as functions $u(x, y)$ over a domain $\Omega \subset \mathbb{R}^2$, open minimal surfaces are those that satisfy Plateau's equation

$$\operatorname{div}\left(\frac{\nabla u}{\sqrt{1 + |\nabla u|^2}}\right) = 0, \tag{4.3}$$

with given boundary conditions. The equation is named after the Belgian physicist J. A. F. Plateau, who studied soap films experimentally and determined some interesting geometric properties [38]. A more general representation is a parametric surface, where the surface is represented by a mapping $\boldsymbol{\varphi} : \widehat{\Omega} \subset \mathbb{R}^2 \to \mathbb{R}^3$, i.e., $\boldsymbol{\varphi}$ is of the form $\boldsymbol{f}$ in (1.12). The minimal surface condition is then

$$\begin{aligned} \Delta_\Omega \boldsymbol{\varphi} &= \mathbf{0} \quad && \text{in } \widehat{\Omega}, \\ \boldsymbol{\varphi} &= \boldsymbol{\varphi}_0 \quad && \text{on } \partial\widehat{\Omega}, \end{aligned} \tag{4.4}$$

where Δ_Ω is the *Laplace-Beltrami operator*, a generalization of the Laplace operator to Riemannian manifolds [45]. The first set of equations represents a system of three non-linear partial differential equations, while the last set of equations represents the boundary conditions (the position of the known wire frame).

Since the minimal surface equations (4.4) are non-linear, it is very difficult to solve them directly. A more feasible approach is to solve them with Newton's method. This requires the construction of an *initial surface*, i.e., a parametric surface that fits the boundary conditions. From each iteration level to the next, a new parametric surface $\boldsymbol{\varphi}^{n+1}$ is constructed based on the previous one $\boldsymbol{\varphi}^n$ by the addition of a displacement

$$\boldsymbol{\varphi}^{n+1} = \boldsymbol{\varphi}^n + \Delta\boldsymbol{\varphi}^{n+1} \tag{4.5}$$

that is determined by the governing equations. A purely Lagrangian approach means updating the surface exactly according to the solution of the governing equations. However, one can choose to add a *tangential displacement* in order to retain a smooth mapping of the surface. This is where mesh update strategies are needed, and algorithms from numerical solutions of free surface problems can be used. The problem is discussed in detail in Paper 4, where customized strategies are compared with common general-purpose strategies, and a new general-purpose strategy is proposed.

Figure 9 shows an example of a minimal surface along with the surface used as initial condition for the computation.

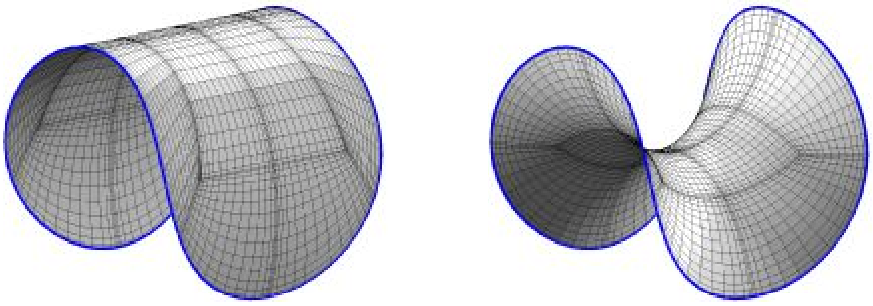

(a) Initial surface **(b)** Minimal surface

Figure 9: Enneper's surface is a minimal surface with a single boundary curve (shown in blue). The surface is parametrized using 12 spectral elements.

5 Summary of papers

5.1 Overview

The papers in this thesis are all related to the topic of geometry representation and coordinate transformations in spectral approximation of PDEs, but from different angles. Paper 1 focuses on the representation of deformed quadrilaterals in $\mathbb{R}^2$ and contains both a discussion of parametric curve interpolation and application to the solution of different Poisson problems. Paper 3 considers parametric curves and surfaces in $\mathbb{R}^3$ and discusses only the interpolation part, not the PDE part. Paper 2 is from the proceedings of the ICOSAHOM-09 conference and touches upon many of the topics discussed in papers 1 and 3.

Paper 4 considers moving boundary problems in the context of high order approximation of minimal surfaces. The problem involves geometry representation in $\mathbb{R}^3$, but only surfaces are considered, not deformed hexahedra. For this class of problems, optimal representation of the geometry is currently infeasible, and the main focus is to avoid break-down in the numerical algorithm due to large errors in the geometry representations. However, in special cases we have an exact steady-state solution and we are able to compare the quality of different geometry update strategies.

The last paper considers customization of the coordinate transformations in order to improve the representation of the field variables, i.e., adaptive methods for the numerical solution of differential equations in $\mathbb{R}^1$.

5.2 List of papers

Paper 1: T. Bjøntegaard, E. M. Rønquist and Ø. Tråsdahl. *Spectral approximation of partial differential equations in highly distorted domains.* To appear in Journal of Scientific Computing. DOI: 10.1007/s10915-011-9561-8

In this paper we discuss spectral approximations of the Poisson equation in deformed quadrilateral domains. High order polynomial approximations are used for both the solution and the representation of the geometry. Following an isoparametric approach, the four edges of the computational domain are first parametrized using high order polynomial interpolation. Transfinite interpolation is then used to construct the mapping from the square reference domain to the physical domain. Through a series of numerical examples we show the importance of representing the boundary of the domain in a careful way; the choice of interpolation points along the edges of the physical domain may significantly affect the overall discretization error. One way to ensure good interpolation points along an edge is based on the following criteria: (i) the points should be on the exact curve; (ii) the derivative of the exact curve and the interpolant should coincide at the internal points along the edge. Following this approach, we demonstrate that the discretization error for the Poisson problem may decay exponentially fast even when the boundary has low regularity.

Paper 2: T. Bjøntegaard, E. M. Rønquist and Ø. Tråsdahl. *High order polynomial interpolation of parameterized curves.* Lecture Notes in Computational Science and Engineering, Vol. 76, 365–372, 2011.

We consider interpolation of parameterized curves in $\mathbb{R}^2$ and $\mathbb{R}^3$. Such curves are represented by vector-valued functions, but interpolation differs from classical interpolation of functions since a curve can be *reparameterized.* This means that a presumably good interpolation (e.g., at the Gauss points) of a given parameterization does not necessarily give the best approximation of the curve, as there may exist a reparameterization better suited for polynomial interpolation. The reparameterization can be done implicitly by choosing different sets of interpolation points along the exact curve. We present common interpolation methods, and propose a new method, based on choosing the interpolation points in such a way that the interpolant is tangential to the exact (reparameterized) curve at these points. The new method is compared to the traditional ones in a series of numerical examples, and results show that classical interpolation is sometimes far from optimal in the sense of the Kolmogorov n-width, i.e., the best approximation using n degrees-of-freedom.

Paper 3: Ø. Tråsdahl. *High order interpolation of parametric curves and surfaces in* $\mathbb{R}^3$. NTNU Preprint Numerics No. 5/2011. Submitted to Advances in Computational Mathematics.

In this paper, high order interpolation of parametric curves and surfaces in $\mathbb{R}^3$ is studied. Curve interpolation is discussed in much greater detail than in Paper 2, and many numerical examples are provided to show the performance of the different interpolation methods in different situations. Two of the proposed interpolation methods, the extra-points method and the equal-tangent method, are discussed in the context of an unproven conjecture made in the community of Computer Aided Geometric Design (CAGD), concerning the highest attainable approximation order in fixed-degree interpolation. This conjecture is based on counting the number of degrees-of-freedom in interpolation of parametric curves. Some of the interpolation methods are extended to interpolation of parametric surfaces, where vast differences in convergence properties between different interpolation methods are observed in some cases. Most notably, exponential convergence is achieved by one of the proposed methods for a surface described by a function of low regularity.

Paper 4: Ø. Tråsdahl and E. M. Rønquist. *High order numerical approximation of minimal surfaces.* Journal of Computational Physics, 230(12): 4795–4810, 2011.

We present an algorithm for finding high order numerical approximations of minimal surfaces with a fixed boundary. The algorithm employs parametrization by high order polynomials and a linearization of the weak formulation of the Laplace-Beltrami operator to arrive at an iterative procedure to evolve from a given initial surface to the final minimal surface. For the steady state solution we measure the approximation error in a few cases where the exact solution is known. In the framework of parametric interpolation, the choice of interpolation points (mesh nodes) is directly affecting the approximation error, and we discuss how to best update the mesh on the evolutionary surface such that the parametrization remains smooth. In our test cases we may achieve exponential convergence in the approximation of the minimal surface as the polynomial degree increases, but the rate of convergence greatly differs with different choices of mesh update algorithms. The present work is also of relevance to high order numerical approximation of fluid flow problems involving free surfaces.

Paper 5: Ø. Tråsdahl and E. M. Rønquist. *Adaptive spectral methods.* Technical report. NTNU Preprint Numerics No. 1/2012.

This paper discusses numerical solution of boundary value problems using spectral methods combined with nonlinear and adaptive mappings between the reference domain and the physical domain. A brief review of existing methods for adaptive mesh generation is given, and a method for finding close to optimal mappings for boundary value problems in $\mathbb{R}^1$ is presented. The method exploits the link between high order numerical solutions of PDEs and approximation of parametric curves. Also, other adaptive methods for boundary value problems in $\mathbb{R}^1$ are proposed, based either on minimizing the discrete L^2-norm of the residual, or interpolating the residual as a parametric curve. The adaptive methods are constructed with the aim of finding optimal mappings, however, this turns out to be a very difficult task. Still, significant improvement from standard (non-adaptive) high order methods is achieved in some cases.

Bibliography

[1] M. Ainsworth and J. T. Oden. A posteriori error estimation in finite element analysis. *Comput. Methods Appl. Mech. Engrg.*, 142(1-2):1–88, 1997.

[2] I. Babuška and M. Suri. The p and h-p versions of the finite element method, basic principles and properties. *SIAM Rev.*, 36(4):578–632, 1994.

[3] A. Bayliss, D. Gottlieb, B. J. Matkowsky, and M. Minkoff. An adaptive pseudo-spectral method for reaction diffusion problems. *J. Comput. Phys.*, 81(2):421–443, 1989.

[4] T. Bjøntegaard and E. M. Rønquist. Accurate interface-tracking for arbitrary Lagrangian-Eulerian schemes. *J. Comput. Phys.*, 228(12):4379–4399, 2009.

[5] N. Bodard, R. Bouffanais, and M. O. Deville. Solution of moving-boundary problems by the spectral element method. *Appl. Numer. Math.*, 58(7):968–984, 2008.

[6] R. Bouffanais and M. O. Deville. Mesh update techniques for free-surface flow solvers using spectral elements. *J. Sci. Comput.*, 27(1-3):137–149, 2006.

[7] J. P. Boyd. *Chebyshev and Fourier spectral methods.* Dover Publications Inc., 2nd edition, 2001.

[8] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods, Fundamentals in Single Domains.* Springer, 2006.

[9] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods, Evolution to Complex Geometries and Applications to Fluid Dynamics.* Springer, 2007.

[10] C. de Boor, K. Höllig, and M. Sabin. High accuracy geometric Hermite interpolation. *Comput. Aided Geom. Design*, 4(4):269–278, 1987.

[11] W. L. F. Degen. High accurate rational approximation of parametric curves. *Comput. Aided Geom. Design*, 10(3-4):293–313, 1993.

[12] W. L. F. Degen. Geometric Hermite interpolation – in memoriam Josef Hoschek. *Comput. Aided Geom. Design*, 22(7):573–592, 2005.

[13] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flow.* Cambridge University Press, 2002.

[14] U. Dierkes, S. Hildebrandt, A. Küster, and O. Wohlrab. *Minimal surfaces. I. Boundary value problems*, volume 295 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences].* Springer-Verlag, 1992.

[15] Y. Y. Feng and J. Kozak. On G^2 continuous cubic spline interpolation. *BIT*, 37(2):312–332, 1997.

[16] M. S. Floater. An $O(h^{2n})$ Hermite approximation for conic sections. *Comput. Aided Geom. Design*, 14(2):135–151, 1997.

[17] B. Fornberg. *A Practical Guide to Pseudospectral Methods.* Cambridge University Press, 1998.

[18] W. J. Gordon and C. A. Hall. Construction of curvilinear co-ordinate systems and applications to mesh generation. *Internat. J. Numer. Methods Engrg.*, 7(4):461–477, 1973.

[19] D. Gottlieb and S. A. Orszag. *Numerical Analysis of Spectral Methods: Theory and Applications*, volume 26 of *Regional Conference Series in Applied Mathematics.* SIAM, 1977.

[20] J. S. Hesthaven. From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex. *SIAM J. Numer. Anal.*, 35(2):655–676, 1998.

[21] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods.* Springer, 2008.

[22] L. W. Ho and A. T. Patera. A Legendre spectral element method for simulation of unsteady incompressible viscous free-surface flows. *Comput. Methods Appl. Mech. Engrg.*, 80(1-3):355–366, 1990.

[23] L. W. Ho and A. T. Patera. Variational formulation of three-dimensional viscous free-surface flows: Natural imposition of surface tension boundary conditions. *Int. J. Numer. Methods Fluids*, 13(6):691–698, 1991.

[24] K. Hollig and J. Koch. Geometric Hermite interpolation. *Comput. Aided Geom. Design*, 12(6):567–580, 1995.

[25] A. Huerta and A. Rodríguez-Ferran (eds.). The Arbitrary Lagrangian-Eulerian Formulation. *Comput. Methods Appl. Mech. Engrg.*, 193(39-41):4073–4456, 2004.

[26] C. Isenberg. *The Science of Soap Films and Soap Bubbles.* Dover Publications Inc., 1992.

[27] G. Jaklič, J. Kozak, M. Krajnc, V. Vitrih, and E. Žagar. On geometric Lagrange interpolation by quadratic parametric patches. *Comput. Aided Geom. Design*, 25(6):373–384, 2008.

[28] G. Jaklič, J. Kozak, M. Krajnc, and E. Žagar. On geometric interpolation by planar parametric polynomial curves. *Math. Comput.*, 76(260):1981–1993, 2007.

[29] G. E. Karniadakis and S. J. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics.* Oxford University Press, 2nd edition, 2005.

[30] D. R. Lovett and J. Tilley. *Demonstrating Science With Soap Films.* Institute of Physics Pub., 1994.

[31] A. E. Løvgren, Y. Maday, and E. M. Rønquist. Global C^1 maps on general domains. *Math. Models Methods Appl. Sci.*, 19(5):803–832, 2009.

[32] Y. Maday and A. T. Patera. Spectral element methods for the Navier-Stokes equations. *In: A. K. Noor, J. T. Oden (eds.), State of the Art Surveys in Computational Mechanics, ASME, New York*, pages 71–143, 1989.

[33] K. Mørken. On geometric interpolation of parametric surfaces. *Comput. Aided Geom. Design*, 22(9):838–848, 2005.

[34] K. Mørken and K. Scherer. A general framework for high-accuracy parametric interpolation. *Math. Comput.*, 66(217):237–260, 1997.

[35] L. S. Mulholland, W.-Z. Huang, and D. M. Sloan. Pseudospectral solution of near-singular problems using numerical coordinate transformations based on adaptivity. *SIAM J. Sci. Comput.*, 19(4):1261–1289, 1998.

[36] R. Pasquetti and F. Rapetti. Spectral element methods on unstructured meshes: which interpolation points? *Numer. Algorithms*, 55(2-3):349–366, 2010.

[37] A. T. Patera. A spectral element method for fluid dynamics: laminar flow in a channel expansion. *J. Comput. Phys.*, 54(3):468–488, 1984.

[38] J. A. F. Plateau. *Statique expérimentale et théorique des liquides soumis aux seules forces moléculaires.* Gauthier-Villars, 1873.

[39] A. Rababah. High order approximation method for curves. *Comput. Aided Geom. Design*, 12(1):89–102, 1995.

[40] R. Schaback. Interpolation with piecewise quadratic visually C^2 Bézier polynomials. *Comput. Aided Geom. Design*, 6(3):219–233, 1989.

[41] C. Schwab. *p- and hp-Finite Element Methods.* Oxford University Press, 1998.

[42] M. A. Taylor, B. A. Wingate, and R. E. Vincent. An algorithm for computing Fekete points in the triangle. *SIAM J. Numer. Anal.*, 38(5):1707–1720, 2000.

[43] T. W. Tee and L. N. Trefethen. A rational spectral collocation method with adaptively transformed Chebyshev grid points. *SIAM J. Sci. Comput.*, 28(5):1798–1811, 2006.

[44] R. Verfürth. *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques.* John Wiley & Sons Inc, 1996.

[45] T. J. Willmore. *Riemannian geometry.* Oxford University Press, 1993.

[46] J. H. Yong and F. F. Cheng. Geometric Hermite curves with minimum strain energy. *Comput. Aided Geom. Design*, 21(3):281–301, 2004.

PAPER 1

SPECTRAL APPROXIMATION OF PARTIAL DIFFERENTIAL EQUATIONS IN HIGHLY DISTORTED DOMAINS

TORMOD BJØNTEGAARD, EINAR M. RØNQUIST AND ØYSTEIN TRÅSDAHL

To appear in
Journal of Scientific Computing.
DOI: 10.1007/s10915-011-9561-8

SPECTRAL APPROXIMATION OF PARTIAL DIFFERENTIAL EQUATIONS IN HIGHLY DISTORTED DOMAINS

TORMOD BJØNTEGAARD, EINAR M. RØNQUIST AND ØYSTEIN TRÅSDAHL

Department of Mathematical Sciences,
Norwegian University of Science and Technology,
Trondheim, Norway

Abstract

In this paper we discuss spectral approximations of the Poisson equation in deformed quadrilateral domains. High order polynomial approximations are used for both the solution and the representation of the geometry. Following an isoparametric approach, the four edges of the computational domain are first parametrized using high order polynomial interpolation. Transfinite interpolation is then used to construct the mapping from the square reference domain to the physical domain. Through a series of numerical examples we show the importance of representing the boundary of the domain in a careful way; the choice of interpolation points along the edges of the physical domain may significantly affect the overall discretization error. One way to ensure good interpolation points along an edge is based on the following criteria: (i) the points should be on the exact curve; (ii) the derivative of the exact curve and the interpolant should coincide at the internal points along the edge. Following this approach, we demonstrate that the discretization error for the Poisson problem may decay exponentially fast even when the boundary has low regularity.

Keywords: Spectral approximation, boundary representation, polynomial interpolation, reparametrization, mapping.

1 Introduction

The motivation behind the work presented in this paper has been to solve partial differential equations in complex domains using high order methods. As a simple example, consider the numerical solution of the Poisson problem in a deformed quadrilateral domain Ω. A numerical solution based on high order polynomials necessitates an accurate representation of the geometry. This is typically achieved by first constructing an accurate representation of the boundary of the domain, $\partial\Omega$, and then constructing a mapping between the reference domain $\hat{\Omega} = (-1, 1)^2$ and Ω. Assuming that the physical domain is not too distorted, the latter can readily be achieved via a Gordon-Hall transfinite interpolation procedure [9].

Despite the fact that spectral (element) methods have been used to solve PDEs in complex geometries for a long time [4, 7, 10, 11], few results exist in the literature for how to best construct a high order representation of the boundary of the domain. In the case of a deformed quadrilateral, we need to approximate four curves in the plane (the four edges of Ω) before we are able to construct the mapping between $\hat{\Omega}$ and Ω.

In this paper we investigate the impact of the choice of geometry representation in the context of solving partial differential equations using spectral methods. We compare previously used methods with a new method.

The outline of the paper is as follows. In Section 2 we discuss an illustrative example of the spectral solution of a Laplace problem. In Section 3 we discuss in some detail alternative ways to approximate a curve in the plane. In Section 4 we again discuss the impact of the different boundary representations on the discretization error of various Poisson problems. Finally, in Section 5 we summarize our main findings.

2 Solving a Laplace problem: Case 1

As a first example, we consider the numerical approximation of the following two-dimensional Laplace problem

$$\begin{aligned} \nabla^2 u &= 0 && \text{in } \Omega, \\ u &= e^x \sin y && \text{on } \partial\Omega, \end{aligned} \tag{2.1}$$

where Ω is a deformed square. Specifically, we consider a domain with three straight edges ($x = \pm 1$ and $y = 0$) and a deformed top edge having the shape of the Runge function [8]

$$y^R(x) = \frac{1}{1 + 16x^2} \qquad \text{for} \quad -1 \le x \le 1.$$

The exact solution to (2.1) is $u(x, y) = e^x \sin y$, which is analytic.

We discretize (2.1) based on the equivalent weak form. To this end, we use a pure spectral discretization based on high order polynomials [12, 1, 7, 4]. As usual, the integrals in the weak form are transformed to integrals over the associated reference domain $\hat{\Omega} = (-1, 1)^2$, and exact integration is replaced by Gauss-Lobatto Legendre (GLL) quadrature.

The mapping between $\hat{\Omega}$ and Ω is constructed based on transfinite interpolation [9], also referred to as the Gordon-Hall algorithm. In the following, we consider an isoparametric mapping, i.e., the geometry is approximated using high order polynomials of degree N similar to the numerical solution u_N. The resulting computational domain is denoted as Ω_N. Using a tensor-product basis to represent the field variables, the mapping between $(\xi, \eta) \in \hat{\Omega}$ and $(x_N, y_N) \in \Omega_N$ is given explicitly as

$$x_N(\xi, \eta) = \sum_{i=0}^{N} \sum_{j=0}^{N} x_{ij} \ell_i(\xi) \ell_j(\eta),$$

$$y_N(\xi, \eta) = \sum_{i=0}^{N} \sum_{j=0}^{N} y_{ij} \ell_i(\xi) \ell_j(\eta),$$

where $\ell_i(\xi)$ is the Nth order Lagrangian interpolant through the GLL points, i.e., $\ell_i(\xi) \in \mathbf{P}_N(-1, 1)$ and $\ell_i(\xi_j) = \delta_{ij}$, $0 \leq i, j \leq N$, and where $x_{ij} = x(\xi_i, \xi_j)$ and $y_{ij} = y(\xi_i, \xi_j)$ are the coordinates of the grid points corresponding to the mapping of the tensor-product GLL points.

The difference between the exact solution u and the numerical solution u_N is measured in the energy norm,

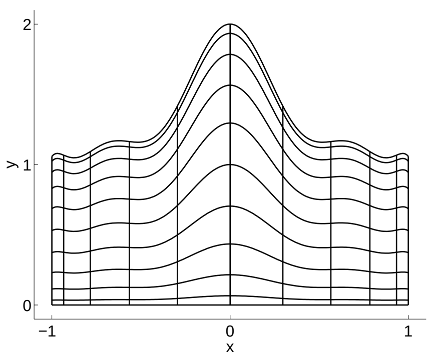
$$|||u - u_N|||^2 = \int_{\Omega_N} \nabla(u - u_N) \cdot \nabla(u - u_N) \, \mathrm{d}A. \tag{2.2}$$

Again, the integral on the right-hand side is transformed to an integral over the associated reference domain $\hat{\Omega} = (-1, 1)^2$, and exact integration is replaced by GLL quadrature. The error measure (2.2) will take into account the quality of the isoparametric mapping for a pure spectral discretization of (2.1). In order to eliminate quadrature errors in the error computations, we will use overintegration based on a polynomial degree $3N$.

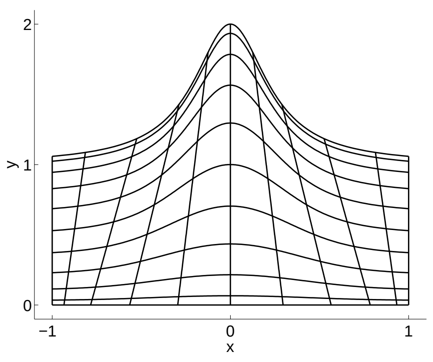
In order to use the Gordon-Hall algorithm, we first approximate the boundary $\partial\Omega$ using high order polynomial interpolation; specifically, we need to choose the grid points along the four edges of Ω. The choice of grid points along the three straight edges is straightforward and corresponds to a GLL distribution.

However, we may consider different alternatives for the choice of grid points along the top edge. In the following, we will compare four different methods. In the standard method, the grid points are chosen according to a GLL distribution along the x-direction; see Figure 1(a). This corresponds to a standard interpolation of the Runge function in the GLL points. In the arc length method, the grid points along the top edge are chosen in such a way that they correspond to a GLL distribution in arc length. In the L^2-method, we choose the $N-1$ internal interpolation points along the top edge in such a way that the interpolation error is minimized in the L^2-norm. However, this is a very difficult problem to solve and we therefore introduce an alternative method, denoted as the equal-tangent method, which is able to obtain a similar result, but more easily.

Figure 1 shows the computational grids using the standard method and the equal-tangent method. For $N = 10$ we can still see the ripples along the top edge when using the standard method; these ripples represent a typical feature of a standard polynomial approximation of the Runge function. In contrast, the top edge of the domain looks very smooth using the equal-tangent method. Figure 2 shows the discretization error (2.2) when solving the Laplace problem in the computational domains induced by our choice of grid points along the top edge.



(a) The standard method



(b) The equal-tangent method

Figure 1: Computational grid for Case 1 using a polynomial degree $N = 10$.

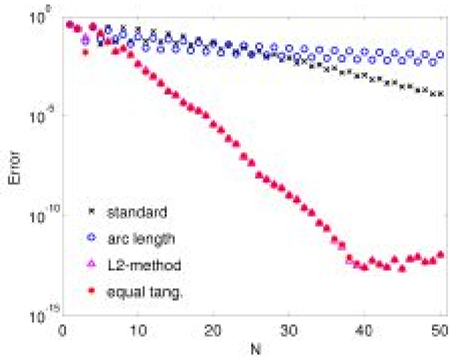


Figure 2: The discretization error in the solution of the Laplace problem (2.1) measured in the energy norm (2.2). The standard method and the arc length method do not represent the top edge well, and the convergence rate is slow. However, the convergence rate of the L^2-method and the equal-tangent method is much faster. The difference in convergence rate between the different methods is fundamentally linked to our choice of interpolation points along the top edge; the rest of the solution procedure is identical for all methods.

3 High-order interpolation of a curve

Before we show more examples of the spectral solution of the Poisson problem, we first discuss in more detail the different methods we use to represent a curve in the plane.

Consider a one-dimensional function $y(x)$, $x \in [a, b]$. A high order polynomial interpolant $y_N(x)$ can be constructed by choosing points x_j, $j = 0, \dots, N$, and evaluating the function at these points, yielding $y_j = y(x_j)$, $j = 0, \dots, N$; see [1, 2, 3]. If the function $y(x)$ is regular, the interpolation error $\|y - y_N\|$ will decay rapidly as N increases.

If the mapping $x(\xi)$ is affine, both the function $y_N(x)$ and the function $y_N(\xi) = y_N(x(\xi))$ are polynomials of degree N. Specifically, $y_N(\xi)$ is given as

$$y_N(\xi) = \sum_{j=0}^{N} y_j \ell_j(\xi), \tag{3.1}$$

where $\ell_j(\xi)$ is the Nth order Lagrangian interpolant through the GLL points, and where $y_j = y(x(\xi_j))$. Note that, in order to keep the notation as simple as possible, we will use the same symbol for the function y both when expressed in terms of the physical coordinate x and in terms of the reference coordinate ξ. It should be clear from the context what is meant.

One way to describe a curve in the plane is to view the curve as a given function $y(x)$, where the specific function $y(x)$ will depend on the orientation of our coordinate system. Another alternative is to use a parametric representation, e.g., a curve in the plane can be given by two functions $g_1(\xi)$ and $g_2(\xi)$,

$$\begin{aligned} x(\xi) &= g_1(\xi), \\ y(\xi) &= g_2(\xi). \end{aligned} \tag{3.2}$$

For any value value $\xi \in [-1, 1]$, there exists a unique point $(x(\xi), y(\xi))$ on the curve. For example, if $g_1(\xi) = a + \frac{b-a}{2}(\xi + 1)$, i.e., an affine mapping $x(\xi)$, we may eliminate ξ by first expressing ξ as a function of x, and then recover the original function $y(x)$ from $y(x) = g_2(\xi(x))$. However, the parametric representation (3.2) allows for additional flexibility, also when it comes to numerical approximation.

In the following we consider high order polynomial approximations $x_N(\xi)$ and $y_N(\xi)$ of $x(\xi)$ and $y(\xi)$. Both $x_N(\xi)$ and $y_N(\xi)$ are elements of $\mathbf{P}_N(-1, 1)$. These approximations can be expressed explicitly as

$$\begin{aligned} x_N(\xi) &= \sum_{j=0}^{N} x_j \ell_j(\xi), \\ y_N(\xi) &= \sum_{j=0}^{N} y_j \ell_j(\xi), \end{aligned} \tag{3.3}$$

where x_j and y_j are the nodal values for each approximation. We will here only consider interpolants of the exact curve $(x(\xi), y(\xi))$, $\xi \in [-1, 1]$, i.e., we require that the nodal points (x_j, y_j), $j = 0, \ldots, N$, are located on the exact curve; in particular, the end points of the exact curve and the numerical curve always coincide. Hence, our problem is to determine the "best" set of (internal) values ξ_j^*, $j = 1, \ldots, N-1$, $-1 < \xi_j^* < 1$, where $x_j = g_1(\xi_j^*)$ and $y_j = g_2(\xi_j^*)$, i.e., this is a problem with $N - 1$ degrees-of-freedom; see [13] for a discussion and analysis.

Let us now briefly consider a few choices which can be used to represent a curve in the plane, e.g., an edge of a deformed quadrilateral domain.

3.1 The standard method

This method assumes that $x(\xi) = g_1(\xi)$ is a simple affine mapping. The numerical approximation is given by (3.3), with nodal values $x_j = a + \frac{b-a}{2}(\xi_j + 1)$ and $y_j = y(x_j)$; see Figure 3. By construction, $y_N(\xi)$ is a polynomial of degree N. However, we also note that $y_N(x)$ will also be a polynomial of degree N, i.e., we consider classical interpolation. In the following, we refer to this method as the "standard" method.

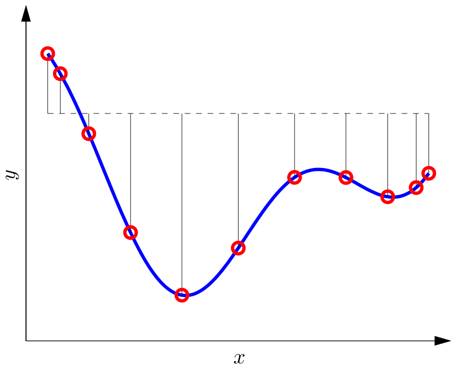


Figure 3: The function $y(x)$ evaluated at the (affinely mapped) GLL-nodes.

3.2 The chord method

Even if the original curve is represented in a coordinate system as depicted in Figure 3, we can transform this representation to a coordinate system where the x-axis is aligned with the chord connecting the two end points of the curve; see Figure 4. This can be achieved through a simple rotation and translation. After the transformation to this new coordinate system (with coordinates x' and y'), we can again apply the standard method as described above.

Note that the transformed function $y'(x')$ is different from the original function $y(x)$ even though both functions describe the same curve. This will, of course, also be true for the associated numerical approximations (x'_N, y'_N) and (x_N, y_N). Note also that the function $y'(x')$ may have steep gradients or may not always exist since y' may take on several different values for a given x'. In such cases the interpolation procedure will give poor results or will break down.

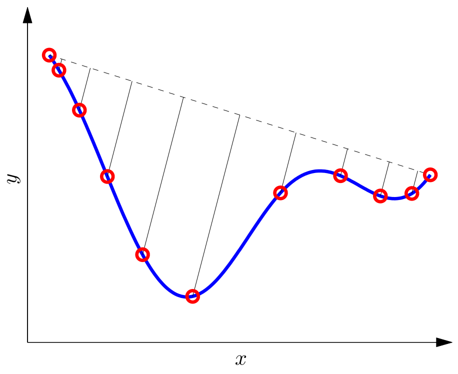


Figure 4: The original function $y(x)$ can be considered in a transformed coordinate system in which the transformed x-axis is aligned with the chord between the two end points of the curve. The function is then evaluated at the (affinely mapped) GLL-nodes relative to this coordinate system.

The parametric representation of the exact curve $(x'(\eta), y'(\eta))$, $\eta \in [-1, 1]$, can be viewed as a reparametrization [14] of the original parametric representation $(x(\xi), y(\xi))$, $\xi \in [-1, 1]$. To this end, there exists a mapping $\xi(\eta)$ from $[-1, 1]$ to $[-1, 1]$ such that $x'(\eta) = x(\xi(\eta))$ and $y'(\eta) = y(\xi(\eta))$.

Using the standard method in the transformed coordinate system, we end up with a set of nodal values (x'_j, y'_j), $j = 0, \dots, N$. These coordinates are then transformed back to the corresponding coordinates (x_j, y_j), $j = 0, \dots, N$, in the original coordinate system. The reason for transforming back to the original coordinate system is motivated by the fact that we ultimately want to use these representations in the context of solving partial differential equations. In this case, all the curve segments associated with the boundary of the computational domain are part of a common description. For example, these representations may be the input to a Gordon-Hall algorithm as discussed earlier.

The numerical representation of the curve will therefore be exactly the same as before and given by (3.3). However, the *numerical values* of (x_j, y_j), $j = 0, \dots, N$, will be different than the standard method applied in the original coordinate system. In the following, we will refer to this method as the "chord" method.

Note that, even though $x_N(\xi)$ and $y_N(\xi)$ both belong to $\mathbf{P}_N(-1, 1)$, the implicitly given function $y_N(x_N)$ is not, in general, a polynomial. In the particular case when the chord is parallel to the x-axis, the chord method coincides with the standard method.

3.3 The arc length method

Let s be the arc length as we move along the curve $y(x)$, with $s = 0$ at $x = a$ and $s = L$ at $x = b$, i.e., the length of the curve is L. Construct the affine mapping $s(\xi) = \frac{L}{2}(\xi+1)$, $\xi \in [-1, 1]$, and define the values $s_j = s(\xi_j)$, $j = 0, \ldots, N$. Each value s_j corresponds to a unique point along the curve with coordinates (x_j, y_j); see Figure 5. The arc length method uses these points as the nodal values in the numerical representation (3.3). In other words, the nodal points are distributed along the curve as a GLL-distribution according to arc length. As for the chord method we note that, even though $x_N(\xi)$ and $y_N(\xi)$ both belong to $\mathbf{P}_N(-1, 1)$, the implicitly given function $y_N(x_N)$ is not, in general, a polynomial.

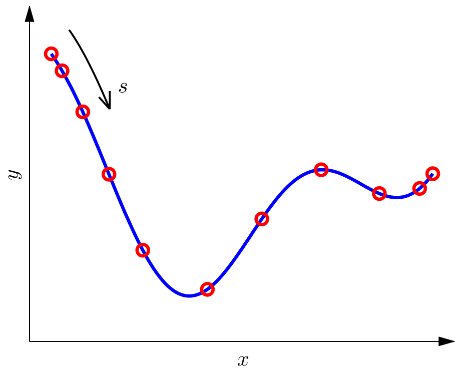


Figure 5: The original function $y(x)$ is here evaluated at points corresponding to a GLL-distribution according to arc length.

3.4 The L^2-method

The three previous representations all rely in one way or another on a point distribution which corresponds to an affinely mapped GLL-distribution: for the standard method, the points along the x-axis correspond to a GLL-distribution; for the chord method, the points along the chord correspond to a GLL-distribution; for the arc length method, the points along the arc of the curve correspond to a GLL-distribution.

We can, of course, also propose a method where there are no such restrictions on the nodal points. Instead, we search for $N-1$ internal points along the curve which will ensure a good numerical approximation to the exact curve. As a way to measure how good the approximation is, we consider the L^2-error

$$\mathcal{J} = ||y - y_N||_{L^2} = \left(\int_a^b (y(x) - y_N(x))^2 \mathrm{d}x \right)^{1/2}. \tag{3.4}$$

The exact curve is given by the parametric representation $(x(\xi), y(\xi))$, $\xi \in [-1, 1]$, while the numerical approximation is given by (3.3). As stated earlier, our problem is to determine the "best" set of (internal) values ξ_j^*, $j = 1, \ldots, N-1$, $-1 < \xi_j^* < 1$, where $x_j = g_1(\xi_j^*)$ and $y_j = g_2(\xi_j^*)$, i.e., this is a problem with $N-1$ degrees-of-freedom. In principle, these unknowns can be determined by minimizing the functional $\mathcal{J}$. Hence, we will refer to such a method as the "L^2-method".

3.5 The equal-tangent method

As mentioned in the previous section, the $N-1$ values ξ_j^*, $j = 1, \ldots, N-1$, $-1 < \xi_j^* < 1$, can be determined by defining $N-1$ independent conditions. As an alternative to the L^2-method we propose the following conditions:

$$\frac{\mathrm{d}x_N}{\mathrm{d}\xi}(\xi_j)\frac{\mathrm{d}y}{\mathrm{d}\xi}(\xi_j^*) - \frac{\mathrm{d}y_N}{\mathrm{d}\xi}(\xi_j)\frac{\mathrm{d}x}{\mathrm{d}\xi}(\xi_j^*) = 0, \qquad j = 1, \ldots, N-1. \tag{3.5}$$

Note that x_N and y_N are evaluated in the GLL points, but still depend on the unknowns ξ_j^* via the coefficients $x_j = x(\xi_j^*)$ and $y_j = y(\xi_j^*)$ in the representation (3.3).

One way to better understand these equations is by considering the exact curve subject to a reparametrization $\xi(\eta)$, $\eta \in [-1, 1]$, $\xi \in [-1, 1]$. That is, we consider the representation $(\tilde{x}(\eta), \tilde{y}(\eta)) = (x(\xi(\eta)), y(\xi(\eta)))$ which, of course, represents the same curve. With this approach, it is possible to find a reparametrization such that the conditions (3.5) can be expressed as

$$\frac{\mathrm{d}x_N}{\mathrm{d}\xi}(\xi_j)\frac{\mathrm{d}\tilde{y}}{\mathrm{d}\eta}(\xi_j) - \frac{\mathrm{d}y_N}{\mathrm{d}\xi}(\xi_j)\frac{\mathrm{d}\tilde{x}}{\mathrm{d}\eta}(\xi_j) = 0, \qquad j = 1, \ldots, N-1, \tag{3.6}$$

i.e., the reparametrization maps the GLL points to the interpolation points. The left hand side of equation (3.6) then represents the dot product of a tangent vector to the numerical approximation and a normal vector to the exact curve at the interpolation points. Hence, by solving the system (3.5) for ξ_j^*, $j = 1, \dots, N-1$, we are implicitly finding a reparametrization of the exact curve such that the exact curve $(\tilde{x}(\eta), \tilde{y}(\eta))$ and the numerical approximation $(x_N(\xi), y_N(\xi))$ have equal tangent directions at the internal GLL points ξ_j, $j = 1, \dots, N-1$; see Figure 6. We will refer to this method as the equal-tangent method.

The system (3.5) consists of $N-1$ nonlinear equations, which in general may not have a real solution. However, it has been conjectured [13, 15] that for a given parametrized curve, there always exists an interpolant that satisfies certain types of additional conditions, as long as the number of equations does not exceed the number of degrees-of-freedom (here, $N-1$). The conjecture applies to the system (3.5), and we have yet to see a counter-example. For an overview of the conjecture, see also [6].

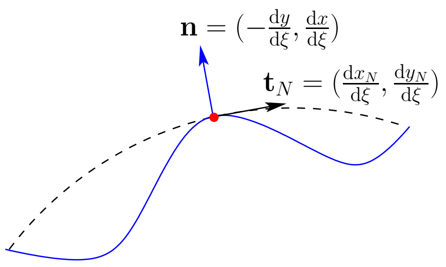

$\mathbf{n} = (-\frac{\mathrm{d}y}{\mathrm{d}\xi}, \frac{\mathrm{d}x}{\mathrm{d}\xi})$
$\mathbf{t}_N = (\frac{\mathrm{d}x_N}{\mathrm{d}\xi}, \frac{\mathrm{d}y_N}{\mathrm{d}\xi})$

Figure 6: The solid (blue) analytical curve and the dashed (black) numerical curve with associated normal and tangent vectors. Here, $N = 2$. At the internal interpolation points (only one in this example), we require that the analytical normal vector, $\mathbf{n}$, and the numerically computed tangent vector, $\mathbf{t}_N$, are orthogonal.

3.6 Remarks on the implementation

We close this section with a few remarks related to the implementation of the various methods. These are included in order to better understand the details of the implementation, and in order to be successful in reproducing these results.

3.6.1 Error computation

We first comment on how the interpolation error (3.4) is computed numerically in these tests. First, the integrand needs to be transformed to an integral over the reference domain so that we can use Gauss quadrature. To this end, we use the mapping $x_N(\xi)$ which will, in general, be nonaffine. We then use GLL quadrature with $M + 1$ points; typically, we use $M = 3N$ in order to ensure that the quadrature error is subdominant the interpolation error.

3.6.2 Solving the nonlinear system of equations

The implementation of the L^2-method and the equal-tangent method necessitates the solution of nonlinear systems of equations. For both methods, solving these systems is challenging. In the present work, we have used a standard Newton iteration. The main difficulty has been to construct a sufficiently good initial guess for the Newton iteration. In order to improve the robustness, we have also limited the change in the solution per iteration by introducing a relaxation parameter.

The system of nonlinear equations resulting from minimizing the L^2-error is particularly hard to solve. The Newton iteration can easily get trapped into finding a local minimum, for which the corresponding point distribution is non-optimal.

The equal-tangent method does not correspond to solving a minimization problem. Instead the method finds a point distribution which (i) interpolates the exact curve and (ii) results in equal tangent vectors at the internal interpolation points. This method is also sensitive to the initial guess. In particular, the system (3.5) may have multiple solutions.

The way we have implemented both the L^2-method and the equal-tangent method is by first solving the problem for a low polynomial degree, and then successively increase the value of N using the solution achieved for $N - 1$ to produce an initial guess. Hence, this corresponds to a bootstrapping approach. This may not always be sufficient and we have therefore also used the point distribution corresponding to the chord method as an initial guess. Each of these initial guesses will result in a set of interpolation points corresponding to a polynomial degree N. In order to proceed to the next value $N + 1$, we start from whichever point distribution produces the smallest L^2-error for the polynomial degree N.

For the curves considered in this paper, the mapping $x(\xi)$ is always bijective, and so we expect $x_N(\xi)$ to be the same. However, non-injective solutions may be produced by the Newton iterations. Note that, even if

$$\xi_0^* < \xi_1^* < \ldots < \xi_N^* \tag{3.7}$$

is satisfied, $x_N(\xi)$ may still not be monotonic due to oscillations. Monotonicity is not strictly enforced in our implementation, but (3.7) is checked during the iterations. Together with the step length restriction, this has proven to be enough in practice.

4 Numerical solution of the Poisson problem

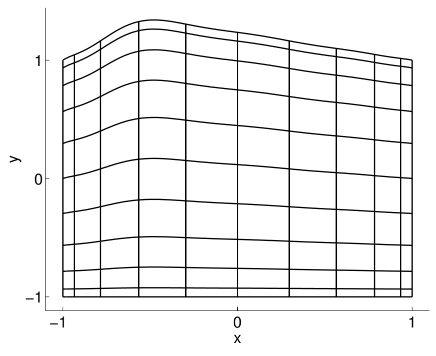
In this section we revisit the Laplace problem (2.1) from Section 2, but for different domains Ω. We will also consider a few Poisson problems with different exact solutions $u(x, y)$.

4.1 Case 2

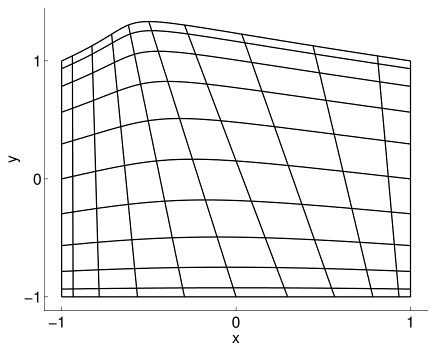
Consider a deformed quadrilateral domain with three straight edges and the top edge parametrized as

$$\begin{aligned} x(\xi) &= \frac{1}{2}(\xi^3 + \xi^2 + \xi - 1), \\ y(\xi) &= 1 + \frac{1}{3}(1 - \xi^2), \end{aligned} \tag{4.1}$$

where $\xi \in [-1, 1]$. The top edge can thus be parametrized by low order polynomials, but the corresponding function representation $y(x)$ is complicated and contains several root functions. Hence, a standard mesh, featuring vertical gridlines in the interior, will not be optimal. Figure 7 shows this mesh, along with the mesh generated by the equal-tangent method.



(a) The standard method



(b) The equal-tangent method

Figure 7: Computational grids in Case 2, using a polynomial degree $N = 10$.

Figure 8 shows the relative error in the numerical solution of the Laplace problem (2.1). The exact solution $u(x, y) = e^x \sin y$ is analytic, and we reach machine precision around $N = 25$ using the equal-tangent mesh. On the other hand, the standard mesh yields errors in the geometry representation that dominate the error in the representation of the solution of the Laplace problem, and the error (2.2) decreases much more slowly.

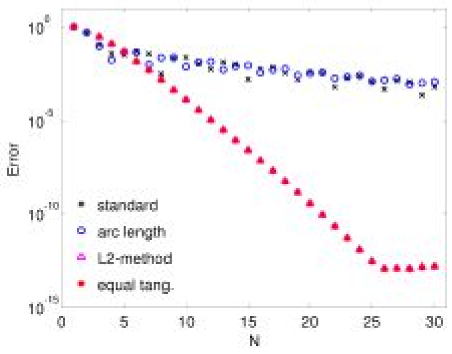


Figure 8: Relative error measured in the discrete energy norm when solving the Laplace problem (2.1) in the domain depicted in Figure 7.

Next, we solve a Poisson problem with a low order polynomial solution over the same domain. The exact solution is given as $u(x, y) = x^3 + y^3$, and we impose corresponding non-homogeneous Dirichlet boundary conditions. With the equal-tangent method we get a mapping where x_N is a polynomial of degree 3 and y_N is a polynomial of degree 2 in ξ, respectively, and hence u is a polynomial of degree 9 in ξ. Unless we introduce any quadrature error, we should get the exact solution for $N \geq 9$. Figure 9 confirms this.

4.2 Case 3

In this case we solve the same two problems as in the previous case, but now investigate the effect of having a domain with four deformed edges where all the edges can be parametrized by low order polynomials. In particular, Ω is a deformed quadrilateral where the top edge is described by the curve (4.1), whereas the other edges are modifications of this curve. All edges are parametrized by polynomials of degree less than or equal to 5, so exact representation of the geometry can be achieved at $N = 5$ by direct interpolation of the given parametric

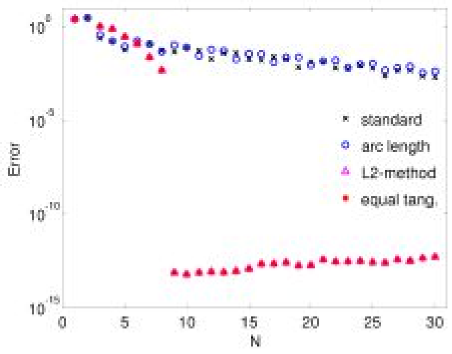


Figure 9: Relative error measured in the discrete energy norm when solving the Poisson equation with an exact polynomial solution $u(x, y) = x^3 + y^3$ in the domain depicted in Figure 7.

functions. If there exist reparametrizations that only consist of polynomials of even lower degree, we may see even faster convergence.

The meshes generated by the standard method and the equal-tangent method are shown in Figure 10. Figure 11 shows very similar behavior to what we saw in Case 2, though the convergence rate is slightly lower. From Figure 12 and an argument similar to that in Case 2 we conclude that exact representation of the geometry is achieved at $N = 5$ by the L^2-method and the equal-tangent method.

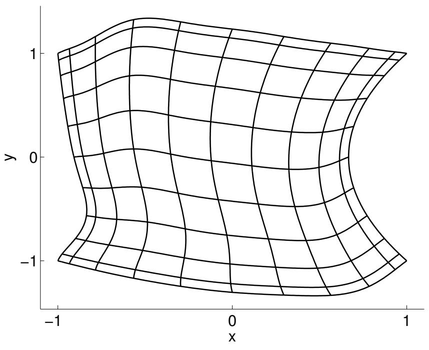
4.3 Case 4

Consider a deformed quadrilateral domain with three straight edges and the top edge described by the function

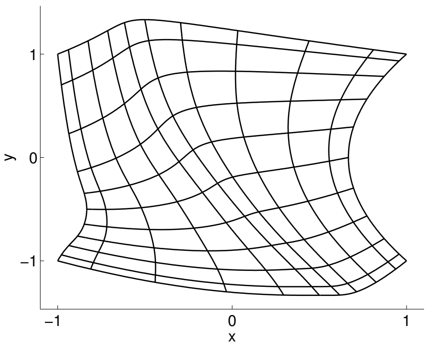
$$y^T(x) = 1 + \frac{1}{10}(1 - |x|). \tag{4.2}$$

The equal-tangent method gives a clustering of interpolation points around $x = 0$ on the top edge, which is due to a strongly nonaffine mapping. This gives a significant distortion of the mesh compared to the mesh generated by the standard method (vertical grid lines); see Figure 13.

Figure 14 shows the relative error in the numerical solution of the Laplace problem (2.1), measured in the energy norm. The standard method yields al-



(a) The standard method



(b) The equal-tangent method

Figure 10: Computational grids in Case 3, using a polynomial degree $N = 10$.

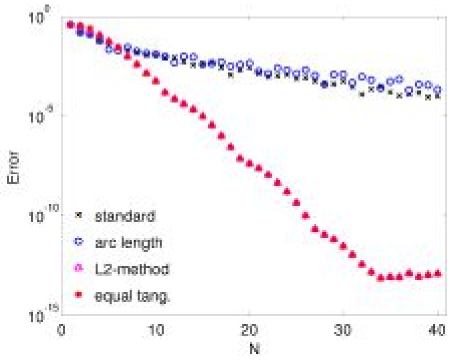


Figure 11: The relative error in the solution of a Laplace problem with an exact solution $u(x, y) = e^x \sin y$ and a domain shown in Figure 10 (Case 3); the error is measured in the discrete energy norm.

gebraic convergence due to the low order approximation of the geometry. The equal-tangent method yields exponential convergence for sufficiently large N; for low values of N there is a trade-off between the representation of the geometry and representation of the solution to the Laplace problem. Note that, even if the exact solution is very regular on the physical domain, it could exhibit steep gradients and boundary layers on the reference domain due to the strongly

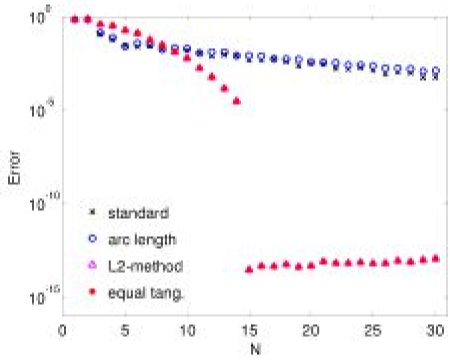
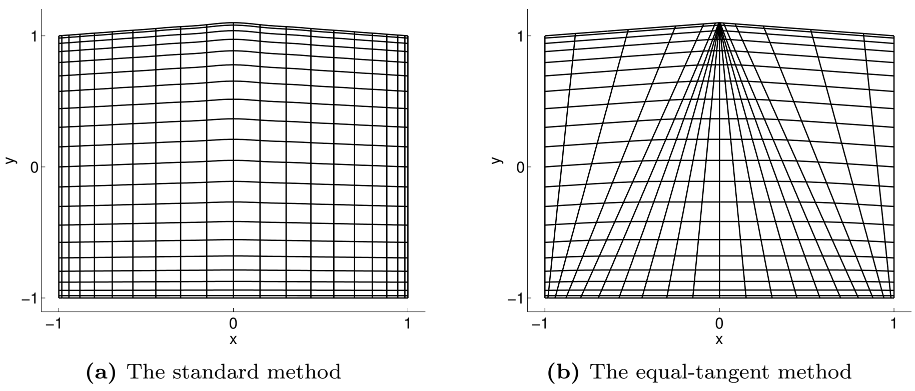


Figure 12: The relative error in the solution of a Poisson problem with an exact solution $u(x, y) = x^3 + y^3$ and a domain shown in Figure 10 (Case 3); the error is measured in the discrete energy norm.



(a) The standard method

(b) The equal-tangent method

Figure 13: Computational grids in Case 4 using a polynomial degree $N = 20$.

nonlinear mapping.

4.4 Case 5

In the final example, we consider a domain where all the four edges are deformed; see Figure 15. The top edge is part of a circle of radius one, the left edge and

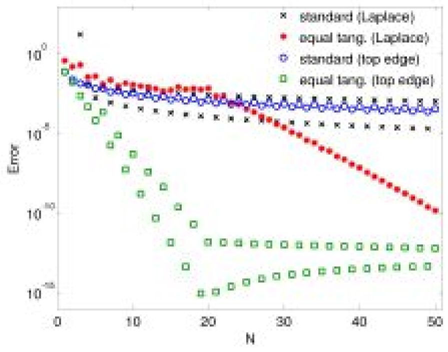


Figure 14: The relative error in the solution of the Laplace problem (2.1) in the domain depicted in Figure 13; the error is measured in the discrete energy norm. Only results for the standard method and the equal-tangent method are shown. For comparison, we also show the L^2-error in the approximation of the top edge.

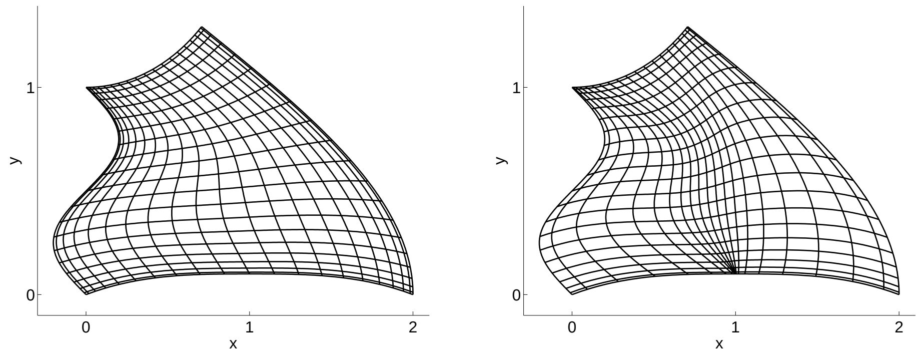
the right edge are trigonometric functions of y, and the bottom edge can be described by the function $y^B(x) = \frac{1}{10}(1 - |x|^3)$.

The left grid in Figure 15 represents a typical way to construct a grid, with the bottom edge and the left and the right edges approximated using the chord method, and the top edge approximated using the arc length method. The right grid in Figure 15 is fully based on using the equal-tangent method for the approximation of the domain boundary. The most notable feature is the clustering of grid points towards the center of the bottom edge; this is due to the finite regularity of $y^B(x)$.

In Figure 16 we compare the discretization error (2.2) as a function of N for the two different grids depicted in Figure 15. Similar to Case 4, the equal-tangent method is not optimal for small N. Again, this is because the clustering of grid points along the bottom edge strongly affects the mapping from the two-dimensional reference domain to the physical domain.

5 Conclusions and final remarks

In this study we have investigated the impact of the choice of geometry representation in the context of solving partial differential equations using spectral methods. We have shown that the overall discretization error for the Poisson



(a) The chord method / arc length method **(b)** The equal-tangent method

Figure 15: Computational grids in Case 5, using a polynomial degree $N = 20$.

problem can sometimes be significantly reduced by a careful representation of the boundary of the domain (in our case, the four edges of a deformed quadrilateral domain).

Each edge of the two-dimensional domain is approximated using high order polynomial interpolation. Once the boundary of the domain has been approximated, the mapping between the reference domain and the physical domain follows readily. The numerical results presented in this study show the importance of, and the sensitivity to, the choice of interpolation points along a curve in the plane. We have also proposed a method (the equal-tangent method) as a way to ensure close to optimal interpolation results. The advantage of this method compared to directly finding an L^2-optimal interpolant is that it is computationally less difficult to solve the nonlinear system of equations. The new method can give significantly faster convergence compared to more standard interpolation methods. For example, approximating a deformation in the form of the classical Runge function using the new method yields a much faster convergence rate compared to classical interpolation. The most extreme case is exponential convergence obtained for a problem with an analytic solution in a domain where the boundary has a finite (low) regularity.

One can easily think of other methods for interpolating a parametrized curve than the ones presented here. The points can be chosen by distributing them according to a weight function along the curve, or along the x-axis, as was done in [5] for construction of optimal finite element meshes. This approach can

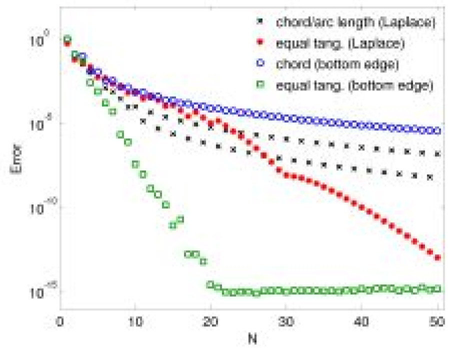


Figure 16: The error in the solution of the Laplace problem (2.1), for the domain depicted in Figure 15, measured in the energy norm. For comparison, we also show the L^2-error in the approximation of the bottom edge. The algebraic convergence of the standard approach is due to the inability to represent the bottom edge (a function of low regularity) accurately. In contrast, the equal-tangent method is able to represent the geometry to machine precision at $N = 20$, at the cost of a highly nonlinear mapping. This is not ideal for representing the field variables on the reference domain, and the method is clearly not optimal for small N. However, for large N the equal-tangent method is able to achieve exponential convergence.

in principle be applied to high order methods, but the analysis that ensures optimality no longer holds. New methods can also be constructed by replacing the equal-tangent conditions by $N-1$ other conditions. One can for example require equal tangents *and* equal curvature at $\lfloor (N-1)/2 \rfloor$ points, or one can reduce the number of points and increase the order of contact even further. According to the conjectures [13, 15], this should give results similar to those seen here using the equal-tangent method. Finally, we can use the $N-1$ degrees-of-freedom to ensure interpolation in a set of $N-1$ *extra interpolation points* such that we have a total of $2N$ interpolation points. Numerical experiments yield varying results, with the best results achieved when the extra set of points are required to be very close to the existing internal interpolation points. This approach gives a very similar behavior as the equal-tangent method.

The extension of the mesh from the boundary of the domain to the interior can also be done in several different ways. Instead of using the Gordon-Hall algorithm one can solve an elliptic PDE, but in our experience the potential gain is very limited.

Future work will focus on extending the current study to other types of partial differential equations and to solving three-dimensional problems. Finding optimal high order representations of curved surfaces in three dimensions is an unsolved problem. Furthermore, it would be of interest to investigate to what extent the choice of interpolation points can be used to construct adaptive algorithms for high order discretizations.

Acknowledgment

The work has been supported by the Research Council of Norway under contract 185336/V30. The support is gratefully acknowledged.

Bibliography

[1] C. Bernardi and Y. Maday. Spectral methods. *In: P. G. Ciarlet, J. L. Lions (eds.), Handbook of Numerical Analysis, Volume V: Techniques of Scientific Computing (Part 2)*, pages 209–485, 1997.

[2] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 45(3):501–517, 2004.

[3] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods, Fundamentals in Single Domains.* Springer, 2006.

[4] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods, Evolution to Complex Geometries and Applications to Fluid Dynamics.* Springer, 2007.

[5] G. F. Carey and H. T. Dinh. Grading functions and mesh redistribution. *SIAM J. Numer. Anal.*, 22(5):1028–1040, 1985.

[6] W. L. F. Degen. Geometric Hermite interpolation – in memoriam Josef Hoschek. *Comput. Aided Geom. Design*, 22(7):573–592, 2005.

[7] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flow.* Cambridge University Press, 2002.

[8] B. Fornberg. *A Practical Guide to Pseudospectral Methods.* Cambridge University Press, 1998.

[9] W. J. Gordon and C. A. Hall. Construction of curvilinear co-ordinate systems and applications to mesh generation. *Internat. J. Numer. Methods Engrg.*, 7(4):461–477, 1973.

[10] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods.* Springer, 2008.

[11] G. E. Karniadakis and S. J. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics.* Oxford University Press, 2nd edition, 2005.

[12] Y. Maday and A. T. Patera. Spectral element methods for the Navier-Stokes equations. *In: A. K. Noor, J. T. Oden (eds.), State of the Art Surveys in Computational Mechanics, ASME, New York*, pages 71–143, 1989.

[13] K. Mørken and K. Scherer. A general framework for high-accuracy parametric interpolation. *Math. Comput.*, 66(217):237–260, 1997.

[14] B. O'Neill. *Elementary Differential Geometry.* Academic Press, 2006.

[15] A. Rababah. High order approximation method for curves. *Comput. Aided Geom. Design*, 12(1):89–102, 1995.

PAPER 2

HIGH ORDER POLYNOMIAL INTERPOLATION OF PARAMETERIZED CURVES

TORMOD BJØNTEGAARD, EINAR M. RØNQUIST AND ØYSTEIN TRÅSDAHL

Published in
Lecture Notes in Computational Science and Engineering,
Vol. 76 (2011), pp. 365–372

HIGH ORDER POLYNOMIAL INTERPOLATION OF PARAMETERIZED CURVES

TORMOD BJØNTEGAARD, EINAR M. RØNQUIST AND ØYSTEIN TRÅSDAHL

Department of Mathematical Sciences,
Norwegian University of Science and Technology,
Trondheim, Norway

Abstract

Interpolation of parameterized curves differs from classical interpolation in that we interpolate each spatial variable separately. A difficult challenge arises from the option of *reparameterization*: a presumably good interpolation (e.g., at the Gauss points) of a given parameterization does not necessarily give the best approximation of the curve, as there may exist a reparameterization better suited for polynomial interpolation. The reparameterization can be done implicitly by choosing different sets of interpolation points along the exact curve. We present common interpolation methods, and propose a new method, based on choosing the interpolation points in such a way that the interpolant is tangential to the exact (reparameterized) curve at these points. The new method is compared to the traditional ones in a series of numerical examples, and results show that classical interpolation is sometimes far from optimal in the sense of the Kolmogorov n-width, i.e., the best approximation using n degrees-of-freedom.

1 Introduction

The topic of polynomial interpolation of parameterized curves appears in practical applications in high order methods for solving partial differential equations in deformed domains [3, 4]. The accuracy of the numerical solution is directly influenced by the accuracy of the geometry representation [7]. If the distortions are not too large, this representation can readily be achieved via a Gordon-Hall transfinite interpolation procedure [6]. For a deformed quadrilateral domain, this algorithm requires that we first construct an accurate representation

$$(x_N(\xi), y_N(\xi)), \qquad x_N, y_N \in \mathbb{P}_N(-1,1) \tag{1.1}$$

of each of the four boundary curves. This is merely an *approximation* of the exact curve, and an easy way to achieve a good approximation is through interpolation. Then the approximation problem simplifies to the problem of choosing a set of interpolation points.

In this paper we explore different ways of choosing these interpolation points. We compare previously proposed methods with a new method. The methods will be introduced in the context of plane curves, and then later extended to space curves. The accuracy of the different interpolation methods will be compared in numerical experiments.

2 Interpolation methods for plane curves

The starting point is a given curve $y(x)$ in the plane, defined by the parameterization

$$(x(\eta), y(\eta)), \qquad \eta \in [-1, 1]. \tag{2.1}$$

We assume that $y(x)$ is C^1, so that there is a unique tangent vector at each point on the curve. Our numerical approximation is an interpolant based on a representation by high order polynomials (1.1). A nodal basis for the polynomial $x_N(\xi)$ is

$$x_N(\xi) = \sum_{j=0}^{N} x_j \, \ell_j(\xi), \tag{2.2}$$

and similarly for $y_N(\xi)$. Here, $\ell_j(\xi)$ is the j'th Lagrangian interpolant through the Gauss-Lobatto-Legendre (GLL) points $\xi_i, i = 0, \ldots, N$, with the property that $\ell_j(\xi_i) = \delta_{ij}$. Hence, the expansion coefficients x_j and y_j are coordinates somewhere on the exact curve, i.e. $x_j = x(\eta_j)$ and $y_j = y(\eta_j)$ for some $\eta_j \in [-1, 1]$. We impose the restriction that the two end points of the numerical curve are interpolation points, i.e., $\eta_0 = -1$ and $\eta_N = 1$. However, we do not require the *internal* interpolation points η_j, $j = 1, \ldots, N-1$ to be the internal GLL points, as there always exists a *reparameterization* $(\tilde{x}(\xi), \tilde{y}(\xi))$, $\xi \in [-1, 1]$, such that the interpolation points are mapped from the GLL points in the reference domain, i.e., $x_j = \tilde{x}(\xi_j)$ and $y_j = \tilde{y}(\xi_j)$. The two parameterizations are connected by the relationship

$$\tilde{x}(\xi) = x(\eta(\xi)) = x\big(\sum_{j=0}^{N} \eta_j \ell_j(\xi)\big), \tag{2.3}$$

and correspondingly for y. The reparameterization is not unique [8], and we have here chosen η to be a polynomial of degree N in ξ. Equation (2.3) *implicitly* defines the reparameterization from the choice of interpolation points.

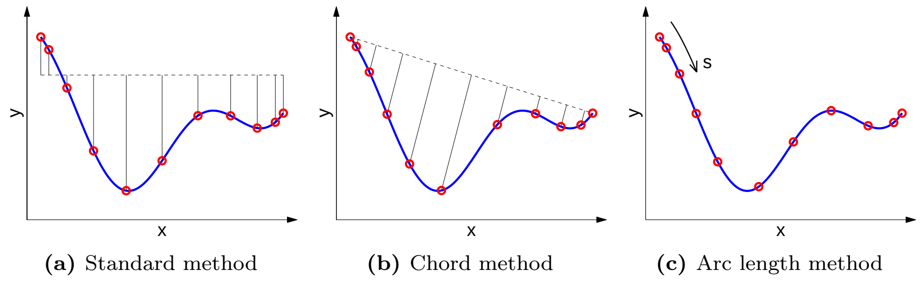
There are some widely known methods for choosing the values η_j. We will first describe them briefly, and then introduce two alternative methods.

2.1 Common interpolation methods

The three most common interpolation methods all rely in some way on an affine mapping of the GLL points from the reference domain to the physical domain [4]. The first, which we will refer to as the *standard method*, uses an affine mapping $x_N(\xi)$ such that the interpolation points are distributed according to a GLL distribution along the x-axis. This implies that y_N will not only be a polynomial as a function of ξ, but also as a function of x_N.

We can also choose a GLL distribution along the chord between the two end points of the curve; see Figure 1. This is the *chord method*, which coincides with the standard method when the chord is parallel to the x-axis.

The last method is based on a GLL distribution in the arc length variable s, and is called the *arc length method*.



(a) Standard method **(b)** Chord method **(c)** Arc length method

Figure 1: Three common methods for choosing interpolation points.

2.2 The L^2-method

The three previous methods each have special types of curves where they work well. However, we do not know how good the resulting interpolants are compared to the best possible interpolant.

In order to be able to define an optimal interpolant, we restrict our study to curves that can be described by a function $y(x)$ for $x \in [a,b]$. Then the L^2-norm can be used to measure the interpolation error, and we define the optimal set $\{\eta_j\}_{j=1}^{N-1}$ of internal interpolation points to be the one that minimizes the functional

$$\mathcal{J} = ||y - y_N||_{L^2}^2 = \int_a^b (y(x) - y_N(x))^2 \, \mathrm{d}x. \qquad (2.4)$$

We can differentiate $\mathcal{J}$ with respect to each independent variable η_j, $j = 1, \ldots, N-1$, and use Newton's method to search for the minimum. We will refer to this method as the L^2-method; see [2] for more details. The resulting minimizer can be viewed as the solution to the Kolmogorov n-width problem applied to the interpolation of curves. Note that we are searching for the *global* minimum of (2.4). Newton's method uses a local search, and is therefore dependent on a good initial guess.

In general, we are not guaranteed that $x_N(\xi)$ is invertible (i.e., that $y_N(x_N)$ is a function), but this does not seem to be a big practical problem.

2.3 The equal-tangent method

The functional $\mathcal{J}$ uses information about the curves on the entire interval $[a,b]$, which makes the L^2-method slow and complicated as N increases. We therefore propose a method which uses information about the curves only at the interpolation points. The idea behind the equal-tangent method is to require that the exact and numerical curves are tangential at the $N-1$ internal interpolation points. This can be achieved if we are able to find the roots $\eta_1, \eta_2, \ldots, \eta_{N-1}$ of the nonlinear system

$$\frac{\mathrm{d}x_N}{\mathrm{d}\xi}(\xi_j)\frac{\mathrm{d}y}{\mathrm{d}\eta}(\eta_j) - \frac{\mathrm{d}y_N}{\mathrm{d}\xi}(\xi_j)\frac{\mathrm{d}x}{\mathrm{d}\eta}(\eta_j) = 0, \qquad j = 1, \ldots, N-1. \qquad (2.5)$$

The left hand side represents an inner product between a tangent vector to the interpolant and a normal vector to the exact curve. In order to solve this system of equations, we will apply a Newton method. This requires that we differentiate the left hand side of (2.5) with respect to the $N-1$ independent variables η_j at the internal interpolation points.

We remark that the solution of (2.5) may not be unique; in such cases the particular solution obtained will depend on the initial guess. The existence of a solution in the general case has not been proven, however we have not yet encountered a counter-example. The method works well on a wide range of curves, and we will show a few examples here; see [2] for more details.

2.4 Numerical results

The following examples are chosen to illustrate the behavior of the various methods in different situations. They are all given as functions $y(x)$; a parameterization (2.1) is readily achieved, using an affine mapping $x(\eta)$.

The interpolation error is measured in the discrete L^2-norm, where the integral in (2.4) is evaluated using GLL quadrature [2].

Case 1 The first example we consider is described by the function $y(x) = \frac{1}{1+16x^2}$, $x \in [-1, 1]$. Classical interpolation theory tells us that this function is particularly difficult to interpolate [5], and as the standard method yields a polynomial $y_N(x_N)$, we expect it to converge very slowly. Figure 2a confirms this, and it shows that the arc length method is even worse. Compared to this, the convergence rate of our proposed method is striking. By construction, the L^2-method is supposed to be the best, but it is only best for $N < 9$; as mentioned earlier, this is due to the complexity of computing the global minimizer of (2.4) as N increases.

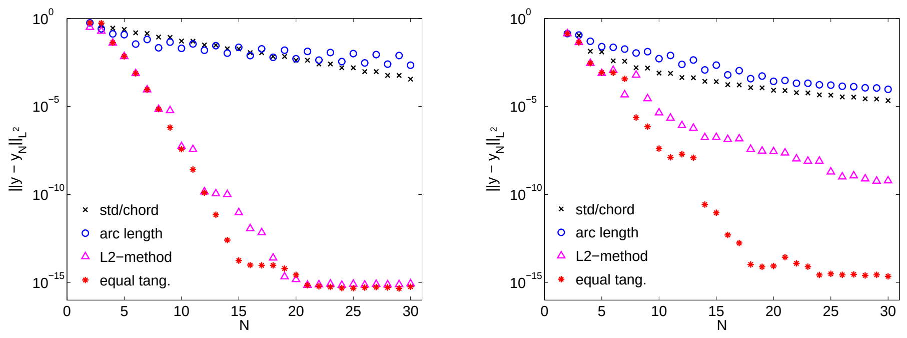
Case 2 From classical interpolation theory we know that approximation of functions of limited regularity with polynomials results in algebraic convergence [1]. Consider the function $y(x) = 1 - |x|^3$, defined on $x \in [-1, 1]$. Both the standard method and the arc length method converge algebraically. The equal-tangent method, however, converges exponentially; see Figure 2b. The L^2-method again converges fast only up to a certain value of N.

3 Interpolation of space curves

We now consider curves in space, defined by a given parameterization

$$(x(\eta), y(\eta), z(\eta)), \qquad \eta \in [-1, 1]. \tag{3.1}$$

In order to be able to compare all methods, we restrict ourselves to curves where both y and z can be described by functions of x. Then, the standard, chord and arc length methods can all be extended in a natural way.



(a) Case 1 **(b)** Case 2

Figure 2: Interpolation error, measured in the discrete L^2-norm. Left: for the Runge function, the standard method and the arc length method both converge slowly, but exponentially. The equal-tangent method and the L^2-method both converge much faster. Right: for this function of limited regularity, the standard method and the arc length method both yield algebraic convergence. The equal-tangent method, on the other hand, converges exponentially. The L^2-method performs reasonably well, but has difficulty finding the global minimum.

3.1 The L^2-method

With our restriction on curves, we can define a functional similar to (2.4), extended to include the error in the z-variable:

$$\mathcal{J} = \int_a^b \left[(y(x) - y_N(x))^2 + (z(x) - z_N(x))^2 \right] \, \mathrm{d}x. \tag{3.2}$$

The integral is transformed to the reference variable ξ and evaluated numerically using GLL quadrature. With this extension, everything is similar to the two-dimensional case, including the minimization procedure.

3.2 The equal-tangent method

The extension of the equal-tangent method is not as straightforward. In the plane, there is a unique normal vector to the curve, but in space there is a whole *normal plane*. Hence, one normal vector is not enough to ensure equal tangents. We propose a method where we use one normal vector from each coordinate plane,

$$\mathbf{n}_1 = \begin{bmatrix} 0 \\ -z'(\eta) \\ y'(\eta) \end{bmatrix}, \qquad \mathbf{n}_2 = \begin{bmatrix} z'(\eta) \\ 0 \\ -x'(\eta) \end{bmatrix}, \qquad \mathbf{n}_3 = \begin{bmatrix} -y'(\eta) \\ x'(\eta) \\ 0 \end{bmatrix}. \tag{3.3}$$

This is one more than we need to span the normal plane, but it gives symmetry in the space variables. Numerical experiments indicate that this may add to the robustness of the method. In order to realize the condition of orthogonality for all the three normal vectors, we *square* the inner products and take the sum

$$\sum_{i=1}^{3} (\mathbf{t}_N \cdot \mathbf{n}_i)^2 = 0, \tag{3.4}$$

where $\mathbf{t}_N = (x_N'(\xi), y_N'(\xi), z_N'(\xi))^T$. Newton's method applied to (3.4) do not result in the same set of equations as Newton's method applied to (2.5) for curves in the plane ($z(\eta) = 0$). However, both systems have the same sets of exact solutions.

3.3 Numerical results

Case 3 The curve we are looking at is a distorted helix, spiraling along the x-axis with a varying radius. It is defined by the parameterization

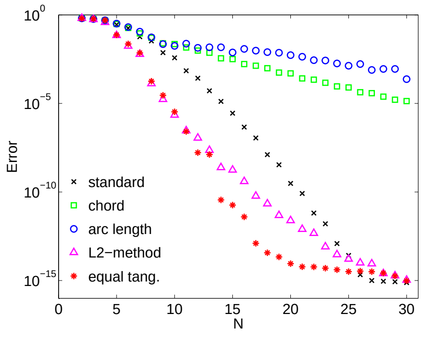
$$\begin{aligned} x(\eta) &= -\frac{5}{2} + \frac{7}{4}(\eta + 1), \\ y(\eta) &= \frac{1}{2} e^{-(1+\eta)} \cos(2\pi\eta), \\ z(\eta) &= \frac{1}{8}(\eta + 2)\sin(2\pi\eta), \end{aligned}$$

for $\eta \in [-1, 1]$. Figure 3a shows that the situation is much the same as it was in two dimensions: the equal-tangent method is the best, and it almost coincides with the L^2-method up to $N = 11$. The standard method works well in this case due to the construction of the example, while the chord and arc length methods converge very slowly.

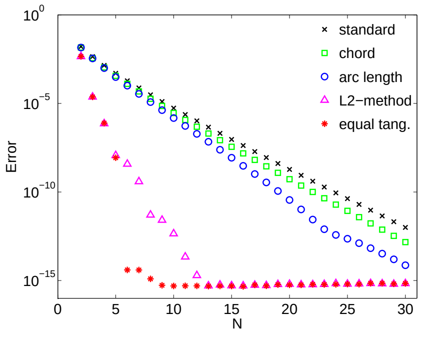
Case 4 Consider the curve parameterized by

$$x(\eta) = \eta + 1,$$
$$y(\eta) = \sqrt{\left(\eta + \frac{9}{4}\right)^{1/3} - 1},$$
$$z(\eta) = \left(\eta + \frac{9}{4}\right)^{2/3} - 1.$$

If we let $\eta(\xi) = ((\alpha\xi + \beta)^2 + 1)^3 - 9/4$ for suitable constants α and β, we get a reparameterization where $\tilde{y}(\xi)$ is affine and $\tilde{x}(\xi)$ and $\tilde{z}(\xi)$ are polynomials of degrees 6 and 4, respectively. Hence, the best distribution of interpolation points should give an *exact* representation of the curve from $N = 6$. Figure 3b shows that the equal-tangent method indeed finds this optimal solution, with no a priori knowledge of the optimal distribution of interpolation points.



(a) Case 3



(b) Case 4

Figure 3: Interpolation error, defined as the square root of (3.2). Left: the curve is well suited for the standard method, but we are still able to achieve faster convergence with the equal-tangent method and the L^2-method. Right: the parameterization consists of square and cubic roots, which makes the standard method a non-optimal choice. In particular, the curve can be reparameterized using polynomials of degree less than or equal to 6, which is detected by the equal-tangent method.

4 Conclusions and future work

We have looked at interpolation of parameterized plane and space curves using high order polynomials. We have proposed a new method, iterative in nature, based on a requirement that the interpolant be tangential to the exact curve at the internal interpolation points. Through numerical experiments we show that the new method can give significantly smaller error than the conventional methods, and we believe it yields results that are close to optimal in the sense of the Kolmogorov n-width, i.e., the best approximation using n degrees-of-freedom. The most extreme case is exponential convergence obtained for a function $y(x)$ with finite regularity.

The motivation behind this study has been the numerical solution of partial differential equations in deformed domains using high order methods. The new method can be applied to the representation of the domain boundary, which affects the error of the resulting numerical solution. The preliminary results are promising, and reported in a separate article [2].

Future work will focus on the representation of surfaces in space, which can then be applied to the numerical solution of PDEs in deformed three-dimensional domains.

Acknowledgment

The work has been supported by the Research Council of Norway under contract 185336/V30. The support is gratefully acknowledged.

Bibliography

[1] C. Bernardi and Y. Maday. Spectral methods. *In: P. G. Ciarlet, J. L. Lions (eds.), Handbook of Numerical Analysis, Volume V: Techniques of Scientific Computing (Part 2)*, pages 209–485, 1997.

[2] T. Bjøntegaard and E. M. Rønquist. Accurate interface-tracking for arbitrary Lagrangian-Eulerian schemes. *J. Comput. Phys.*, 228(12):4379–4399, 2009.

[3] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods, Evolution to Complex Geometries and Applications to Fluid Dynamics.* Springer, 2007.

[4] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flow.* Cambridge University Press, 2002.

[5] B. Fornberg. *A Practical Guide to Pseudospectral Methods.* Cambridge University Press, 1998.

[6] W. J. Gordon and C. A. Hall. Construction of curvilinear co-ordinate systems and applications to mesh generation. *Internat. J. Numer. Methods Engrg.*, 7(4):461–477, 1973.

[7] Y. Maday and E. M. Rønquist. Optimal error analysis of spectral methods with emphasis on non-constant coefficients and deformed geometries. *Comput. Methods Appl. Mech. Engrg.*, 80(1-3):91–115, 1990.

[8] B. O'Neill. *Elementary Differential Geometry.* Academic Press, 2006.

PAPER 3

HIGH ORDER INTERPOLATION OF PARAMETRIC CURVES AND SURFACES IN $\mathbb{R}^3$

ØYSTEIN TRÅSDAHL

NTNU Preprint Numerics No. 5/2011.
Submitted to Advances in Computational Mathematics

HIGH ORDER INTERPOLATION OF PARAMETRIC CURVES AND SURFACES IN $\mathbb{R}^3$

ØYSTEIN TRÅSDAHL

Department of Mathematical Sciences,
Norwegian University of Science and Technology,
Trondheim, Norway

Abstract

In this paper, high order interpolation of parametric curves and surfaces in $\mathbb{R}^3$ is studied. The topic differs from classical interpolation of functions since any reparametrization of the given curve or surface can be interpolated. This leads to the question whether there exists an optimal reparametrization that results in the lowest possible interpolation error. This can also be viewed as a Kolmogorov n-width problem in terms of polynomial interpolation: how to best use the available degrees-of-freedom in order to minimize the interpolation error. Here, this problem is studied numerically, and different interpolation methods are presented and compared. The methods are introduced in the context of parametric curves and then extended to parametric surfaces when possible. The results are relevant for numerical solution of PDEs using high order methods.

Keywords: Geometric Hermite interpolation, reparametrization, high order polynomials, geometric continuity

Mathematics Subject Classifications: 41A10, 41A25, 65D05, 65D17, 65N35, 65N50

1 Introduction

Polynomial interpolation of parametric curves and surfaces is a central part of Computer Aided Geometric Design (CAGD). The traditional way to interpolate a given parametric curve $\boldsymbol{f}$ in $\mathbb{R}^d$ is to view it as a vector-valued *function* and interpolate each of the d components separately. Polynomials of degree N can be made to interpolate the curve in $N+1$ points in this way. If $\boldsymbol{f}$ is in C^k (i.e., each component is a C^k function), one can also choose to interpolate fewer points and instead match both function values and derivatives in the interpolation points. A polynomial of degree N can interpolate a function and its k first derivatives at n points if $N = n(k+1) - 1$. Common for both approaches is that the interpolant can be constructed by solving systems of *linear* equations, and that the approximation order (as defined for approximation of functions) is $N + 1$. As an example, consider cubic spline curves in $\mathbb{R}^2$ which can be constructed to interpolate function values and derivatives at the end points of each curve segment, giving the approximation order four.

The parametrization of a curve can be thought of as the position vector for a particle traversing the curve. This implies that the first derivative of $\boldsymbol{f}$ describes its velocity, the second derivative the acceleration and so forth. If our goal is to approximate the curve as a geometric object, these quantities are of little interest. Instead we are interested in geometric properties of the curve, such as tangent directions, curvature and torsion. It is possible to construct interpolants based on such quantities, and it can yield a higher approximation order than classical interpolation. In [6] it was shown that under certain conditions, cubic polynomial curves in $\mathbb{R}^2$ can interpolate both function values, tangent directions *and* curvature at the end points, resulting in approximation order six. The price to pay for the increased accuracy is a system of non-linear equations that must be solved. The interpolation method was viewed as a generalization of Hermite interpolation based on geometric quantities and was therefore called *geometric Hermite interpolation*.

In recent years we have seen a lot of work on geometric Hermite interpolation in the CAGD community; e.g., see [7, 8, 9, 11, 13, 17, 22, 24, 27]. The work has led to a conjecture [15] about the highest possible approximation order that can be attained when interpolating parametric curves in $\mathbb{R}^d$ by polynomials of degree N. The conjecture has been confirmed in some special cases, but it remains unproven. Most of the authors focus on planar curves, but there has also been some work done on curves in $\mathbb{R}^3$: cubic interpolation was studied in [12, 15], quartic in [5, 26], and quintic in [25].

The concept of geometric Hermite interpolation can also be applied in the context of parametric surfaces, but the problem is much harder due to the increased number of unknowns. Mørken [19] gives a detailed discussion of the optimal approximation order and constructs a quadratic Taylor approximant with approximation order four. Lagrange interpolation of surfaces with quadratic polynomials is considered in [16].

There has been surprisingly little work done on *high order* interpolation of parametric curves and surfaces. In the field of CAGD polynomials of degree $N > 5$ are not so common in applications. However, the subject is relevant in the context of solving PDEs in deformed geometries using high order methods [4, 10]. Here, the accuracy of the numerical solution is directly influenced by the accuracy of the geometry representation [18]. It is common to use an *isoparametric approach*, representing the geometry with the same polynomial degree as the other field variables. For example, in a Legendre spectral element method, deformed quadrilaterals or hexahedrons are approximated by tensor-product polynomials, constructed by interpolating the exact geometry. In this context a reparametrization may yield a better representation of not only the geometry, but also the primary field variables [2]. Still, the topic has not been given much attention in the literature.

In the context of high order interpolation the concept of approximation order is not commonly used since the approximation approach is global and we only use one polynomial curve segment for the entire curve (as opposed to for example a spline approach). Convergence is rather defined in terms of how the interpolation error, measured in some appropriate norm, decreases as the polynomial degree N increases. It is well known from classical interpolation theory that smooth functions can be interpolated by polynomials to exponential convergence, i.e., the interpolation error decreases faster than any algebraic power of N [3]. An optimal interpolation method may thus be defined as one that yields exponential convergence with the highest possible rate, i.e., with the largest possible negative exponent. Functions of finite regularity yield algebraic convergence in classical interpolation, but as we will see, the choice of interpolation points *implicitly defines* a reparametrization, which is the function that is actually being interpolated. A good interpolation method may give us exponential convergence, even if the given parametrization is a function of low regularity.

The outline of the paper is a follows. In Section 2 we first present the framework for polynomial interpolation of parametric curves and discuss how the option of reparametrization makes the subject different from classical interpolation. We then present two interpolation methods that are commonly used in the high order methods community, and we introduce three new methods: one optimization method aimed at directly minimizing the interpolation error, and two methods in the family of geometric Hermite interpolation. In Section 3 we compare the different methods through several numerical examples. In Section 4 we discuss how to extend the methods to interpolation of parametric surfaces, and some numerical examples are presented in Section 5. The conclusions of this study are summarized in Section 6.

2 Interpolation of parametric curves

Consider a curve $\mathscr{C}$ in $\mathbb{R}^3$, defined by a given parametrization

$$\boldsymbol{f}(\eta) = (f_1(\eta), f_2(\eta), f_3(\eta)), \qquad \eta \in [-1, 1]. \tag{2.1}$$

The curve is C^k-continuous if each of the parametric functions f_i, $i = 1, 2, 3$, are in C^k. The problem we set out to solve is how to best interpolate this curve by polynomials, i.e., a parametric curve

$$\boldsymbol{p}(\xi) = (p_1(\xi), p_2(\xi), p_3(\xi)), \qquad \xi \in [-1, 1], \tag{2.2}$$

where p_i, $i = 1, 2, 3$ are functions in $\mathbb{P}_N([-1, 1])$, the (discrete) space of polynomials of degree less than or equal to N. This problem is different from classical polynomial approximation of functions since $\mathscr{C}$ can be *reparametrized.* Specifically, for all $\varphi \in W = \{\psi \in C^\infty([-1, 1]) \mid \psi(\pm 1) = \pm 1, \text{ and } \psi' > 0\}$ the function

$$\boldsymbol{g}(\xi) = \boldsymbol{f}(\varphi(\xi)) \tag{2.3}$$

describes the same curve, so interpolation of $\boldsymbol{g}$ instead of $\boldsymbol{f}$ gives an approximation of the same geometric object. Intuitively, a reparametrization means traversing the curve at a different speed. There exist infinitely many reparametrizations of any given curve, and some may be better suited for polynomial interpolation than others. Hence, finding the best interpolant involves finding the best parametrization, a problem which is very difficult.

From classical interpolation theory we know that for a well chosen set of interpolation points (e.g., the Gauss points), a scalar function $u \in H^\sigma([-1, 1])$ can be interpolated by polynomials $I_N u$ with an interpolation error [3]

$$||u - I_N u||_{L^2} \leq c N^{-\sigma} ||u||_{H^\sigma(\Omega)}, \tag{2.4}$$

where c is a constant. If u is analytic, the error will decrease faster than any algebraic power of N, and we obtain exponential convergence. This also translates to vector-valued functions.

Let us illustrate the importance of reparametrization with an example. Consider a curve defined by the parametrization

$$\begin{aligned} f_1(\eta) &= \eta + 1, \\ f_2(\eta) &= \sqrt{(\eta + c)^{1/3} - 1}, \\ f_3(\eta) &= (\eta + c)^{2/3} - 1, \end{aligned} \tag{2.5}$$

where $\eta \in [-1,1]$ and c is a constant. For $c > 2$ the parametric functions are smooth, and $\boldsymbol{f}$ can be interpolated by polynomials to exponential convergence. However, applying the particular change of variable

$$\eta = \varphi(\xi) = \left((a\xi + b)^2 + 1\right)^3 - c$$

to $\boldsymbol{f}$, we get the reparametrization

$$\begin{aligned} g_1(\xi) &= ((a\xi + b)^2 + 1)^3 - c + 1, \\ g_2(\xi) &= a\xi + b, \\ g_3(\xi) &= ((a\xi + b)^2 + 1)^2 - 1, \end{aligned} \tag{2.6}$$

where all the parametric functions are polynomials of degree less than or equal to six. Interpolating this parametrization will result in exact representation of the curve for $N \geq 6$, which is obviously a great improvement.

When using Legendre spectral element methods in deformed hexahedra, the edges are approximated by parametric curves and the faces are approximated by parametric surfaces [10]. The end points of a curve are interpolation points and the interpolation points are mapped from the Gauss-Lobatto-Legendre (GLL) points ξ_j, $j = 0, \ldots, N$ by the interpolant. We represent such an interpolant using Lagrange interpolation polynomials through the GLL points,

$$p_i(\xi) = \sum_{j=0}^{N} \alpha_j^i \ell_j(\xi), \qquad i = 1, 2, 3, \tag{2.7}$$

where the coefficients α_j^i are determined by the interpolation conditions. In classical interpolation, this means simply evaluating the given parametrization $\boldsymbol{f}$ in the GLL points, i.e., $\alpha_j^i = f_i(\xi_j)$. Reparametrizing the curve before interpolating yields $\alpha_j^i = g_i(\xi_j)$, in which case the interpolation points are no longer mapped from the GLL points by $\boldsymbol{f}$, but rather from the points

$$\eta_j = \varphi(\xi_j), \qquad j = 0, \ldots, N. \tag{2.8}$$

Note that due to the nodal representation (2.7), the interpolant always maps the GLL points to the interpolation points.

In the current context, the mapping $\varphi(\xi)$ is unknown, since we do not know *a priori* which reparametrization is best suited for polynomial interpolation. The η_j, $j = 0, \ldots, N$, in (2.8) can thus be viewed as *free variables* which can be manipulated subject to certain restrictions, imposed by W. Specifically, we must require all $\eta_j \in [-1, 1]$, and that they appear in consecutive order, i.e.,

$$-1 \leq \eta_0 < \eta_1 < \ldots < \eta_N \leq 1. \tag{2.9}$$

Interpolation of the end points implies setting $\eta_0 = -1$ and $\eta_N = 1$, and we are left with $N - 1$ *degrees-of-freedom* which can be used to improve the approximation properties of the interpolant.

The interpolant is uniquely defined by the choice of η_j through the definition

$$\alpha_j^i = f_i(\eta_j) \tag{2.10}$$

of the expansion coefficients in (2.7). The change of variable φ, on the other hand, is only partially determined by (2.8). To turn the statement around, one can say that there are (infinitely) many reparametrizations that, when interpolated in the classical sense, yield the same interpolant. It will be convenient to choose φ to be the polynomial of lowest degree that satisfies (2.8). This can be done if the polynomial interpolating $\eta_0, \ldots, \eta_N$ is monotonic. It is then a (uniquely determined) function in $W \cap \mathbb{P}_N([-1, 1])$, and we will refer to it as φ_N.

2.1 Measuring the interpolation error

It is not trivial to define a norm for the interpolation error in the context of parametric curves. The norm should measure the distance between the geometric objects represented by $\boldsymbol{f}$ and $\boldsymbol{p}$, regardless of the particular parametrizations chosen. One metric satisfying this requirement is the *Hausdorff metric* [21]. Unfortunately, this norm is not very well suited for numerical calculations. Other possibilities are the *normal distance* proposed by Degen [7], or the metric proposed by Mørken and Scherer [20].

We will use none of these metrics, but rather the L^2-like norm

$$||\boldsymbol{f} - \boldsymbol{p}|| = \Big(\int_a^b \sum_{i=2}^{3} \big(f_i(f_1^{-1}(x)) - p_i(p_1^{-1}(x))\big)^2 \, \mathrm{d}x \Big)^{1/2}, \tag{2.11}$$

where $a = f_1(-1)$ and $b = f_1(1)$. It will be implemented using GLL quadrature with overintegration to ensure that the quadrature error is sub-dominant. The reason for choosing this norm is that it makes the interpolation error an explicit function of the free variables η_j, $i = 1, \ldots, N - 1$. This will enable us to define an interpolation method based on direct minimization of the interpolation error.

It might not be immediately clear why a norm like

$$||\boldsymbol{f} - \boldsymbol{p}|| = \Big(\int_{-1}^{1} \sum_{i=1}^{3} \big(f_i(\varphi_N(\xi)) - p_i(\xi) \big)^2 \,\mathrm{d}\xi \Big)^{1/2} \tag{2.12}$$

is not acceptable. The problem is that basing the norm on an integral over the parametric domain makes the norm parametrization-dependent. Even though $\boldsymbol{f}(\varphi_N(\xi_j)) = \boldsymbol{p}(\xi_j), j = 0, \ldots, N$ (i.e., the reparametrization and the interpolant reach the interpolation points "at the same time"), it is not given that $\boldsymbol{p}(\xi)$ is the best approximation of $\boldsymbol{f}(\varphi_N(\xi))$ for any other given ξ. After all, φ_N was *chosen* among all the changes of variable that yield the interpolant $\boldsymbol{p}$.

It should be noted that this is mainly a theoretical problem. If $\boldsymbol{f}$ is smooth, then so is $\boldsymbol{f} \circ \varphi_N$, and hence $\boldsymbol{p}$ will approximate it to exponential convergence. This implies that the curves are being traversed with approximately the same velocity. Moreover, if $f_1(\eta) = \eta$, then (2.11) and (2.12) coincide.

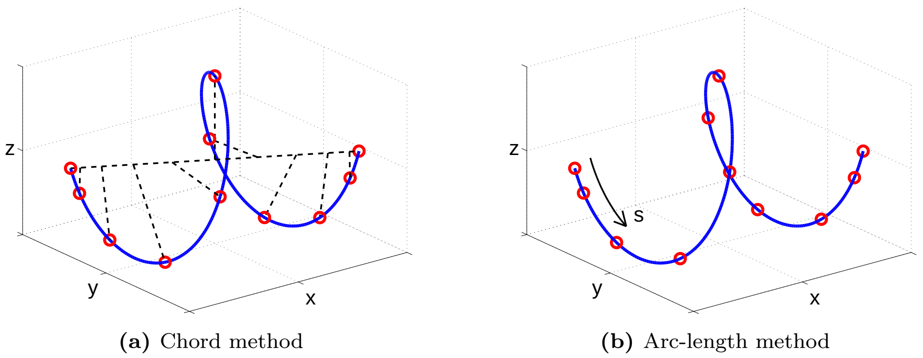
The definition of the norm (2.11) puts a restriction on the curves that can be studied, since it is only defined when the first parametric function $f_1(\eta)$ is monotonic, i.e., when the curve can be uniquely determined by specifying its x-coordinate. This restriction on the curves is not a limitation on the interpolation methods studied here. However, the numerical results will only include curves from this subset in order to be able to quantitatively compare the different methods.

2.2 Interpolation methods

As already mentioned, the simplest way to interpolate a parametric curve $\boldsymbol{f}$ is to view it as a vector-valued function and let $\alpha_j^i = f_i(\xi_j)$, $j = 0, \ldots, N$. This may be satisfactory if we know f_i, $i = 1, 2, 3$ to be smooth functions, but in other cases it may be far from optimal.

In the high order methods community there are two common interpolation methods that are independent of the parametrization [10]. Both rely in some way on an affine mapping of the GLL points from the parametric variable to the physical domain.

The first interpolation method considered here, referred to as the *chord method*, is defined by first mapping the GLL points affinely to the chord between the two end points of the curve, and then letting the interpolation points be the intersection between the exact curve and the normal planes to the chord at these affinely mapped points (Figure 1a). Finding these intersection points requires an iterative procedure like Newton's method. The chord method obviously does not work for closed curves, and it also fails in cases where the the curve intersects a normal plane to the chord in more than one point. However, this is not a significant limitation in the context of high order methods for solving PDEs. The chord method will yield (rapid) exponential convergence if the curve can be described by a smooth function in a rotated coordinate system where the abscissa is parallel to the chord.



(a) Chord method **(b)** Arc-length method

Figure 1: Two common methods for choosing interpolation points. Both methods involve an affine mapping of the GLL points: the chord method along the chord between the end points, the arc-length method in the arc-length variable s.

The second method is based on a GLL distribution in the arc-length variable s, and is called the *arc-length method*. On a curve of length L, construct the affine mapping $s(\xi) = \frac{L}{2}(\xi + 1)$, $\xi \in [-1, 1]$, and define the values $s_j = s(\xi_j)$, $j = 0, \ldots, N$, associated with the GLL points ξ_j. Each value s_j corresponds to a unique point along the curve with coordinates (x_j, y_j, z_j), which is then defined as an interpolation point (Figure 1b). Again, an iterative procedure such as Newton's method is required.

The arc-length method is equivalent to interpolating a reparametrization $\boldsymbol{g}$ with constant Jacobian

$$J(\xi) = \left(\sum_{i=1}^{3} g_i'(\xi)^2 \right)^{1/2} .$$

This, of course, does not guarantee that the parametric functions are smooth.

In the example with the curve parametrized by (2.5) and (2.6), none of these two interpolation methods correspond to classical interpolation of $\boldsymbol{f}$ or $\boldsymbol{g}$.

2.3 The L^2-method

A good interpolant should yield a small interpolation error, measured in the norm (2.11). The norm was chosen because it enables us to explicitly evaluate the measured interpolation error as a function of the free variables $\eta_1, \ldots, \eta_{N-1}$ (when approximated with GLL quadrature). This makes it possible to define an interpolation method based on direct minimization of the measured interpolation error, using a (global) optimization algorithm. We will use the objective function

$$\Lambda(\eta_1, \ldots, \eta_{N-1}) = ||\boldsymbol{f} - \boldsymbol{p}||^2, \tag{2.13}$$

and the method will be referred to as the L^2*-method.*

In order to avoid the evaluation of the inverses of f_1 and p_1, we only consider parametrizations $\boldsymbol{f}$ where the first component $f_1(\eta)$ is linear. All curves that can be measured by the norm (2.11) can be reparametrized this way. We then have

$$f_1(\varphi_N(\xi)) = p_1(\xi), \tag{2.14}$$

which allows a change of variable $x = f_1(\varphi_N(\xi))$, and the norm can be rewritten as

$$||\boldsymbol{f} - \boldsymbol{p}|| = \Big(\int_{-1}^{1} \sum_{i=2}^{3} \left(f_i(\varphi_N(\xi)) - p_i(\xi)\right)^2 p_1'(\xi)\, \mathrm{d}\xi \Big)^{1/2}, \tag{2.15}$$

Here, all terms can be evaluated explicitly; GLL quadrature will be used to evaluate the integrals.

The minimization is implemented using Newton's method, which requires the first and second order partial derivatives of Λ w.r.t. $\eta_1, \ldots, \eta_{N-1}$. These can be found explicitly from the formulation (2.15) when the change of variable φ_N is viewed as a function of both ξ and $\eta_1, \ldots, \eta_{N-1}$. It is represented using a standard linear combination of N-th order Lagrangian interpolants

$$\varphi_N(\xi; \eta_1, \ldots, \eta_{N-1}) = \sum_{j=0}^{N} \eta_j \ell_j(\xi). \tag{2.16}$$

Similarly, the components of the polynomial interpolant are defined as

$$p_i(\xi; \eta_1, \ldots, \eta_{N-1}) = \sum_{j=0}^{N} f_i(\eta_j) \ell_j(\xi). \tag{2.17}$$

The first partial derivatives of (2.13) with respect to the η_j are then given by

$$\frac{\partial \Lambda}{\partial \eta_j} = \int_{-1}^{1} \sum_{i=2}^{3} 2\left(f_i(\varphi_N(\xi)) - p_i(\xi)\right)\left(f_i'(\varphi_N(\xi))\frac{\partial \varphi_N}{\partial \eta_j}(\xi) - \frac{\partial p_i}{\partial \eta_j}(\xi)\right) p_1'(\xi)$$
$$+ \left(f_i(\varphi_N(\xi)) - p_i(\xi)\right)^2 \frac{\partial p_1'}{\partial \eta_j}(\xi)\,\mathrm{d}\xi,$$

where the simplified notation $p_1'(\xi)$ is used to remind the reader that p_1 is originally a function of ξ, even though it also depends on the *parameters* $\eta_1, \dots, \eta_{N-1}$. The second order partial derivatives $\frac{\partial^2 \Lambda}{\partial \eta_j \partial \eta_k}$ are easily derived by repeated partial differentiation, and we do not write them out here. They include terms with the first and second derivatives of f_2 and f_3, so these need to be known explicitly. The remaining functions can be differentiated numerically without error by means of differentiation matrices, since all of the functions are polynomials.

The L^2-method should, by construction, give the best interpolant in terms of the measured interpolation error. However, it is based on a very hard global minimization problem. The objective function Λ is almost never globally convex, and its complexity increases as the polynomial degree N increases. This is connected to the global interpolation approach: moving just one interpolation point (locally) changes the entire interpolant (globally). Newton's method is a local minimization algorithm and can not be expected to find the global minimum. Some measures will be taken to make the method more robust (see Section 2.7), but the increasing complexity will be reflected in the numerical results; see Section 3.

2.4 Geometric interpolation

From classical interpolation theory we know that interpolation of a smooth function in $N+1$ points yields approximation order $N+1$. This is an incentive for using the available degrees-of-freedom in curve interpolation to *increase* the number of interpolation points.

By construction, the points $\boldsymbol{p}(\xi_j)$, $j = 0, \dots, N$ are always interpolation points. To achieve interpolation in one *additional* point, we need $\boldsymbol{p}$ to satisfy the interpolation condition

$$\boldsymbol{f}(\varphi_N(\xi^*)) = \boldsymbol{p}(\xi^*) \tag{2.18}$$

for a ξ^* that is *not* a GLL point. Equation (2.18) represents a system of three (non-linear) equations, and ξ^* is a free variable. This means that we need two more degrees-of-freedom to find a solution in the general case. Since we have $N-1$ free variables $\eta_1, \dots, \eta_{N-1}$, it is in principle possible to achieve a total of $N + 1 + \lfloor (N-1)/2 \rfloor$ interpolation points.

This argumentation can also be applied to interpolation of curves in $\mathbb{R}^d$. In this case we still have $N-1$ free variables, but now an additional interpolation point requires $d-1$ degrees-of-freedom. Assuming that the system of non-linear equations always has a solution leads to the following conjecture [20]:

Conjecture 1. *Let $\mathscr{C}$ be a curve in $\mathbb{R}^d$. A polynomial curve of degree N can be made to interpolate $\mathscr{C}$ at*

$$m = N + 1 + \left\lfloor \frac{N-1}{d-1} \right\rfloor \tag{2.19}$$

points.

The conjecture also applies to Hermite interpolation if one defines interpolation of k coalescing interpolation points as interpolation of a (yet unknown) reparametrization $\boldsymbol{g}$ and its $k-1$ first derivatives. For the first derivative, this means that we do not have to require $\boldsymbol{f}'(\varphi(\xi^*)) = \boldsymbol{p}'(\xi^*)$, only that they point in the same direction. This kind of requirement can be expressed in terms of *geometric continuity.* A curve $\mathscr{C}$ is said to be *G^k-continuous* if its arc-length parametrization is C^k-continuous [1]. An equivalent definition can be found in [9]. In terms of interpolation we say that two curves have *contact order k* if the left segment of the interpolant meets the right segment of the exact curve with G^k-continuity, and vice-versa. First order contact means a common tangent direction, while second order contact additionally requires common curvature *and* coinciding osculating planes.

With this definition, coalescing interpolation points means increased contact order. Interpolation in the conjectured maximum number of interpolation points, but with some points coalescing, is exactly the same as *geometric Hermite interpolation* that was described in the introduction. Consider for example cubic polynomial curves in $\mathbb{R}^2$, which can interpolate a given curve in six points, according to the conjecture. If three interpolation points coalesce at each end point, the contact order is raised to two, and we have the same interpolation conditions as in [6].

In the extreme case where all interpolation points coalesce to *one* point we get a Taylor approximation of $\boldsymbol{f}$. For a planar parametric curve $\boldsymbol{f}(\eta) = (\eta, y(\eta))$ one can easily show [22] that a one-point G^k-interpolant coincides with the $k+1$ first terms of the Taylor expansion of y. Hence, one-point G^k-interpolation in

$\mathbb{R}^2$ yields *approximation order* $k+1$. The argument can be extended to general curves in $\mathbb{R}^d$; see [20] for definitions of norms and approximation order. Then, according to the conjecture, (2.19) is the highest attainable approximation order for curve interpolation in $\mathbb{R}^d$ at a given N. Some authors [15, 22] have actually formulated the conjecture in terms of approximation order, stating that the approximation order (2.19) can be attained for any curve $\mathscr{C}$ in $\mathbb{R}^d$.

The conjecture has turned out to be very difficult to prove. Since the subject has been studied mostly within the CAGD community, most authors are concerned with low polynomial degrees ($N \leq 5$). Of more general results, we mention Rababah [22, 23] who showed that curves in $\mathbb{R}^d$ can be interpolated by one-point interpolation to approximation order $4N/3$ for arbitrary N, and Floater [13] who showed optimal approximation order $2N$ for conic sections.

2.5 The extra-points method

We propose an interpolation method based on Conjecture 1, which we will refer to as the *extra-points method*.

Assuming that f_1 is invertible, the interpolation condition (2.18) can be reduced to a system of two equations by *choosing* a specific ξ^* and *defining* $x^* = f_1(\varphi_N(\xi^*)) = p_1(\xi^*)$. Interpolation at x^* then requires the two equations

$$f_i(f_1^{-1}(x^*)) = p_i(p_1^{-1}(x^*)), \qquad i = 2, 3$$

to be solved. Furthermore, when f_1 is linear then (2.14) holds, and the inverses can be eliminated. The extra-points method can then be defined as finding a root of the vector-valued function $\boldsymbol{\Psi}$ with components

$$\Psi_k^i(\eta_1, \dots, \eta_{N-1}) = f_i(\varphi_N(\xi_k^*)) - p_i(\xi_k^*), \qquad i = 2, 3, \tag{2.20}$$

where ξ_k^*, $k = 1, \dots, \lfloor (N-1)/2 \rfloor$ are pre-defined values in $[-1, 1]$. The dependency of Ψ_k^i on η_j, $j = 1, \dots, N-1$ is explicit in the representations (2.16) and (2.17) of φ_N and p_i, respectively. Note that for odd N there is one more degree-of-freedom than the number of equations to be solved, so we are left with one "unused" degree-of-freedom.

Applying Newton's method to solve (2.20) requires the partial derivatives of Ψ_k^i, which are given by

$$\frac{\partial \Psi_k^i}{\partial \eta_j} = f_i'(\varphi_N(\xi_k^*)) \frac{\partial \varphi_N}{\partial \eta_j}(\xi_k^*) - \frac{\partial p_i}{\partial \eta_j}(\xi_k^*).$$

The derivatives of the parametric functions $f_i(\eta)$ must be known; the rest are computed numerically with differentiation matrices.

The solution obtained depends on the choice of the values ξ_k^*. Numerical experiments show that the best results are usually obtained when the ξ_k^* are close to a subset of the GLL points. In the limit when the points coalesce, the proposed method becomes useless because (2.20) is always satisfied. According to the previous discussion, one should instead raise the contact order at the coalescing points. This leads to the next proposed method.

2.6 The equal-tangents method

Two coalescing interpolation points should yield first order contact (common tangent directions) between the exact curve and the interpolant. Common tangent directions implies that a tangent vector $\boldsymbol{t}_N$ to the interpolant is orthogonal to all vectors in the normal plane of the exact curve, i.e.,

$$\boldsymbol{t}_N \cdot \boldsymbol{n} = 0 \qquad \forall \boldsymbol{n} \text{ such that } \boldsymbol{t} \cdot \boldsymbol{n} = 0,$$

where $\boldsymbol{t}$ is a tangent vector to the exact curve. The space of normal vectors to a curve in $\mathbb{R}^3$ is two-dimensional, so we must use two linearly independent normal vectors and make both dot products zero. The tangent vector is easily found by differentiating the parametrization,

$$\boldsymbol{t}_N = \begin{bmatrix} p_1'(\eta) \\ p_2'(\eta) \\ p_3'(\eta) \end{bmatrix}.$$

It does not have to be normalized for our application, which is an advantage, since it would have resulted in more complicated non-linear equations. We then *choose* the two normal vectors

$$\boldsymbol{n}_1 = \begin{bmatrix} f_2'(\eta) \\ -f_1'(\eta) \\ 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{n}_2 = \begin{bmatrix} f_3'(\eta) \\ 0 \\ -f_1'(\eta) \end{bmatrix}.$$

Again, this is a choice to ease the implementation. It clearly would not work for curves where $f_1'(\eta)$ can be zero, since the two vectors then become linearly dependent, but we have already excluded such curves from the current study for the definition of the norm (2.11).

The number of degrees-of-freedom allows G^1-interpolation in $\lfloor (N-1)/2 \rfloor$ (unique) points. By construction we have interpolation in $N+1$ points, so a subset of the interpolation points must be chosen; we choose the set of internal points ξ_k with odd indices k. The interpolation conditions can then be expressed as a system of equations

$$\Theta_k^i(\eta_1, \dots, \eta_{N-1}) = f_1'(\varphi_N(\xi_k))\, p_i'(\xi_k) - f_i'(\varphi_N(\xi_k))\, p_1'(\xi_k) = 0, \qquad i = 2, 3. \tag{2.21}$$

Again Newton's method is used for finding a solution of the non-linear system. This requires us to differentiate (2.21) with respect to the independent variables $\eta_1, \dots, \eta_{N-1}$. The partial derivatives can be written out explicitly as

$$\begin{aligned}\frac{\partial \Theta_k^i}{\partial \eta_j} =& f_1''(\varphi_N(\xi_k))\frac{\partial \varphi_N}{\partial \eta_j}(\xi_k)\, p_i'(\xi_k) + f_1'(\varphi_N(\xi_k))\frac{\partial p_i'}{\partial \eta_j}(\xi_k) \\ &- f_i''(\varphi_N(\xi_k))\frac{\partial \varphi_N}{\partial \eta_j}(\xi_k)\, p_1'(\xi_k) - f_i'(\varphi_N(\xi_k))\frac{\partial p_1'}{\partial \eta_j}(\xi_k).\end{aligned}$$

Note again that for odd N we have one more degree-of-freedom than the number of equations.

2.7 Implementation

The three interpolation methods proposed here (the L^2-method, the extra-points method and the equal-tangents method) are based on quite simple criteria, but their implementations are challenging. We mention a few aspects here that are important in order for the methods to work well in practice.

As mentioned previously, the extra-points method depends on the choice of ξ_k^*; letting these parameters be close to some of the GLL points is often a good choice. In all the numerical experiments here $\xi_k^* = \xi_{2k-1} + \varepsilon$ for $k = 1, \dots, \lfloor (N-1)/2 \rfloor$ and $\varepsilon = 10^{-2}$.

All the proposed methods are highly dependent on a good set of initial values η_j for the Newton iterations. For the L^2-method, this is connected to the fact that we are trying to solve a *global* minimization problem with a *local* minimization algorithm. For the extra-points and equal-tangents methods, it is due to the fact that non-linear functions may have several roots. The conjecture says nothing about the uniqueness of the solution, and numerical experiments have confirmed the existence of several solutions in many cases. We want the solution with the smallest interpolation error, which we will refer to as the optimal solution for the given interpolation method.

We have made the observation that when we are able to find the optimal solution (or something close to optimal), the reparametrization seems to converge to a particular function as the polynomial degree increases. In other words, for a given (high) N, the functions $p_i(\xi)$, $i = 1, 2, 3$ are very similar to the corresponding functions at $N - 1$. This leads us toward the idea of a bootstrapping algorithm, in which we use the solution from $N - 1$ as the initial guess by evaluating $\varphi_{N-1}(\xi)$ in the current $N + 1$ GLL points. Starting all the way from a polynomial degree of one, such a bootstrapping approach implies an added computational cost. However, the improved robustness has been more important in the current study.

The bootstrapping approach often yields good initial guesses, but not always. Newton's method may not succeed, or it may find a non-optimal solution (we can usually recognize non-optimal solutions from a sudden change in the convergence rate as N increases). In such cases it may help to use other initial guesses, e.g., perturbations of the bootstrapping solution, or the solutions found with the chord or arc-length methods. Here, we use such additional initial values to increase the robustness of the methods. When different initial values yield different solutions after the Newton iterations, the interpolation error is compared and the solution with the smallest interpolation error is chosen.

One can also add to the robustness by making sure that the solution never violates the restriction (2.9) imposed by W during the Newton iterations. Experience shows that the extra-points and equal-tangents methods sometimes find "illegal solutions". To avoid this, we limit the step sizes in the Newton iterations, and we also explicitly check the condition (2.9).

3 Numerical Results

We now present a series of numerical tests to illuminate the challenges of reparametrization and to illustrate the performance of the various methods in different situations. All the parametrizations are defined on the interval $[-1, 1]$.

Case 1

Consider one and a half rotations of a helix, which is most naturally parametrized by

$$\begin{aligned} f_1(\eta) &= \frac{3}{2}\pi\eta, \\ f_2(\eta) &= \sin(\frac{3}{2}\pi\eta), \\ f_3(\eta) &= \cos(\frac{3}{2}\pi\eta). \end{aligned}$$

This particular parametrization yields a constant Jacobian $J = 3\pi/\sqrt{2}$, so the arc-length method corresponds to interpolating $\boldsymbol{f}$ in the GLL points. Due to the regularity of the given parametric functions, it gives rapid exponential convergence; see Figure 2. The chord method also yields exponential convergence, but much slower than the arc-length method. To reach the same level of interpolation error, approximately three times the polynomial degree N is needed with the chord method. In the context of solving PDEs using high order methods, this has a huge impact on the computation time.

The extra-points and equal-tangents methods give almost exactly the same interpolation error; in fact, the solutions are almost exactly the same. This is due to our choice of ξ_k^* in the extra-points method. Both methods converge quite a bit faster than the arc-length method, reaching machine precision at $N = 15$.

By construction, the L^2-method should give the optimal solution. However, experience shows that the L^2-method usually finds the optimal solution (or something very close) for small N when the objective function is easier to minimize globally. If an interpolation method is able follow and maintain this convergence rate for higher N, it is a strong indication that it is able to yield a solution which is close to optimal. This is what we observe in this case, strongly suggesting that the extra-points and equal-tangents methods yield close to the optimal solution.

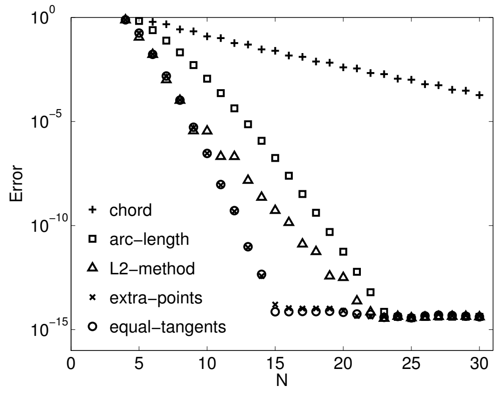


Figure 2: Interpolation error for Case 1, the helix, measured in a discrete version of the norm (2.11). Note the large difference in convergence rate between the chord method and the arc-length method. The latter is usually considered optimal for this particular case, but the new methods show that it is possible to do better.

Case 2

The curve defined by the parametrization (2.5) is designed in such a way that we know that there exists a reparametrization (2.6) by low order polynomials. However, none of the interpolation methods get this reparametrization as an initial value for their iterative procedures.

The chord method and the arc-length method both converge exponentially; see Figure 3. Note that this could not be foreseen from the given parametrizations, as these methods do not correspond to interpolating any of them. We also see that the new methods give considerably better results, although none of them give exact representation of the curve at $N = 6$. The extra-points and equal-tangents methods are very close, with error on machine level precision from $N = 7$. The L^2-method also yields good results, reaching machine level precision at $N = 14$. It does, however, display the weakness that is characteristic for this method: the functional to be minimized becomes increasingly complicated as N increases, with many local minima, and our simple minimization algorithm has difficulty finding a global minimum. The result is a convergence rate that decreases as N increases.

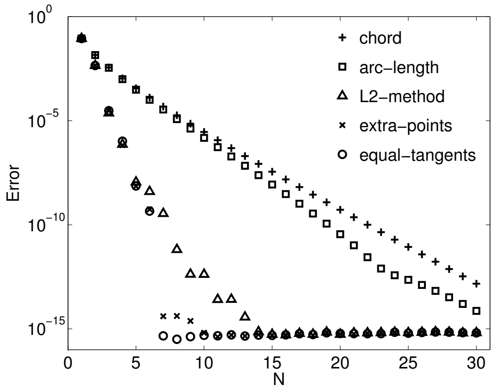


Figure 3: Interpolation error for Case 2, measured in the same norm as before. The reparametrization (2.6) shows that exact representation of the curve is possible at $N = 6$. None of the methods achieve this, but the extra-points and equal-tangents methods yield rapid convergence, representing the curve to machine precision at $N = 7$. The chord method and the arc-length method both converge exponentially, but much slower than the new methods.

Case 3

The parametric curve

$$\begin{aligned} f_1(\eta) &= -\frac{5}{2} + \frac{7}{4}(\eta + 1), \\ f_2(\eta) &= \frac{1}{2}\sin\left(\frac{3\pi\eta}{2}\right) + \frac{9}{8}(\eta+1)^2 - \frac{9}{4}(\eta + 1), \\ f_3(\eta) &= \frac{1}{2}\cos(\pi\eta), \end{aligned} \tag{3.1}$$

has only analytic components, so a classical approach, simply interpolating the given functions in the GLL points, will give (rapid) exponential convergence. However, the chord method and the arc-length method both yield very slow (although still exponential) convergence; see Figure 4. Hence, neither of them correspond to classical interpolation.

The new methods all give vastly better performance (and better than classical interpolation of (3.1)). They converge at approximately the same rate and reach machine precision between $N = 15$ and $N = 20$. Compared to the two traditional methods, only a small fraction of the polynomial degree is needed to reach the same level of accuracy.

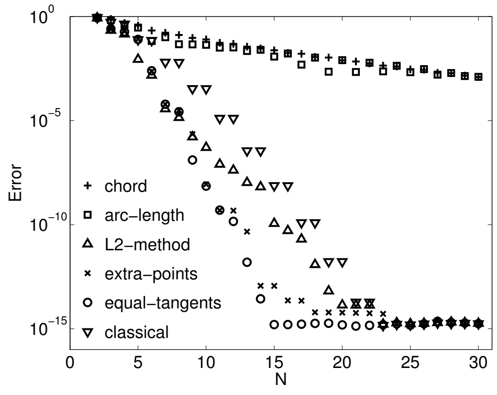


Figure 4: Interpolation error for Case 3. The given parametrization (3.1) consists of smooth functions, so classical interpolation results in rapid exponential convergence. Neither the chord method nor the arc-length method corresponds to classical interpolation of (3.1), and both methods yield very low convergence rate.

Case 4

The parametric curve

$$
\begin{aligned}
f_1(\eta) &= \eta + 1, \\
f_2(\eta) &= \sin(\pi(\eta + 1)), \\
f_3(\eta) &= \sin(2\pi\eta),
\end{aligned}
$$

bears some resemblance to the parametrization of the helix, except that the trigonometric functions are phase-shifted and have different periods. Interestingly, comparing the Figures 2 and 5, we see that the relative performance of the chord method and the arc-length method are opposite. In the current case, the chord method is vastly better than the arc-length method, which converges extremely slowly and is rather useless.

Again, the new methods outperform the traditional ones. The extra-points and equal-tangents methods give very similar results and converge very fast, and the L^2-method is almost as good.

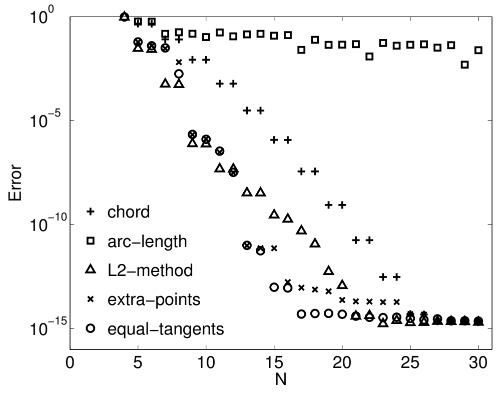


Figure 5: Interpolation error for Case 4. The results are similar to Case 3, except that the chord method is now efficient, while the arc-length method is useless.

Case 5

Consider the parametric curve

$$\begin{aligned} f_1(\eta) &= \eta, \\ f_2(\eta) &= |\eta - \frac{1}{2}|, \\ f_3(\eta) &= |\eta + \frac{1}{2}|. \end{aligned}$$

Again the Jacobian $J = \sqrt{3}$ is constant, so classical interpolation of $\boldsymbol{f}$ corresponds to the arc length method. The curve is G^0, i.e., it has break points, and $\boldsymbol{f}$ has two C^0 components. Such a curve is normally considered unsuited for classical high order interpolation, and indeed the arc-length method gives low order algebraic convergence. So does the chord method, as Figure 6 shows.

The new methods, on the other hand, give exponential convergence. The interpolation points are clustered close to the break points, so the implicitly defined reparametrization is almost stationary at these points.

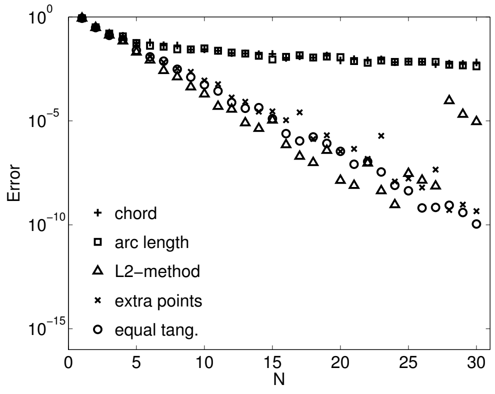


Figure 6: Interpolation error for Case 5. The chord and arc-length methods yield low order algebraic convergence, whereas the new methods converge exponentially, except for some instability for large N.

Case 6

The parametric curve

$$\begin{aligned} f_1(\eta) &= \eta, \\ f_2(\eta) &= \frac{1}{1+16\eta^2}, \\ f_3(\eta) &= \frac{1}{1+16(\eta+1)^2}, \end{aligned}$$

is difficult to interpolate since two of its components are Runge functions. It was shown in [2] that the Runge function can be very well approximated (without oscillations) when viewed as a planar parametric curve. Here, one of the functions is shifted along the x-axis so that the curve is not equivalent to the standard Runge function.

The chord method and the arc-length method both result in very low convergence rates; see Figure 7. The chord method yields unwanted oscillations in the solution, whereas the arc-length method method results in a poor approximation of $f_2(\eta)$ around $\eta = 0$. The three new methods converge fast in comparison, reaching machine precision around $N = 30$. These methods yield practically no unwanted oscillations.

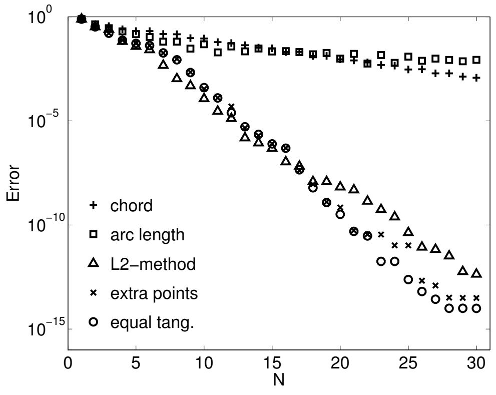


Figure 7: Interpolation error for Case 6. The chord and arc-length methods yield very slow convergence. Again there is a huge gap in performance between these methods and the three new methods.

4 Interpolation of parametric surfaces

Let $\boldsymbol{f}$ be a parametric surface in $\mathbb{R}^3$, described in a Cartesian coordinate system by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_1(\eta_1, \eta_2) \\ f_2(\eta_1, \eta_2) \\ f_3(\eta_1, \eta_2) \end{bmatrix} = \boldsymbol{f}(\eta_1, \eta_2), \qquad \eta_1, \eta_2 \in [-1, 1]. \tag{4.1}$$

The parametrization can be viewed as mapping $\boldsymbol{f}$ from a *reference domain* $\widehat{\Omega} = [-1, 1] \times [-1, 1] \subset \mathbb{R}^2$ to a physical domain $\Omega \in \mathbb{R}^3$. A reparametrization of the surface can be found by a change of variables: for all bijective maps φ from $\widehat{\Omega}$ onto itself, the function

$$\boldsymbol{g}(\xi_1, \xi_2) = \boldsymbol{f}(\varphi(\xi_1, \xi_2)) = \boldsymbol{f}(\eta_1, \eta_2)$$

describes the same surface. Note that ξ_1 and ξ_2 here represent two independent variables and not two GLL points; it should be clear from the context what is meant. Different parametrizations can consist of functions of different regularity, and this will affect the convergence rate in polynomial interpolation.

The interpolant is a parametric surface $\boldsymbol{p}$ described by

$$\boldsymbol{p}(\xi_1, \xi_2) = \begin{bmatrix} p_1(\xi_1, \xi_2) \\ p_2(\xi_1, \xi_2) \\ p_3(\xi_1, \xi_2) \end{bmatrix}, \qquad \xi_1, \xi_2 \in [-1, 1],$$

where each component p_i is a polynomial of degree less than or equal to N in each reference variable. It is conveniently represented by sums of Lagrangian interpolants in the tensor-product GLL points, i.e.,

$$p_i(\xi_1, \xi_2) = \sum_{m=0}^{N} \sum_{n=0}^{N} \alpha^i_{mn} \ell_m(\xi_1) \ell_n(\xi_2), \qquad i = 1, 2, 3. \tag{4.2}$$

We note that, for a given ξ_1, $\boldsymbol{p}$ is a parametric curve when viewed as a function of ξ_2, and vice versa. Hence, the rectilinear mesh that is made up by the interpolation points in $\widehat{\Omega}$ is mapped to a curvilinear mesh on Ω; see Figure 8.

The expansion coefficients are determined by letting them be points somewhere on the exact surface, i.e., $\alpha^i_{mn} = f_i(\eta_{1,mn}, \eta_{2,mn})$. The coordinates $\eta_{1,mn}$ and $\eta_{2,mn}$ in $\widehat{\Omega}$ can be viewed as *free parameters* that implicitly determine the change of variable φ_N, a polynomial of degree N in ξ_1 and ξ_2 such that $(\eta_{1,mn}, \eta_{2,mn}) = \varphi_N(\xi_m, \xi_n)$, $0 \le m, n \le N$, i.e., $(\eta_{1,mn}, \eta_{2,mn})$ are the images

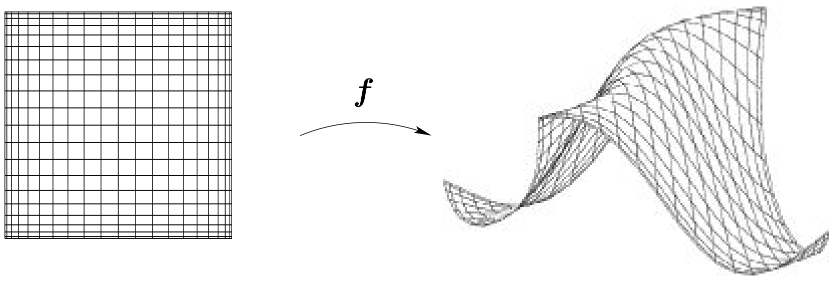


Figure 8: The surface is mapped from a reference domain $\widehat{\Omega} = [-1,1] \times [-1,1]$ by the parametrization $\boldsymbol{f}$. The interpolation points are mapped from the tensor-product GLL points.

of the tensor-product GLL points. There are $2(N+1)^2$ values to be interpolated; hence the dimension of the discrete space is $2(N+1)^2$.

We want the interpolation methods described in this paper to be applicable in the context of high order methods for solving PDEs in deformed hexahedra. The construction of a numerical approximation of the hexahedron often starts with an interpolation of the six faces, followed by a transfinite interpolation method to patch them together. For the latter to be possible, a consistent representation of the shared edges is necessary. This puts a few restrictions on the choice of $\eta_{1,mn}$ and $\eta_{2,mn}$. In particular, we require that the interpolation points that are mapped from the boundary of $\widehat{\Omega}$ interpolate the boundary of Ω, and that the corner points map to the corner points. This leaves $N-1$ free parameters on each of the four boundary curves of an individual face. Together with the $2(N-1)^2$ free parameters in the interior of Ω, we have a total of

$$2(N-1)^2 + 4(N-1) = 2N^2 - 2$$

degrees-of-freedom.

When each boundary curve is considered separately, these requirements are in essence the same as the requirements that we made for the curve interpolants in Section 2. This enables us to use the interpolation methods from Section 2 on each of the four boundary curves of Ω.

4.1 Interpolation error and approximation order

In order to be able to measure the interpolation error in a parametrization-independent norm, we will consider surfaces that can be represented as functions

$$z = h(x, y),$$

in a Cartesian coordinate system. This allows the use of the L^2-norm

$$||\boldsymbol{f} - \boldsymbol{p}|| = \Big(\iint_{\Pi_\Omega} \big(h(x,y) - h_N(x,y)\big)^2 \,\mathrm{d}x\,\mathrm{d}y \Big)^{1/2}, \tag{4.3}$$

where Π_Ω is the projection of Ω to the xy-plane and $h_N(x,y)$ is a parametrization-independent representation of the interpolant. The latter is not always readily available from the parametric description of the interpolant. However, in the case where f_1 and f_2 are affine in both variables the norm can (without error) be transformed to an integral over the reference domain $\widehat{\Omega}$,

$$||\boldsymbol{f} - \boldsymbol{p}|| = \Big(\int_{-1}^{1} \int_{-1}^{1} \big(f_3(\varphi_N(\xi_1,\xi_2)) - p_3(\xi_1,\xi_2)\big)^2 J \,\mathrm{d}\xi_1\,\mathrm{d}\xi_2 \Big)^{1/2}, \tag{4.4}$$

where

$$J = \frac{\partial f_1}{\partial \eta_1}\frac{\partial f_2}{\partial \eta_2} - \frac{\partial f_2}{\partial \eta_1}\frac{\partial f_1}{\partial \eta_2}$$

is the Jacobian of the mapping $\boldsymbol{f}$. To keep the evaluation of the norm simple, we restrict our investigation to parametric surfaces that fit this requirement. Note again that this is not a restriction for the interpolation methods. The error norm is implemented as a discrete version of (4.4), based on GLL quadrature with overintegration.

In the context of interpolation using a fixed polynomial degree N (and possibly using several patches to represent the entire surface), the concept of approximation order is relevant. With traditional interpolation methods, interpolation by polynomials of degree N gives approximation order $N + 1$. For example, bicubic Bézier patches give fourth order convergence as the size of the patch decreases.

The conjectured optimal approximation order in curve interpolation was based on counting the number of equations that must be solved and comparing it with the number of free parameters. The same can be done in the context of interpolation of surfaces. Mørken [19] did it by counting the number of (non-linear) equations that must be solved in order to reduce the degree of a classical bivariate Taylor approximant without reducing the approximation order. He showed that the approximation order k is bounded by

$$k \leq \sqrt{3N^2 + 9N - 23/4} - \frac{1}{2},$$

which means that one theoretically can achieve approximation order $k = 2N$ for $N < 7$. However, the asymptotically optimal approximation order as N increases is approximately $k = \sqrt{3}N$.

As before, in the context of high order interpolation, convergence is more conveniently evaluated in terms of how the interpolation error decreases as a function of N. Interpolating a parametric surface $\boldsymbol{f}$ where all the components f_i are analytic will give exponential convergence, while low order components will give algebraic convergence. Again, the goal is exponential convergence with the highest possible rate for all surfaces. The note on approximation order serves only as indication of the possible improvement when the free parameters are chosen in a clever way.

4.2 Interpolation methods

Classical interpolation of the parametric surface $\boldsymbol{f}$ means discarding the possibility of reparametrization and setting $\alpha^i_{mn} = f_i(\xi_m, \xi_n)$. This corresponds to interpolation of the vector-valued *function* $\boldsymbol{f}$ in the tensor-product GLL points.

A simple, parametrization-independent alternative is to apply one of the curve interpolation method from Section 2 to the boundary curves of the surface, and then to find the internal points by a method for transfinite interpolation, e.g., the Gordon-Hall algorithm [14]. However, relying only on a transfinite interpolation method can yield a very crude approximation of the interior of the surface. If all three components of the interpolant are determined by the Gordon-Hall algorithm, the interior points will in general not be interpolation points. We therefore add a third step to make sure that all the coefficients in (4.2) are interpolation points. This will be our basic interpolation procedure:

1. Interpolate the boundary as four separate space curves.
2. Use Gordon-Hall transfinite interpolation for x and y.
3. Find z by function evaluation $h(x, y)$ at the internal interpolation points.

If the function $h(x, y)$ is not known, the last step would require an iterative procedure to find η_1 and η_2 from the system

$$\begin{aligned} x &= f_1(\eta_1, \eta_2) \\ y &= f_2(\eta_1, \eta_2) \end{aligned}$$

in each internal interpolation point. The z-coordinate could then be found by evaluating $z = f_3(\eta_1, \eta_2)$.

This algorithm will be the basis for our extension of the chord method and the arc-length method to interpolation of surfaces. Hence, they are only applied to the boundary curves; the interpolation points in the interior are determined by steps 2 and 3. Since there is no natural way to define the chord or the arc-length across a surface, this is in fact the most natural way to extend the methods to surface interpolation.

Since the Gordon-Hall algorithm in principle represents the interior as a weighted sum of the boundaries, it is clear that given a smooth boundary representation, we get a smooth representation of the interior. However, we have no way of knowing if this will be an *optimal* representation of the interior. Hence, we add another step in the algorithm:

4. Apply a surface interpolation algorithm to improve the distribution of interpolation points in the interior.

There are two main reasons why we do not skip the first three steps and go directly to a surface interpolation algorithm for the entire surface. First, experience from numerical experiments have shown that a good representation of the boundary is sometimes the single most important factor in achieving a good representation of the surface. Secondly, the restriction we have made on the interpolation points on the boundary means that there is a difference between the boundary and the interior in the number of degrees-of-freedom associated with each interpolation point. Hence, a boundary point cannot be treated exactly like an interior point. This does not prohibit us from interpolating the entire surface simultaneously, but it makes it more natural to treat them separately.

The L^2-method can be defined as the optimization procedure to find the interpolant that minimizes the functional

$$\mathcal{J} = ||\boldsymbol{f} - \boldsymbol{p}||^2, \tag{4.5}$$

where $||\cdot||$ is the norm (4.3). For parametric surfaces $\boldsymbol{f}$ where f_1 and f_2 are affine, the method can be implemented based on the simpler form (4.4). Viewing the functional (4.5) as a function of the $2N^2 - 2$ independent variables $\eta_{1,mn}$

and $\eta_{2,mn}$, it is in principle possible to minimize it with Newton's method in the same way as was done for curves in Section 2.3. One can do this for the entire surface simultaneously, using all the free parameters, or one can apply it as step 4 in the algorithm, using only the free parameters in the interior.

Based on our experience with interpolation of parametrized curves, it should not come as a surprise that (4.5) is very hard to minimize. The rapidly increasing number of free parameters makes the method infeasible, and it will not be implemented here.

The extra-points method can also be extended to interpolation of surfaces. Due to the similarity between this method and the equal-tangents method, we choose to focus on only one of these methods in the context of surface interpolation. To avoid the dependence of the method on a choice of extra interpolation points, the equal-tangents method is chosen.

4.3 The equal-tangents method

When it comes to tangent and normal vectors, the situation for surfaces in $\mathbb{R}^3$ is in a sense opposite to the curve case: there is one unique surface normal and a two-dimensional tangent plane. The equal-tangents method must therefore be based on requiring equal *tangent spaces* or, equivalently, equal normal vectors. This can be achieved by making a normal vector to the exact surface orthogonal to two linearly independent tangent vectors to the interpolant at the chosen interpolation points. The normal vector is given by

$$\boldsymbol{n} = \Big(\frac{\partial f_2}{\partial \eta_1}\frac{\partial f_3}{\partial \eta_2} - \frac{\partial f_3}{\partial \eta_1}\frac{\partial f_2}{\partial \eta_2}, \frac{\partial f_3}{\partial \eta_1}\frac{\partial f_1}{\partial \eta_2} - \frac{\partial f_1}{\partial \eta_1}\frac{\partial f_3}{\partial \eta_2}, \frac{\partial f_1}{\partial \eta_1}\frac{\partial f_2}{\partial \eta_2} - \frac{\partial f_2}{\partial \eta_1}\frac{\partial f_1}{\partial \eta_2}\Big)^T, \tag{4.6}$$

and the natural choice of tangent vectors is

$$\boldsymbol{t}_1^N = \Big(\frac{\partial p_1}{\partial \xi_1}, \frac{\partial p_2}{\partial \xi_1}, \frac{\partial p_3}{\partial \xi_1}\Big)^T \qquad \text{and} \qquad \boldsymbol{t}_2^N = \Big(\frac{\partial p_1}{\partial \xi_2}, \frac{\partial p_2}{\partial \xi_2}, \frac{\partial p_3}{\partial \xi_2}\Big)^T. \tag{4.7}$$

Equal tangent spaces is achieved when

$$\boldsymbol{n} \cdot \boldsymbol{t}_i^N = 0, \qquad i = 1, 2. \tag{4.8}$$

Just as in the case of curve interpolation, this means two equations have to be solved for equal tangents in one interpolation point. However, each interpolation point in the interior of Ω is associated with *two* degrees-of-freedom, as opposed to points on curves that only yield *one* degree-of-freedom. Hence, if we are able to solve the resulting system of non-linear equations, equal tangents should be possible in *all* the internal interpolation points. The boundary curves, on the other hand, can be interpolated with equal tangents in only $\lfloor (N-1)/2 \rfloor$ the points.

The equal-tangents method can be implemented either according to the four-step algorithm or as a method for the entire surface. In order to study the importance of a good representation of the boundary, we implement the method according to the four-step algorithm, and consider the solution before and after the last step. We will refer to the first as *equal-tangents boundary* and the second as *equal-tangents surface*.

All the systems of non-linear equations are solved with Newton's method, and the remarks from Section 2.7 still apply. Most importantly, the dependency on good initial guesses is important to achieve the best solution in the interior of Ω, and a bootstrapping method will be applied.

5 Numerical Results

Case 1

For surfaces that can be described by functions on the form

$$h(x, y) = f(x) + g(y), \qquad a \leq x \leq b, \quad c \leq y \leq d,$$

the Gordon-Hall method should be sufficient for an optimal representation of the entire surface, given that we are able to find an optimal representation of the boundary. The reason for this is that for a fixed x^*, the curve described by $h(x^*, y)$ is the same as the boundary curves $h(a, y)$ and $h(b, y)$, only shifted vertically. Hence, the optimal set of interpolation points is the same. For example, consider the surface described by the function

$$h(x, y) = \sqrt{x+2} + \frac{1}{2}\arctan(y), \qquad -1 \leq x, y \leq 1. \tag{5.1}$$

The interpolation methods considered here are all based on a parametric description of the surface. This is trivial to find from the function description, using affine mappings f_1 and f_2.

Figure 9 shows that applying step 4 in the interpolation process, i.e., enforcing equal tangents also in the interior, makes no difference in the convergence rate. It is the representation of the boundary that separates the two equal-tangents methods from the other two methods.

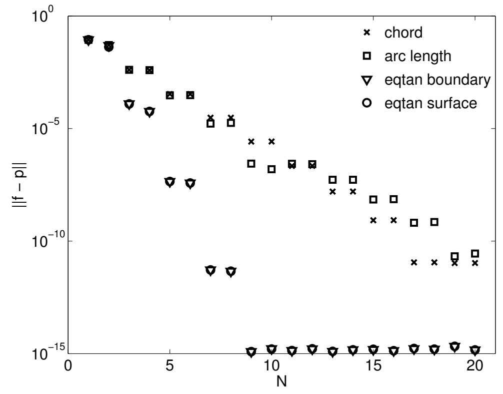


Figure 9: The interpolation error in Case 1, measured in the discrete L^2-norm. The surface can be described by a function of the form $h(x, y) = f(x) + g(y)$. With a good representation of the boundary, the Gordon-Hall algorithm is sufficient for a good representation of the entire surface.

Case 2

Consider the surface given by

$$h(x, y) = \frac{3}{2} + \frac{3}{10}\left(\frac{3}{2} - y\right) \sin\left(\frac{\pi}{3}x\right) + \frac{3}{10}\, y \cos\left(\frac{4}{3}\pi x\right), \qquad 0 \le x, y \le \frac{3}{2}. \quad (5.2)$$

Figure 10 shows that there is a vast difference between the chord method and the arc-length method in the convergence rate. Applying the equal-tangents method on the boundary gives only a slight improvement compared to the chord method, whereas the equal-tangents surface method gives a significant improvement. However, at $N = 9$ the latter seems to lose track of the optimal solution, and the convergence rate decreases dramatically. This is most likely due to a failure of Newton's method to find a solution to the equal-tangents problem in the interior. After all, at $N = 9$ there are $2(N - 1)^2 = 128$ free parameters in the interior, far more than what we ever encountered in curve interpolation.

Case 3

Consider now a surface described by a function of low regularity,

$$h(x, y) = (x^2 + y^2)^{3/2}, \qquad -1 \le x, y \le 1. \quad (5.3)$$

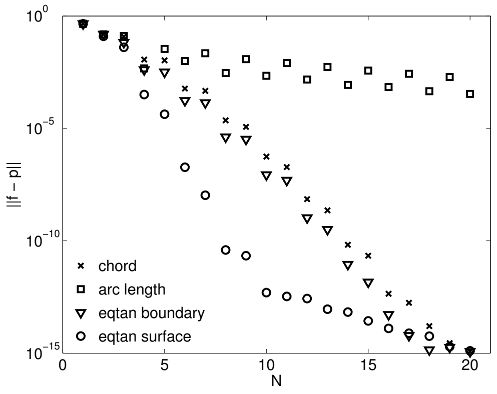


Figure 10: The interpolation error in Case 2. The surface is described by a smooth function, but the arc-length method results in a very low convergence rate. The equal-tangents surface method shows that very rapid convergence is possible, but we are not able to maintain the convergence rate until we reach machine precision, most likely due to the difficulty of finding solutions of the non-linear system of equations that arises from the equal-tangents conditions.

Affine mappings f_1 and f_2 will make f_3 a function of low regularity in both η_1 and η_2, and classical interpolation will give low order algebraic convergence. In fact, the chord method corresponds to the classical interpolant, and Figure 11 confirms the poor convergence rate. The arc-length method works even worse for this surface.

The reason for the low regularity of f_3 is a singularity in the third partial derivatives at the origin. The boundary curves, on the other hand, are smooth functions that can be interpolated by high order polynomials to exponential convergence. This is a problem for the equal-tangents boundary method; it may give a good representation of the boundary, but the Gordon-Hall algorithm does not take the singularity at the origin into account. The result is low order algebraic convergence.

However, with the addition of the fourth step in the algorithm, we are indeed able to get exponential convergence. Figure 12 shows the mesh of interpolation points projected onto the xy-plane, with polynomial degree $N = 15$ and using the equal-tangents boundary and equal-tangents surface methods. The latter results in all internal interpolation points moving toward the origin, the position of the singularity. This corresponds to interpolating a reparametrization of the

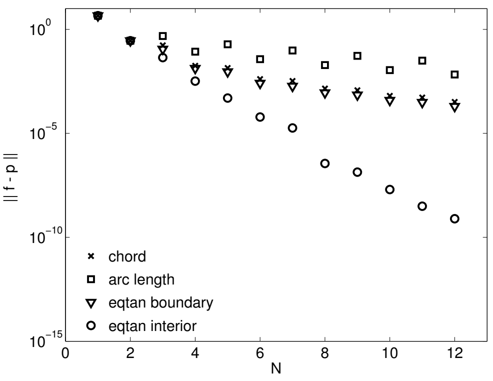


Figure 11: The interpolation error in Case 3. The surface is described by a function of low regularity, but the equal-tangents surface method finds a set of interpolation points that corresponds to interpolating a smooth reparametrization. This results in exponential convergence. The plot only extends to $N = 12$ because we are not able to maintain the same convergence rate for higher N.

surface of higher regularity than $\boldsymbol{f}$ – hence the increased convergence rate.

The exact surface is rotationally symmetric around the z-axis, and one may therefore expect the optimal mesh of interpolation points to be rotationally symmetric as well. The mesh in Figure 12b is not entirely symmetric, but one should be careful with concluding that a more symmetric mesh will give better approximation properties. It is in general impossible to predict the convergence rate from the mesh unless one knows which parametrization it interpolates.

Case 4

The last surface is

$$h(x, y) = \frac{\arctan(2x)\,\sin(2x + (y + 1)^2)}{1 + x^2 + y^2}, \qquad -1 \leq x, y \leq 1, \tag{5.4}$$

which is a little more complicated than the other surfaces. Both the chord method and the arc-length method give low exponential convergence rates. By applying the equal-tangents method on the boundary, we achieve a better convergence rate. When we apply equal-tangents in the interior as well, the convergence rate is improved even more. However, we see that the convergence rate

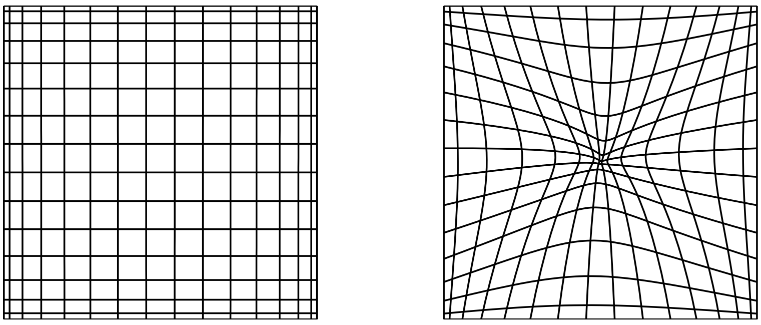

(a) Equal-tangents boundary **(b)** Equal-tangents surface

Figure 12: The interpolant at $N = 15$, projected onto the xy-plane. Left: solution obtained using the equal-tangents boundary method. Right: solution obtained using the equal-tangents surface method. The latter yields a clustering of interpolation points around the origin, since this is the position of the singularity in the exact surface. The non-linear reparametrization enables us to achieve exponential convergence.

decreases as N increases, most likely due to the failure of Newton's method to find the optimal solution.

6 Conclusion

High order interpolation of parametric curves and surfaces is an important part of the geometry representation in high order methods for solving PDEs in deformed rectangles and hexahedra, and the interpolation method chosen may have a big influence on the error in the numerical solution [2]. Still, the topic has received very little attention in the literature. On the other hand, a lot of work has been done on interpolation of parametric curves (and some on parametric surfaces) in the CAGD environment, but almost all of it concerns only *low* order polynomial interpolation.

Any parametric curve or surface can be reparametrized before being interpolated, and some reparametrizations will result in a smaller interpolation error than others. Finding the optimal reparametrization is in general a very difficult (and unsolved) problem. In the context of high order methods for solving PDEs, most authors settle with relatively simple and computationally inexpensive interpolation methods. The two most common methods of this kind is the chord method and the arc-length method, both of which are studied in this

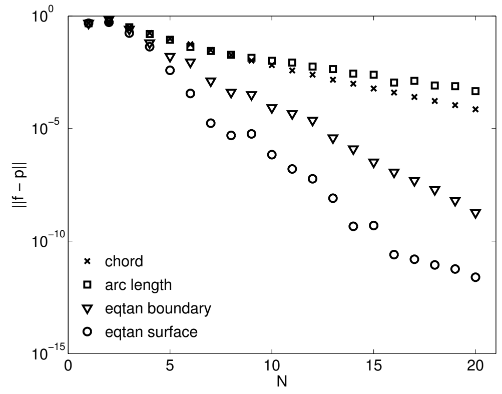


Figure 13: The interpolation error in Case 4.

paper. These methods are based on heuristic arguments and rarely yield significantly better results than classical interpolation (i.e., interpolation without reparametrization).

In order to construct better interpolation methods for parametric curves, a new interpolation method (the L^2-method) is introduced. It is based on doing a direct minimization of the interpolation error using Newton's method. The method works quite well for low polynomial degrees N (although it is expensive), but for high N it often degrades, since Newton's method is not sufficient for finding the *global* minimum of the objective function.

In the CAGD community, interpolation methods based on parametrization-independent quantities such as tangents, curvature and torsion have been suggested, and they are referred to as *geometric Hermite interpolation* methods. These methods are conjectured to be optimal in terms of approximation order (as defined for interpolation using a *fixed* N), but they are costly and difficult to implement, since they require systems of non-linear equations to be solved. Two methods in the family of geometric Hermite interpolation are proposed, and they yield very good results. Some of the results are assumed to be very close to optimal, since they are approximately equal to the solution found by the L^2-method for low N, and they often maintain a constant convergence rate until machine precision.

Some of the interpolation methods are extended to interpolation of parametric surfaces, and the relative performance of the different methods is often similar to the curve interpolation results. However, in the context of surfaces one can choose to apply costly interpolation methods only on the boundary, or one can do it over the entire surface. In some cases the former is enough to achieve a vast improvement from classical interpolation, but sometimes one needs to consider the entire surface to achieve any significant improvement.

Some important commonly known limitations of high order interpolation are challenged when considering interpolation of parametric curves and surfaces, because of the option of reparametrization. For example, curves and surfaces of low regularity can be interpolated to exponential convergence (as a function of N), as shown by examples in this paper. Whether this is possible for *all* curves and surfaces of low regularity is a topic for future work.

Another such limitation is the Runge phenomenon, which describes the unwanted oscillations displayed by the interpolant for certain functions and point distributions. Examples from this paper and from [2] show that this can be avoided by reparametrization. In fact, in all the numerical experiments considered, we have seen no examples of curves that could not be interpolated without oscillations. This includes the Runge function (viewed as a parametric curve), C^0 curves and functions with boundary layers (again, viewed as a parametric curve). Verifying (or disproving) the claim that any parametric curve can be interpolated by high order polynomials without oscillations is a topic for future work.

Acknowledgments

The work has been supported by the Research Council of Norway under contract 185336/V30. The author would like to thank Prof. E. M. Rønquist for many helpful comments and suggestions throughout the work on this paper.

Bibliography

[1] B. A. Barsky and T. D. DeRose. Geometric continuity of parametric curves: three equivalent characterizations. *IEEE Comput. Graphics Appl.*, 9(6):60–69, 1989.

[2] T. Bjøntegaard, E. M. Rønquist, and Ø. Tråsdahl. High order interpolation of curves in the plane. Technical report, Norwegian University of Science and Technology, http://www.math.ntnu.no/preprint/numerics/2009/N11-2009.pdf, 2009.

[3] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods, Fundamentals in Single Domains.* Springer, 2006.

[4] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods, Evolution to Complex Geometries and Applications to Fluid Dynamics.* Springer, 2007.

[5] X. D. Chen, W. Ma, and J. Zheng. Geometric interpolation method in R^3 space with optimal approximation order. *Comput.-Aided Des. Applic.*, 7(6):919–928, 2010.

[6] C. de Boor, K. Höllig, and M. Sabin. High accuracy geometric Hermite interpolation. *Comput. Aided Geom. Design*, 4(4):269–278, 1987.

[7] W. L. F. Degen. Best approximations of parametric curves by splines. In *Geometric modelling*, volume 8 of *Comput. Suppl.*, pages 59–73. Springer, Vienna, 1993.

[8] W. L. F. Degen. High accurate rational approximation of parametric curves. *Comput. Aided Geom. Design*, 10(3-4):293–313, 1993.

[9] W. L. F. Degen. Geometric Hermite interpolation – in memoriam Josef Hoschek. *Comput. Aided Geom. Design*, 22(7):573–592, 2005.

[10] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flow.* Cambridge University Press, 2002.

[11] Y. Y. Feng and J. Kozak. On G^2 continuous cubic spline interpolation. *BIT*, 37(2):312–332, 1997.

[12] Y. Y. Feng and J. Kozak. On spline interpolation of space data. *Mathematical Methods for Curves and Surfaces II, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville*, pages 167–174, 1998.

[13] M. S. Floater. An $O(h^{2n})$ Hermite approximation for conic sections. *Comput. Aided Geom. Design*, 14(2):135–151, 1997.

[14] W. J. Gordon and C. A. Hall. Construction of curvilinear co-ordinate systems and applications to mesh generation. *Internat. J. Numer. Methods Engrg.*, 7(4):461–477, 1973.

[15] K. Hollig and J. Koch. Geometric Hermite interpolation. *Comput. Aided Geom. Design*, 12(6):567–580, 1995.

[16] G. Jaklič, J. Kozak, M. Krajnc, V. Vitrih, and E. Žagar. On geometric Lagrange interpolation by quadratic parametric patches. *Comput. Aided Geom. Design*, 25(6):373–384, 2008.

[17] G. Jaklič, J. Kozak, M. Krajnc, and E. Žagar. On geometric interpolation by planar parametric polynomial curves. *Math. Comput.*, 76(260):1981–1993, 2007.

[18] Y. Maday and E. M. Rønquist. Optimal error analysis of spectral methods with emphasis on non-constant coefficients and deformed geometries. *Comput. Methods Appl. Mech. Engrg.*, 80(1-3):91–115, 1990.

[19] K. Mørken. On geometric interpolation of parametric surfaces. *Comput. Aided Geom. Design*, 22(9):838–848, 2005.

[20] K. Mørken and K. Scherer. A general framework for high-accuracy parametric interpolation. *Math. Comput.*, 66(217):237–260, 1997.

[21] C. W. Patty. *Foundations of Topology.* Jones & Bartlett Publishers, Inc., 2nd edition, 2009.

[22] A. Rababah. High order approximation method for curves. *Comput. Aided Geom. Design*, 12(1):89–102, 1995.

[23] A. Rababah. High accuracy Hermite approximation for space curves in $\mathbb{R}^d$. *J. Math. Anal. Appl.*, 325(2):920–931, 2007.

[24] R. Schaback. Interpolation with piecewise quadratic visually C^2 Bézier polynomials. *Comput. Aided Geom. Design*, 6(3):219–233, 1989.

[25] K. Scherer. Parametric polynomial curves of local approximation of order 8. *Curve and Surface Fitting: Saint-Malo 99*, pages 375–384, 2000.

[26] L. Xu and J. Shi. Geometric Hermite interpolation for space curves. *Comput. Aided Geom. Design*, 18(9):817–829, 2001.

[27] J. H. Yong and F. F. Cheng. Geometric Hermite curves with minimum strain energy. *Comput. Aided Geom. Design*, 21(3):281–301, 2004.

PAPER 4

HIGH ORDER NUMERICAL APPROXIMATION OF MINIMAL SURFACES

ØYSTEIN TRÅSDAHL AND EINAR M. RØNQUIST

Published in
Journal of Computational Physics,
Vol. 230, No. 12 (2011), pp. 4795–4810

HIGH ORDER NUMERICAL APPROXIMATION OF MINIMAL SURFACES

ØYSTEIN TRÅSDAHL AND EINAR M. RØNQUIST

Department of Mathematical Sciences,
Norwegian University of Science and Technology,
Trondheim, Norway

Abstract

We present an algorithm for finding high order numerical approximations of minimal surfaces with a fixed boundary. The algorithm employs parametrization by high order polynomials and a linearization of the weak formulation of the Laplace-Beltrami operator to arrive at an iterative procedure to evolve from a given initial surface to the final minimal surface. For the steady state solution we measure the approximation error in a few cases where the exact solution is known. In the framework of parametric interpolation, the choice of interpolation points (mesh nodes) is directly affecting the approximation error, and we discuss how to best update the mesh on the evolutionary surface such that the parametrization remains smooth. In our test cases we may achieve exponential convergence in the approximation of the minimal surface as the polynomial degree increases, but the rate of convergence greatly differs with different choices of mesh update algorithms. The present work is also of relevance to high order numerical approximation of fluid flow problems involving free surfaces.

Keywords: Minimal surfaces, mean curvature, free surface flow, evolutionary surfaces, mesh update techniques

1 Introduction

Surfaces of least area, called minimal surfaces, is a field of study that has intrigued scientists for many years and has been studied extensively [8, 21, 24]. Part of the interest stems from the fact that they are so easily realizable physically in the form of soap films, and for this reason they have been studied not only mathematically, but also physically for many years. An important early contribution came from the physicist J. A. F. Plateau, who studied them experimentally and determined some interesting geometric properties [25]. A breakthrough in the mathematical study of minimal surfaces came around 1930 with the works of J. Douglas [10] and T. Radó [26], who established some important theory around the existence of minimal surfaces.

The problem of finding exact minimal surfaces is very hard and in general unsolved. Only a few minimal surfaces have been found in closed form, and numerical methods are therefore an important tool. For non-parametric surfaces, methods have been proposed by Concus [6], Greenspan [16], Elcrat and Lancaster [14], Hoppe [19].

For parametric surfaces, the minimal surface problem has been solved with finite element methods by Dziuk and Hutchinson [12, 13], Brakke [3], Hinata *et al.* [17] and Wagner [28], whereas Coppin and Greenspan [7] use direct simulation of surface tension forces on a grid of marker particles. Chopp [5] has proposed a level set method which allows for natural handling of topological changes, but gives only linear convergence. It also employs a three-dimensional volume mesh, which is expensive and undesirable in our case, since we are only interested in a three-dimensional surface.

Minimal surfaces are smooth provided that the boundary curve is smooth. Spectral (or spectral element) methods based on high order polynomials should therefore, in principle, be very suitable numerical methods for such problems. However, better algorithms are still needed in order for high order methods to reach their full potential for computing minimal surfaces or for tracking time-dependent interfaces. We feel that the sensitivity to the choice of interpolation points have not been properly addressed in the literature before, and these challenges are particularly acute for high order methods.

The goal with this work is to find a high order numerical approximation of a minimal surface with a given boundary. We start off with an introduction to minimal surface conditions, and from there we derive a weak form of the problem. The problem is discretized using high order polynomials, and we show how it results in a nonlinear system that can be solved with an iterative method. The iterations make our solution an evolutionary surface, and it leads to the question of mesh update techniques, which will be discussed in some detail. These techniques are also needed in moving boundary problems with arbitrary Lagrangian-Eulerian (ALE) formulations [20], and the algorithms presented in this paper are also relevant in an ALE setting.

We conclude with numerical results showing the convergence properties of our method; these results are based on considering surfaces with analytically known solutions (the catenoid, the Scherk surface, and the Enneper surface). We also show examples of cases where the exact solution is unknown.

2 Problem formulation

Consider a two-dimensional surface Ω in $\mathbb{R}^3$ with a fixed boundary $\partial\Omega$, represented locally by a diffeomorphism $\boldsymbol{\varphi} : \hat{\Omega} \subset \mathbb{R}^2 \to \mathbb{R}^3$. The mapping $\boldsymbol{\varphi}$ satisfies the following set of three partial differential equations [29]

$$\Delta_\Omega \boldsymbol{\varphi} = 2\,\kappa\,\boldsymbol{n}, \tag{2.1}$$

where κ is the mean curvature, $\boldsymbol{n}$ is a unit surface normal vector, and Δ_Ω is the Laplace-Beltrami operator, a generalization of the Laplace operator to Riemannian manifolds. Minimal surfaces are characterized by the property that the mean curvature is everywhere zero. From (2.1) we conclude that minimal surfaces are solutions to the following system of equations,

$$\begin{aligned} \Delta_\Omega \boldsymbol{\varphi} &= \boldsymbol{0} \quad \text{in } \hat{\Omega}, \\ \boldsymbol{\varphi} &= \boldsymbol{\varphi}_0 \quad \text{on } \partial\hat{\Omega}, \end{aligned} \tag{2.2}$$

where $\boldsymbol{\varphi}_0$ is simply a parametrization of the boundary $\partial\Omega$. Note that $\boldsymbol{\varphi}$ has three components, one for each coordinate direction. Apart from the trivial case where $\partial\Omega$ lies in a plane, these partial differential equations are nonlinear.

As an example, consider the simpler case where the surface can be described by a function $z(x, y)$. Then (2.2) reduces to the (scalar) Plateau problem [15, 27]

$$\nabla \cdot \left(\frac{\nabla z}{\sqrt{1 + |\nabla z|^2}} \right) = 0,$$

with prescribed boundary conditions.

2.1 Weak formulation

A peculiar aspect of the minimal surface problem (2.2) is that the differential operator Δ_Ω is inextricably linked to the solution itself. This makes it very hard to solve the problem analytically except for in a few special cases.

We therefore start by considering the simpler, but related problem

$$\begin{aligned} \Delta_\Omega \hat{u} &= 0 \quad \text{in } \hat{\Omega}, \\ \hat{u} &= \hat{u}_0 \quad \text{on } \partial\hat{\Omega}, \end{aligned} \tag{2.3}$$

where $\hat{u}$ is a *scalar* function defined on $\hat{\Omega}$, and $\hat{u}_0$ is some given boundary condition. Here we assume the mapping $\boldsymbol{\varphi}$ to be known *a priori.* The problem can then be transformed to a problem defined on Ω by means of $\boldsymbol{\varphi}$. Assume that this mapping describes the surface in a Cartesian coordinate system,

$$\boldsymbol{\varphi}(\xi,\eta) = \begin{pmatrix} \varphi_1(\xi,\eta) \\ \varphi_2(\xi,\eta) \\ \varphi_3(\xi,\eta) \end{pmatrix} = \begin{pmatrix} x(\xi,\eta) \\ y(\xi,\eta) \\ z(\xi,\eta) \end{pmatrix}.$$

The Jacobian associated with this mapping is given as

$$J = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \\ z_\xi & z_\eta \end{pmatrix}.$$

Let $u = \hat{u} \circ \boldsymbol{\varphi}^{-1}$; the inverse $\boldsymbol{\varphi}^{-1}$ exists since $\boldsymbol{\varphi}$ is a diffeomorphism. In the Cartesian coordinate system the Laplace-Beltrami operator simplifies to the well-known Laplace operator in $\mathbb{R}^3$,

$$\Delta_\Omega \hat{u} = \Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}.$$

Hence, solving (2.3) is equivalent to solving the problem

$$\begin{aligned} \Delta u &= 0 \quad \text{in } \Omega, \\ u &= u_0 \quad \text{on } \partial\Omega, \end{aligned} \tag{2.4}$$

where $u_0 = \hat{u}_0 \circ \boldsymbol{\varphi}^{-1}$, and $\partial\Omega$ is the image of $\partial\hat{\Omega}$ under the mapping $\boldsymbol{\varphi}$. The change of variables has moved the complexity from the operator to the domain itself. However, one advantage of this transformation is that the derivation of a weak formulation of (2.4) becomes easy. The Galerkin problem is given as: find $u \in Y^D \equiv \{v \in H^1(\Omega) \bigm| v|_{\partial\Omega} = u_0\}$ such that

$$\int_\Omega (\nabla v)^T \nabla u \,\mathrm{d}\Omega = 0 \qquad \forall v \in Y \equiv H_0^1(\Omega), \tag{2.5}$$

where $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})^T$ is the standard gradient operator in $\mathbb{R}^3$. The integral (2.5) is not readily evaluated since Ω is a curved surface. We therefore apply a change of variables to transform it back to the reference domain $\hat{\Omega}$. An infinitesimal surface area $\mathrm{d}\Omega$ on the curved surface can be expressed in the reference variables as

$$\mathrm{d}\Omega = g \,\mathrm{d}\hat{\Omega},$$

where the metric g is defined in terms of the Jacobian J of $\boldsymbol{\varphi}$ by

$$g = \sqrt{\det(J^T J)}.$$

Gradients on Ω are related to gradients in the two-dimensional reference domain $\hat{\Omega}$ through the Jacobian $\tilde{J}$ of $\boldsymbol{\varphi}^{-1}$. Written out, we have

$$\nabla u = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} = \begin{pmatrix} \hat{u}_\xi \xi_x + \hat{u}_\eta \eta_x \\ \hat{u}_\xi \xi_y + \hat{u}_\eta \eta_y \\ \hat{u}_\xi \xi_z + \hat{u}_\eta \eta_z \end{pmatrix} = \begin{pmatrix} \xi_x & \eta_x \\ \xi_y & \eta_y \\ \xi_z & \eta_z \end{pmatrix} \begin{pmatrix} \hat{u}_\xi \\ \hat{u}_\eta \end{pmatrix} = \tilde{J}^T \hat{\nabla} \hat{u},$$

where $\hat{\nabla} = (\frac{\partial}{\partial \xi}, \frac{\partial}{\partial \eta})^T$ is the two-dimensional gradient on the reference domain. Note the reappearance of the function $\hat{u}$; it is the same as in (2.3) since we use the particular inverse mapping $\boldsymbol{\varphi}^{-1}$.

The integral in (2.5) can now be expressed in reference variables as

$$\begin{aligned} \int_\Omega (\nabla v)^T \nabla u \, \mathrm{d}\Omega &= \int_{\hat{\Omega}} (\tilde{J}^T \hat{\nabla} \hat{v})^T \tilde{J}^T \hat{\nabla} \hat{u} \, g \, \mathrm{d}\hat{\Omega} \\ &= \int_{\hat{\Omega}} (\hat{\nabla} \hat{v})^T \tilde{J} \tilde{J}^T \hat{\nabla} \hat{u} \, g \, \mathrm{d}\hat{\Omega}. \end{aligned}$$

We can eliminate the dependence on the inverse mapping $\boldsymbol{\varphi}^{-1}$ by using the fact that the two Jacobian matrices J and $\tilde{J}$ are related as

$$\tilde{J}\tilde{J}^T = (J^T J)^{-1}.$$

The resulting integral can then be expressed as the bilinear form

$$a(\hat{v}, \hat{u}) = \int_{\hat{\Omega}} k \; (\hat{\nabla} \hat{v})^T G \, \hat{\nabla} \hat{u} \, g \, \mathrm{d}\hat{\Omega}, \tag{2.6}$$

where $G = (J^T J)^{-1}$ and $k = 1$ (the reason for introducing the parameter k will be explained below). The matrix G is obviously symmetric, and it is also positive definite, since, for all $\boldsymbol{q} \in \mathbb{R}^2$, $\boldsymbol{q} \neq \boldsymbol{0}$,

$$\boldsymbol{q}^T G^{-1} \boldsymbol{q} = \boldsymbol{q}^T J^T J \boldsymbol{q} = (J\boldsymbol{q})^T (J\boldsymbol{q}) > 0.$$

Hence, $a(\hat{v}, \hat{v}) > 0$ for all $\hat{v} \in H_0^1(\hat{\Omega})$, $\hat{v} \neq 0$, and thus the bilinear form $a(\cdot,\cdot)$ is symmetric and positive definite (SPD).

We now introduce the space $\hat{Y}^D = \{\hat{w} \in H^1(\hat{\Omega}) \mid \hat{w}|_{\partial\hat{\Omega}} = \hat{u}_0\}$. The weak formulation of (2.3) is then given as: find $\hat{u} \in \hat{Y}^D$ such that

$$a(\hat{v}, \hat{u}) = 0 \qquad \forall \hat{v} \in H_0^1(\hat{\Omega}). \tag{2.7}$$

By comparing the strong formulations (2.2) and (2.3), we see that a weak formulation of the former is just a vector version of the latter: find $\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \varphi_3)^T = (x, y, z)^T \in \hat{X}^D \equiv \{\hat{\boldsymbol{w}} \in (H^1(\hat{\Omega}))^3 \mid \hat{\boldsymbol{w}}|_{\partial\hat{\Omega}} = \boldsymbol{\varphi}_0\}$ such that

$$a(\hat{v}_i, \varphi_i) = 0 \qquad \forall \hat{v}_i \in H_0^1(\hat{\Omega}), \quad i = 1, 2, 3. \tag{2.8}$$

The notation in (2.8) hides an important fact: $a(\cdot, \cdot)$ is *not* a bilinear form for this particular argument because of the hidden dependence of $\boldsymbol{\varphi}$ in G and g. We will get back to this problem shortly.

2.2 Relation to free surface flow

There is a close link between minimal surfaces and free surface flows that deserves some attention here, particularly because the mesh update techniques described later in this paper also have relevance to the numerical treatment of such flows.

For free surface flow, the surface tension represents a molecular force that acts to minimize the free surface at all time. Consider a three-dimensional unsteady flow with a free surface Ω. The total stress force acting on the free surface is the sum of a normal component $\boldsymbol{F}_n$ and a tangential component $\boldsymbol{F}_t$ and is given by [23]

$$\boldsymbol{F} = \boldsymbol{F}_n + \boldsymbol{F}_t = \gamma\kappa\boldsymbol{n} + \nabla_\Omega\gamma,$$

where γ is the surface tension, $\boldsymbol{n}$ is the outward unit normal vector and ∇_Ω is the surface gradient. The free surface flow is described by the Navier-Stokes equations, for which surface tension forces are represented by the boundary conditions

$$\begin{aligned} n_i\sigma_{ij}n_j &= \gamma\kappa, \\ t_i\sigma_{ij}n_j &= t_i(\nabla_\Omega\gamma)_i, \end{aligned}$$

where n_i and n_j are components of the unit normal vector $\boldsymbol{n}$, t_i is a component of a unit tangent vector $\boldsymbol{t}$, and σ_{ij} is a component of the stress tensor. Summation over repeated indices is assumed. A natural imposition of the free surface boundary conditions in a variational formulation of the Navier-Stokes equations yields the integral

$$\int_\Omega v_i\sigma_{ij}n_j \, \mathrm{d}\Omega, \qquad i = 1, 2, 3,$$

where v_i is a test function. This term includes both normal and tangential contributions. In [18] it was shown that this integral can be expressed as

$$\int_{\Omega} v_i \sigma_{ij} n_j \,\mathrm{d}\Omega = -\int_{\hat{\Omega}} \gamma\, \hat{v}_{i,\alpha}\, g_i^{\alpha}\, g \,\mathrm{d}\hat{\Omega}, \qquad i = 1,2,3,$$

where $\hat{v}_{i,\alpha}$ denotes the partial derivative of $\hat{v}_i = v_i \circ \boldsymbol{\varphi}$ with respect to the reference variable r^{α} (here, $r^1 = \xi$ and $r^2 = \eta$), and g_i^{α} is the i'th component of the *contravariant base-vector* $\boldsymbol{g}^{\alpha}$. From differential geometry we have [22]

$$\boldsymbol{g}^{\alpha} = g^{\alpha\beta} \boldsymbol{g}_{\beta}$$

where $g^{\alpha\beta}$ is the *contravariant metric tensor*, which in matrix notation is nothing but our matrix $G = (J^T J)^{-1}$. The vector $\boldsymbol{g}_{\beta}$ is the *covariant base-vector* and is defined as the partial derivative of the mapping with respect to the reference variable r^{β}, i,e., $\boldsymbol{g}_{\beta} = \boldsymbol{\varphi}_{,\beta}$. Thus, $\boldsymbol{g}_{\beta}$, $\beta = 1,2$, represent two vectors spanning the tangent plane at a particular point on the surface. Inserting this into the integral yields

$$\begin{aligned}
\int_{\hat{\Omega}} \gamma\, \hat{v}_{i,\alpha}\, g_i^{\alpha}\, g \,\mathrm{d}\hat{\Omega} &= \int_{\hat{\Omega}} \gamma\, \hat{v}_{i,\alpha}\, g^{\alpha\beta} \varphi_{i,\beta}\, g \,\mathrm{d}\hat{\Omega} \\
&= \int_{\hat{\Omega}} \gamma\, (\hat{\nabla} \hat{v}_i)^T G\, \hat{\nabla} \varphi_i\, g \,\mathrm{d}\hat{\Omega} \\
&= a(\hat{v}_i, \varphi_i), \qquad i = 1,2,3.
\end{aligned}$$

Hence, it is interesting to observe that the contributions from the free surface boundary conditions (both normal and tangential) can be expressed by the form (2.6) in the particular case with $\hat{u} = \varphi_i$, $i = 1,2,3$, and with $k = 1$ replaced by $k = \gamma$. Note that for surface-tension-driven flows (Marangoni-type problems), γ is not a constant, but is still a positive quantity over the free surface.

2.3 Linearization and iterative scheme

The system (2.7) is linear in the unknown $\hat{u}$ and is readily solved with a finite or spectral (element) method. It also has the advantage that the bilinear form is SPD, so that the corresponding algebraic system can easily be solved using the Conjugate Gradients (CG) method.

The problem (2.8) is nonlinear, but can be solved by introducing an iterative scheme. At each iteration we start with a *known* surface Ω^n, parametrized by $\boldsymbol{\varphi}^n$, and we move to the next iteration by letting

$$\boldsymbol{\varphi}^{n+1} = \boldsymbol{\varphi}^n + \Delta \boldsymbol{\varphi}^{n+1},$$

where $\Delta\boldsymbol{\varphi}^{n+1}$ is a vector field with components $\Delta\varphi_i^{n+1}$, $i = 1, 2, 3$, that are the solutions of

$$a(\hat{v}_i, \varphi_i^n + \Delta\varphi_i^{n+1}) = 0, \qquad i = 1, 2, 3. \tag{2.9}$$

Here, $a(\cdot,\cdot)$ represents an integral over the unknown surface Ω^{n+1}, and the unknown $\Delta\boldsymbol{\varphi}^{n+1}$ enters into the nonlinear terms G and g and makes the entire system nonlinear. However, assuming that the update $\Delta\boldsymbol{\varphi}^{n+1}$ is relatively small, we can approximate $a(\cdot,\cdot)$ by an integral over the *known* surface Ω^n. We do this by "freezing" G and g at the values G^n and g^n that are computed from the current mapping $\boldsymbol{\varphi}^n$. This approximation yields a bilinear form

$$a^n(\hat{v}, \hat{w}) = \int_{\hat{\Omega}} (\hat{\nabla}\hat{v})^T \, G^n \, \hat{\nabla}\hat{w} \, g^n \, \mathrm{d}\hat{\Omega} \tag{2.10}$$

which is also SPD, since G^n is an SPD matrix and we consider $\hat{v} \in H_0^1(\hat{\Omega})$. Note that we have omitted k in (2.10) since $k = 1$. The linearized version of (2.9) is then

$$a^n(\hat{v}_i, \Delta\varphi_i^{n+1}) = -a^n(\hat{v}_i, \varphi_i^n), \qquad i = 1, 2, 3, \tag{2.11}$$

which is suitable for a numerical discretization.

3 Discretization

For the numerical solution of the Galerkin problem (2.8) we apply a spectral discretization based on high order polynomials [4]. For simplicity, we consider a pure spectral method here, i.e., $\hat{\Omega} = (-1, 1)^2$; the extension to spectral elements is straight-forward and standard. The relevant discrete function spaces are

$$\begin{aligned}
\hat{X}_N &= \{\hat{\boldsymbol{w}} \in H_0^1(\hat{\Omega})^3 \mid \hat{\boldsymbol{w}} \in \mathbb{P}_N(\hat{\Omega})^3\}, \\
\hat{X}_N^D &= \{\hat{\boldsymbol{w}} \in H^1(\hat{\Omega})^3 \mid \hat{\boldsymbol{w}} \in \mathbb{P}_N(\hat{\Omega})^3 \text{ and } \hat{\boldsymbol{w}} = \boldsymbol{\varphi}_0 \text{ on } \partial\hat{\Omega}\}.
\end{aligned}$$

As a basis for these spaces we choose the tensor-product Lagrangian interpolants through the Gauss-Lobatto-Legendre (GLL) points $\xi_0, \ldots, \xi_N$. If ψ_N represents a component of an element in $\hat{X}_N$ or $\hat{X}_N^D$, this component is expressed as

$$\psi_N(\xi, \eta) = \sum_{i=0}^{N} \sum_{j=0}^{N} \psi_{ij} \ell_i(\xi) \ell_j(\eta), \tag{3.1}$$

where some of the basis coefficients are given by the prescribed boundary values. This enables us to compute partial derivatives easily via differentiation matrices, and we can evaluate all integrals with sufficient accuracy with GLL quadrature. Applying quadrature leads to the definition of the discrete version of $a(\cdot,\cdot)$,

$$a_N(\hat{v},\hat{w}) = \sum_{\alpha=0}^{N}\sum_{\beta=0}^{N} \rho_\alpha \rho_\beta \left((\hat{\nabla}\hat{v})^T G \hat{\nabla}\hat{w}\, g \right)\Big|_{\alpha\beta},$$

where $\rho_\alpha, \alpha = 0,\dots,N$, are the GLL quadrature weights and the subscript $\alpha\beta$ means that we evaluate the integrand in the tensor-product GLL point (ξ_α, ξ_β). The discrete problem, in vector notation, is then: find $\boldsymbol{\varphi}_N \in \hat{X}_N^D$ such that

$$\boldsymbol{a}_N(\hat{\boldsymbol{v}}_N, \boldsymbol{\varphi}_N) = \mathbf{0} \qquad \forall \hat{\boldsymbol{v}}_N \in \hat{X}_N. \tag{3.2}$$

The boundary conditions are met by choosing the nodal values of $\boldsymbol{\varphi}_N$ (corresponding to the basis coefficients in (12)) to be interpolation points on the boundary $\partial\Omega$.

By applying discretization to the iterative scheme (2.11) we arrive at an algebraic system

$$A^n \Delta\boldsymbol{\phi}^{n+1} = -A^n \boldsymbol{\phi}^n \tag{3.3}$$

where A^n is the discrete, linearized Laplace-Beltrami operator, $\boldsymbol{\phi}^n$ is a vector containing the nodal values of $\boldsymbol{\varphi}_N$ at iteration level n, and $\Delta\boldsymbol{\phi}^{n+1}$ is a vector of the change in the nodal values of $\boldsymbol{\varphi}_N$. Since the bilinear form (2.10) is SPD, the matrix A^n is SPD, and the system is readily solved with CG iterations.

3.1 Mesh construction

Since the Lagrangian interpolants satisfy $\ell_j(\xi_i) = \delta_i^j$ at the GLL points, the basis coefficients in (3.1) represent the nodes on a curvilinear mesh on the numerical surface. In the context of polynomial interpolation, i.e., if $\psi_N = I_N\psi$ for a given function ψ, then the mesh nodes are defined by evaluating ψ in a predefined set of interpolation points,

$$\psi_{ij} = \psi(\xi_i, \xi_j),$$

in our case the tensor-product GLL points. In interpolation of parametric surfaces, each parametric function is interpolated separately. From basic interpolation theory we know that for a scalar function $\hat{u} \in H^\sigma(\hat{\Omega})$ the interpolation error is bounded by [4]

$$||\hat{u} - I_N\hat{u}||_{L^2(\hat{\Omega})} \leq CN^{-\sigma}||\hat{u}||_{H^\sigma(\hat{\Omega})}, \tag{3.4}$$

where N is the polynomial degree and C is a constant. In our case this holds for each of the parametric functions. Hence, the accuracy of the interpolation of the surface depends on the regularity of the parametric functions.

As an example, consider the catenoid which is a minimal surface. A natural parametrization of a catenoid of height H and "waist" radius R_m (radius at the midpoint between the two boundary circles) is [9]

$$\begin{aligned}\varphi_1(\xi,\eta) &= R_m \cosh\left(\frac{H\eta}{2R_m}\right)\cos(\pi\xi),\\ \varphi_2(\xi,\eta) &= R_m \cosh\left(\frac{H\eta}{2R_m}\right)\sin(\pi\xi),\\ \varphi_3(\xi,\eta) &= \frac{H}{2}\eta,\end{aligned}\tag{3.5}$$

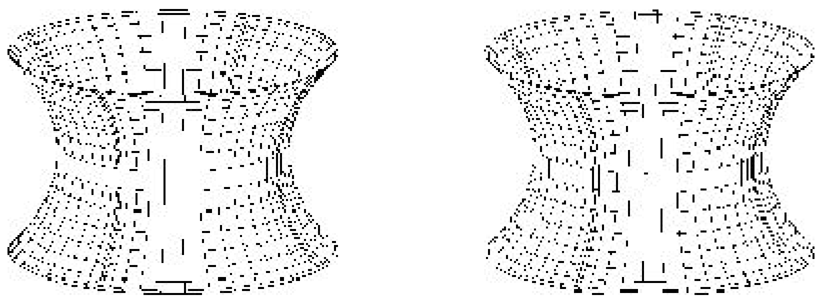
where $-1 \leq \xi, \eta \leq 1$. However, consider also the alternative parametrization

$$\begin{aligned}\tilde{\varphi}_1(\xi,\eta) &= R_m \cosh\left(\frac{H\eta}{2R_m}\right)\xi,\\ \tilde{\varphi}_2(\xi,\eta) &= \pm R_m \cosh\left(\frac{H\eta}{2R_m}\right)\sqrt{1-\xi^2},\\ \tilde{\varphi}_3(\xi,\eta) &= \frac{H}{2}\eta,\end{aligned}\tag{3.6}$$

again with $-1 \leq \xi, \eta \leq 1$. It is easy to see that $\boldsymbol{\varphi}$ and $\tilde{\boldsymbol{\varphi}}$ represent the same surface. However, when we approximate them with polynomial interpolation, $\boldsymbol{\varphi}$ yields a GLL distribution in arc length of interpolation points along the two boundary circles, whereas $\tilde{\boldsymbol{\varphi}}$ yields a chord distribution [2]. Figure 1 shows the two meshes generated by interpolating the two parametrizations in the case of using four spectral elements and a polynomial degree $N = 15$. In this case the surface is first decomposed into four deformed quadrilateral elements and the reference domain $(-1,1)^2$ is mapped to each of these four spectral elements.

In order to measure the interpolation error we consider the distance between two surfaces measured along the surface normal to one of the surfaces, in our case the interpolant. To find this distance requires an iterative procedure, but in cases where a non-parametric representation of the exact surface is known this is straightforward and can be accomplished with Newton's method. The interpolation error is then defined as

$$||\boldsymbol{\varphi} - \boldsymbol{\varphi}_N|| = \left(\frac{\int_{\Omega_N} e_N^2 \,\mathrm{d}\Omega_N}{\int_{\Omega_N} \mathrm{d}\Omega_N}\right)^{1/2}\tag{3.7}$$



(a) Interpolation based on (3.5) **(b)** Interpolation based on (3.6)

Figure 1: The two meshes are generated by interpolating a multi-domain version of (3.5) and (3.6); here, four spectral elements are used. The meshes look almost the same.

where e_N is the distance along the surface normal from the interpolant to the exact surface. The integrals over the numerical surface Ω_N are evaluated using Gauss quadrature of high degree.

The two different parametrizations (3.5) and (3.6) are now compared by considering a multi-domain version based on four spectral elements. We define the height of the catenoid to be $H = 2$ such that the radius of the boundary circles are $R = R_m \cosh \frac{1}{R_m}$. The catenoid is only stable if $R/H > 0.755$ [8]; we safely choose $R = 1.6$.

As expected, both mappings yield exponential convergence; see Figure 2. However, this example illustrates the sensitivity to the particular mapping used: despite the fact that the difference between the two grids is not noticeable, the convergence rate is quite different.

Our main objective is not to interpolate a given surface since we do not assume a priori knowledge of the exact solution. However, in the case of the catenoid, the solution of (3.2) will be a polynomial approximation (although not an interpolation), and the results from this section will then serve as a reference.

3.2 Mesh update algorithms

Our iterative scheme can be stated as a two-step algorithm:

1. Solve (3.3) for $\Delta\phi^{n+1}$.

2. Update the geometry accordingly.

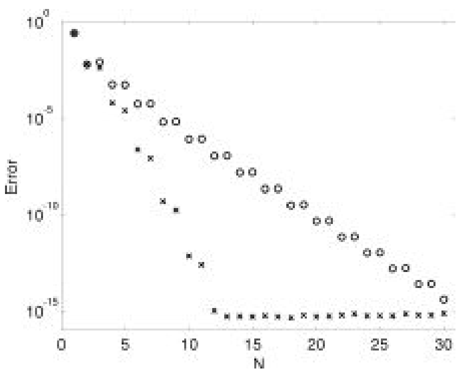


Figure 2: A multi-domain version of the two different parametrizations (3.5) ($\times$) and (3.6) ($\circ$) are interpolated in the tensor-product GLL points; here, four spectral elements are used. The interpolation error is measured in a discrete version of the norm (3.7). Both parametrizations yield exponential convergence, but there is a significant difference in the convergence rate.

The most straightforward implementation of the second step is

$$\boldsymbol{\phi}^{n+1} = \boldsymbol{\phi}^{n} + \Delta\boldsymbol{\phi}^{n+1}. \tag{3.8}$$

However, this is not the only option, as we can choose to add small tangential components to $\Delta\boldsymbol{\phi}^{n+1}$ to obtain a different mesh in the next configuration $\boldsymbol{\varphi}_N^{n+1}$. This can be used to control the distribution of the mesh nodes during the iterations and retain a "good" mesh, i.e., a mesh that corresponds to a smooth mapping $\boldsymbol{\varphi}_N^{n+1}$.

Our main problem is that we do not know the surface we are approximating, so we do not know which mesh gives us the best representation of the next state of the surface. Retaining an optimal mesh in an evolutionary geometry is a very complicated and generally unsolved problem [1]. It makes it even more difficult that the problem of optimal representation of a given *stationary* parametric surface remains unsolved. Numerical investigation of the problem is made difficult by the dearth of geometrically interesting evolutionary surface problems for which the exact solutions are known at all times.

For the numerical results we present later, we will compare three different mesh update algorithms which highlight some of the important aspects of evolutionary surfaces and with particular relevance to high order discretization methods. One algorithm is the straightforward one (3.8), which we will refer to as the *Lagrangian update*. The method is the simplest of all since it does not require any post-processing after solving (3.3), but we have little control over the regularity of the resulting mapping.

A second algorithm is defined by removing all tangential components from $\Delta\boldsymbol{\phi}^{n+1}$ (where the tangents are computed numerically based on $\boldsymbol{\varphi}_N^n$) and moving the mesh nodes in a direction normal to the current surface Ω_N^n. This *normal update* approach can be motivated from the fact that it is the displacement in the normal direction which changes the shape of the surface (similar to the kinematic condition for free surface flows). It is implemented by finding the unit surface normal at each mesh point (numerically) and then projecting the update $\Delta\boldsymbol{\phi}^{n+1}$ onto these vectors. The algorithm is also discussed in [1].

The third algorithm can be viewed as a compromise between the two previous ones. If the normal component of $\Delta\boldsymbol{\phi}^{n+1}$ is larger than the tangential component in all the nodes on the computational surface, we do a Lagrangian update. Otherwise, we scale the tangential component everywhere by the largest factor such that the tangential component is never larger than the normal component. We denote this as a *restricted Lagrangian update*.

Besides these three, we will also consider a few special mesh update algorithms customized for the particular test case at hand.

3.3 Comparison with mean curvature flow

Finding minimal surfaces can also be done by solving the *time-dependent* PDE

$$\frac{\partial \boldsymbol{\varphi}}{\partial t} = \Delta_{\Omega}\boldsymbol{\varphi}, \tag{3.9}$$

over a large time interval $[T_0, T]$. If the solution reaches a steady state within $t = T$, then that is necessarily also a solution to (2.2) and hence a minimal surface. This problem is called *mean curvature flow*, since the time-derivative of the solution points in the direction of the mean curvature. It has been studied numerically with a finite element method in [11].

A numerical treatment of (3.9) with a spectral element method will involve much the same ingredients as we have seen in the previous sections. The starting point is a weak formulation of the PDE, and spatial discretization is applied based on high order polynomial representations. This results in the semi-discrete system

$$\frac{\partial}{\partial t} B\boldsymbol{\phi} = -A\boldsymbol{\phi},$$

where B is the mass matrix and A is the discrete Laplace-Beltrami operator. We would prefer to treat this problem with an implicit time integration method due to the step restrictions induced by A. However, we have the same problem with the nonlinear factors G and g as before, so in order for the system to be solvable with CG iterations, these terms must be treated explicitly. Hence the system will never be fully implicit. Actually, this imposes a relatively severe time step restriction which makes the method inefficient if we are only interested in steady state solutions.

There is also another drawback with the time-dependent problem compared to our iterative scheme, namely the lack of control over the mesh. In mean curvature flow $\boldsymbol{\phi}^{n+1}$ is fully determined by the algebraic system we solve at each time-step, and we may have to re-mesh in order to avoid severely distorted meshes and possible breakdowns.

4 Numerical Results

4.1 The Catenoid

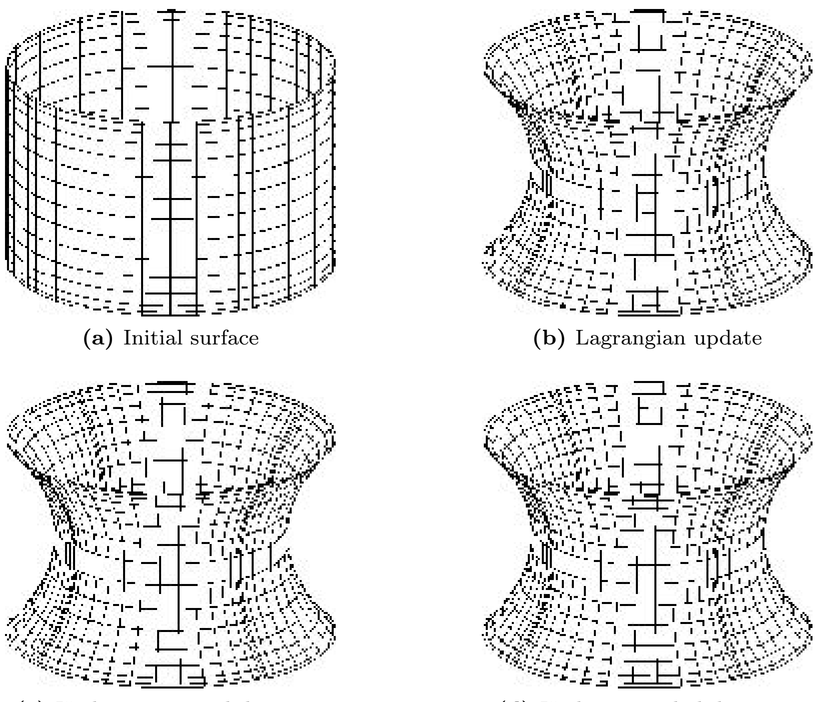
We first revisit the surface from Section 3.1, and use the same parameters H and R in order to make the results comparable to those in Figure 2.

The iterative scheme requires the definition of an initial surface. A natural starting point is the cylinder with radius R. This surface is most naturally parametrized by trigonometric functions for x and y and an affine mapping for z, which yields a conformal mapping from the reference domain. We use four spectral elements, which can be recognized in the mesh-structure in Figure 3a.

The chosen parametrization of the initial surface consists of analytic functions and is very suitable for polynomial interpolation. It is also relatively similar to a good parametrization of the catenoid (see (3.5)), so if the chosen mesh update algorithm yields small distortions of the mesh, then we can expect something close to an optimal polynomial representation of the catenoid at steady state.

For the catenoid we also consider a customized mesh update algorithm: we restrict the mesh update to the radial direction by projecting $\Delta\boldsymbol{\phi}^{n+1}$ onto the unit radial vector towards the z-axis. Hence, the affine mapping along the z-axis of the initial surface will be retained during the iterations.

The difference between the different mesh update algorithms is hardly visible in the steady state solutions shown in Figure 3. Still, there is a significant difference in the convergence rate for the different algorithms; see Figure 4. The



(a) Initial surface **(b)** Lagrangian update **(c)** Update in normal direction **(d)** Update in radial direction

Figure 3: The initial surface (a) is a cylinder represented using four spectral elements and a polynomial degree $N = 15$. The steady state solutions are displayed for the three different mesh update algorithms: (b) Lagrangian update (3.8); (c) update in the direction of the surface normal; and (d) update in the radial direction. The solution obtained using the restricted Lagrangian update algorithm is essentially the same as shown in (d).

radial update algorithm will, by construction, end up in almost exactly the mesh defined by interpolating (3.5), and therefore converges with approximately the same rate. Both the pure and the restricted Lagrangian update algorithms need about twice the polynomial degree to reach machine precision, while the normal vector update algorithm needs about three times the polynomial degree. The relatively poor performance of the latter is caused by a slight movement of the mesh points towards the boundary circles, thus resulting in a nonaffine mapping $\varphi_{3,N}(\xi, \eta)$.

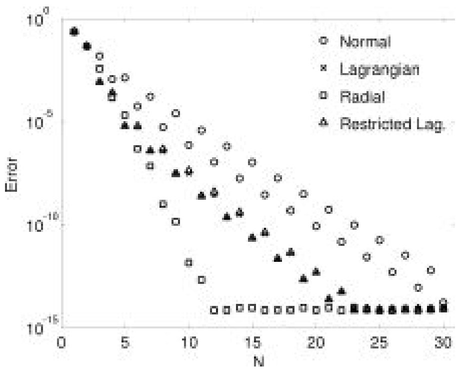


Figure 4: Error in the steady state solution for the catenoid problem, measured in a discrete version of the norm (3.7), as a function of the polynomial degree, N. The initial state is a cylinder. Different mesh update algorithms yield different levels of accuracy for a given N.

However, there is a problem with the pure Lagrangian update algorithm that is not shown in Figure 4: the algorithm is unstable. After the steady state is reached, small numerical errors in the solution of (2.10) keep causing small perturbations in the mesh, and the surface evolves *away* from the steady state. This is illustrated in Figure 5, which displays the error as a function of the iteration number at a fixed polynomial degree $N = 15$. We see that all four update algorithms converge at approximately the same rate, reaching the steady state within 100 iterations; the number of iterations needed depends on the level of accuracy reached. The solutions corresponding to the restricted Lagrangian, the normal, and the radial update algorithms remain at steady state, and the size of the updates $\Delta\boldsymbol{\varphi}^{n+1}$ remain at machine precision level. The pure Lagrangian update algorithm, on the other hand, sees an *increase* in the error from around $n = 200$, and from there it continues to increase until the surface collapses. This behavior is also seen for other values of N.

This first example had a clear symmetry in the mapping of the initial surface. To show that the results do not depend on this symmetry, we repeat the numerical experiment, but with element boundaries spiraling around the cylinder; see Figure 6. It is important to note that this “twisting” of the cylinder must be done in a smooth fashion; if the element boundaries cannot be represented by parametric functions of high regularity, then the representation of the entire surface will suffer. By twisting the cylinder like a spiral with a constant “angle

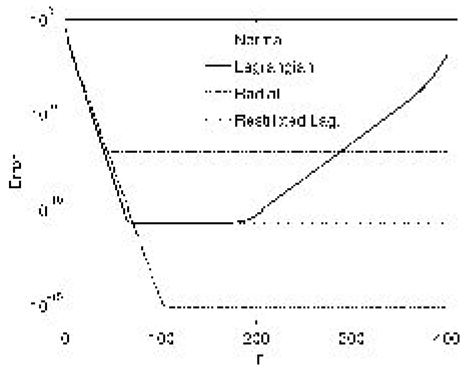
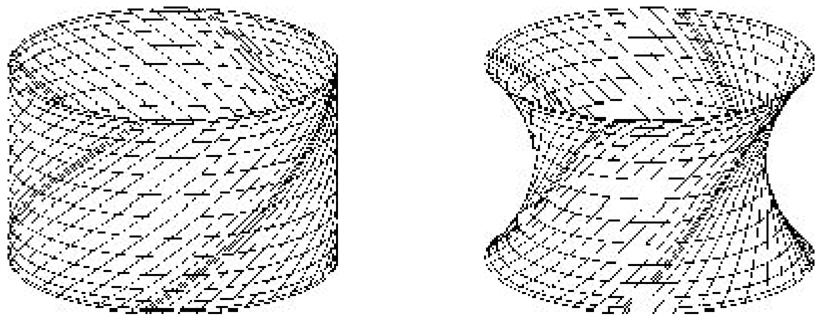


Figure 5: Error in the stationary solution, measured in the same norm, but now as a function of the iteration level n at a fixed polynomial degree $N = 15$. The restricted Lagrangian, the normal, and the radial mesh update algorithms are all stable at steady state, whereas the pure Lagrangian update algorithm yields small perturbations of the mesh at steady state, which after many iterations cause large mesh distortions.

of rotation", we retain a smooth parametrization.

The similarity between the cylinder and the catenoid again makes the radial update algorithm the best alternative. Figure 7 shows the same relation between the mesh update algorithms as we saw with the plain parametrization of the cylinder in Figure 4. Note that the Lagrangian update algorithm is still unstable with this new mesh configuration.

We now investigate the impact of starting "further away" from the minimal surface (in terms of the norm (3.7)). Let the initial surface be the rotational surface with radius $R(z) = 1.6 + \frac{1}{2}(1 + z)(1 - z)$. This yields a surface that resembles a sphere with parts of the upper and lower hemispheres cut off; see Figure 8a. The parametrization includes an affine mapping $\varphi_{3,N}(\xi, \eta)$, meaning that the radial update scheme should converge at exactly the same speed as before. On the other hand, we expect the normal update scheme to be affected, since the normal vectors on the initial surface are no longer horizontal. Figure 8b confirms this, showing that the mesh nodes have been displaced vertically during the iterations. This also affects the error in the steady state solution; see Figure 9. The normal update algorithm seems to stop converging when the error caused by the mesh distortion becomes dominant. The Lagrangian update algorithm converges at the same rate as before, but is still unstable.



(a) Initial surface **(b)** Minimal surface

Figure 6: The initial surface mesh is "twisted" such that each element boundary spirals along the cylinder wall. The steady state is a catenoid with a "twisted" mesh. The solution shown is obtained using the restricted Lagrangian mesh updates.

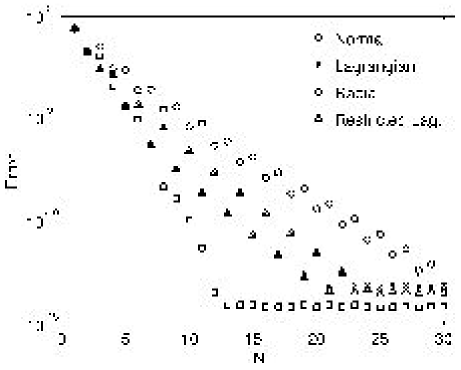


Figure 7: Error in the steady state solution for the catenoid with a "twisted" mesh. The performance of the different mesh update algorithms is almost exactly the same as for the plain parametrization; see Figure 4.

4.2 Scherk's fifth surface

This surface can be represented non-parametrically by [9]

$$\sin(z) = \sinh(x)\sinh(y). \tag{4.1}$$

In the following, we consider this surface for x and y in the range -0.8 to 0.8. This allows us to represent the surface as a function $z(x, y)$ while at the same

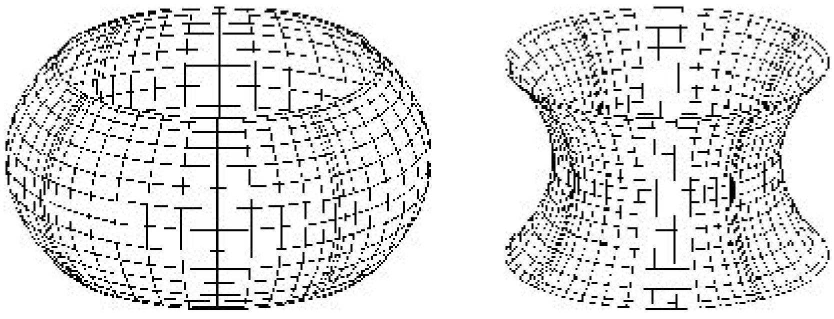

(a) Initial surface **(b)** Minimal surface

Figure 8: (a) The initial surface resembles part of a sphere; the radius is a parabola as a function of z. The surface normals on this initial surface have a non-zero component in the z direction. (b) The minimal surface obtained with the normal update algorithm. The vertical component of the updates has changed the vertical distribution of the mesh nodes, resulting in a nonaffine mapping $\varphi_{3,N}(\xi,\eta)$.

time avoiding $\frac{\partial z}{\partial x}$ and $\frac{\partial z}{\partial y}$ becoming unbounded and the corner angles going to zero. We represent this surface using a pure spectral method.

We start the iteration using the initial surface shown in Figure 10. Note that the projection of the initial mesh to the xy-plane is still a (affinely mapped) tensor-product GLL mesh.

The Lagrangian, the restricted Lagrangian, and the normal mesh update algorithms are "general purpose" algorithms which can be used on any surface in our current framework. As for the catenoid, we also now construct a customized mesh update algorithm by restricting the updates to the z-direction. This ensures that the projected mesh will remain a tensor-product GLL mesh during the iterations. Since the minimal surface can be represented by an analytic function $z(x,y)$, it is well represented on such a mesh, and we can expect rapid exponential convergence for the customized mesh update scheme.

Figure 11 shows the error in the steady state solution as measured by (3.7). As expected, the customized, strictly vertical mesh update algorithm yields exponential convergence, but we need a relatively high polynomial degree to reach machine precision; this is due to the fact that we use a pure spectral method for the entire surface. The multi-purpose mesh update techniques also yield exponential convergence, but the convergence rate is slower than the customized scheme. The Lagrangian update method is somewhat better than the normal update scheme. The reduction in convergence rate for the "general-purpose"

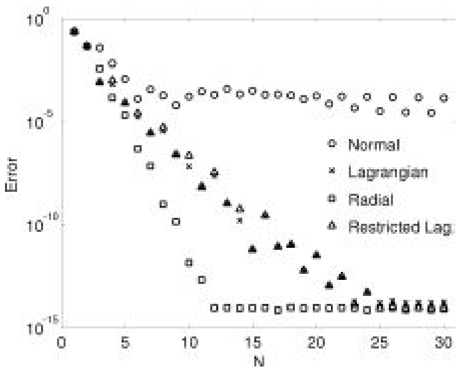


Figure 9: Error in the steady state solution for the catenoid with an initial surface as displayed in Figure 8a. The Lagrangian and radial update algorithms yield the same performance as before, but the normal update algorithm now seems to stabilize on an error of magnitude 10^{-4}.

methods is mainly due to a non-optimal mesh at steady state, which again is related to the particular initial surface. This example indicates the importance, as well as the sensitivity, of a good mapping between the reference domain $\hat{\Omega}$ and the deformed surface Ω.

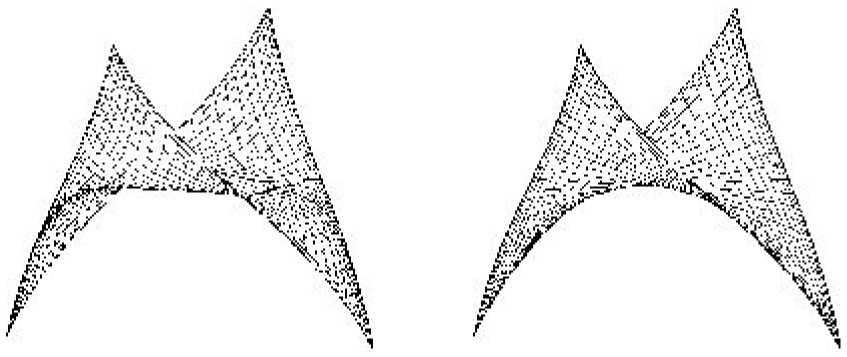
4.3 Enneper's surface

This surface can be parametrized by [9]

$$\begin{aligned}\varphi_1(u,v) &= u(1-\frac{1}{3}u^2+v^2),\\ \varphi_2(u,v) &= v(1-\frac{1}{3}v^2+u^2),\\ \varphi_3(u,v) &= u^2-v^2,\end{aligned} \tag{4.2}$$

where u and v are coordinates on a circular domain of radius R. For $R \le 1$ the surface is stable and a global area minimizer, whereas for $1 < R < \sqrt{3}$ the given parametrization gives an unstable minimal surface. In the latter case there also exist two (symmetrically similar) stable minimal surfaces which are also global area minimizers. For $R \ge \sqrt{3}$ the boundary curve intersects itself.

We first consider the case $R = 0.8$, for which the solution is unique. The surface is represented numerically using 12 spectral elements; see Figure 12.



(a) Initial surface **(b)** Minimal surface

Figure 10: Scherk's fifth surface, represented using a pure spectral method and a polynomial degree $N = 25$. The minimal surface is obtained using strictly vertical mesh updates, preventing severe mesh distortions.

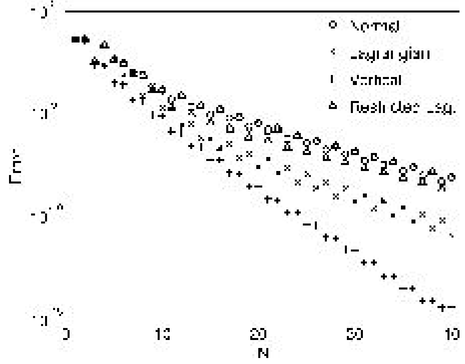


Figure 11: Error in the steady state solution for Scherk's fifth surface measured by (3.7) and plotted as as function of the polynomial degree N.

The initial surface is created by a cylindrical extension of the boundary curve.

The normal, the Lagrangian, and the restricted Lagrangian mesh update algorithms are again applied. The error, as measured by (3.7), is shown in Figure 13. All mesh update algorithms yield exponential convergence, but the normal algorithm displays a slower and more uneven convergence rate.

For the Enneper surface, there is no particular symmetry that enables us

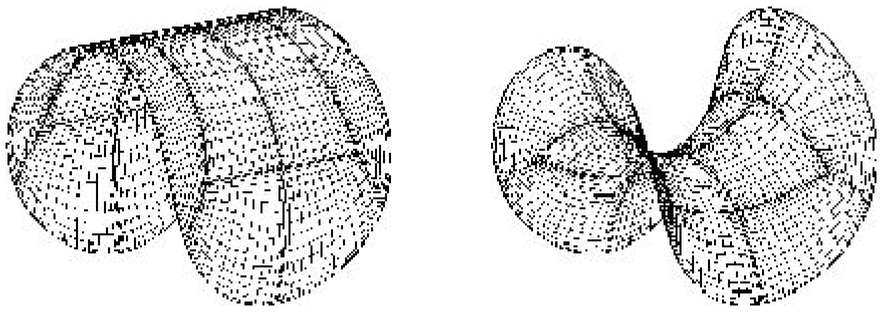


Figure 12: Enneper's surface, represented using 12 spectral elements and a polynomial degree $N = 15$. The minimal surface is obtained using the restricted Lagrangian mesh updates.

to *a priori* determine a customized mesh update algorithm that will lead to a good mesh at steady state and hence an even more rapid convergence rate than observed with the "general-purpose" algorithms.

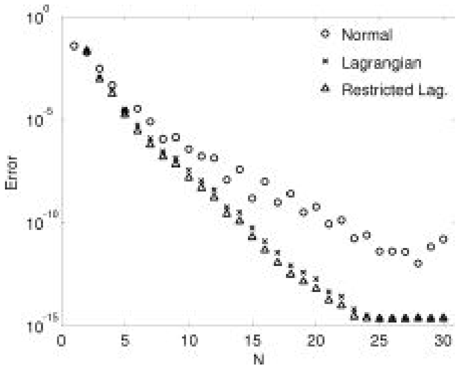
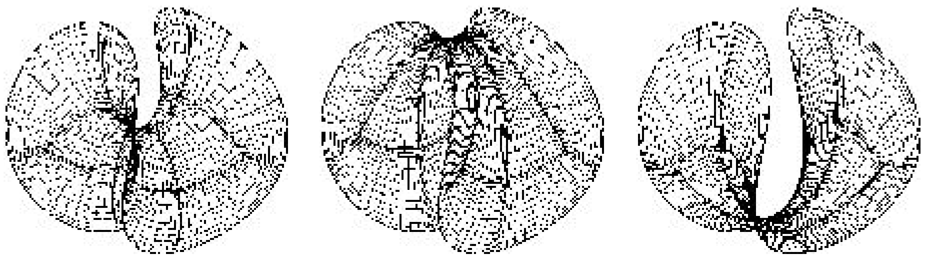


Figure 13: Error in the steady state solution for the Enneper surface with $R = 0.8$ measured by the norm (3.7) and plotted as a function of the polynomial degree N.

For $1 < R < \sqrt{3}$, we can obtain the two stable solutions and at the same time verify that the known solution (4.2) is unstable. We first use an interpolation of (4.2) to construct an initial surface. We then add random perturbations of order 10^{-10} to all the internal points of this surface and start the iterative algorithm.

At steady state we end up in one of the two stable solutions. Figure 14 shows all the three surfaces.



(a) Unstable minimal surface **(b)** Minimal surface 1 **(c)** Minimal surface 2

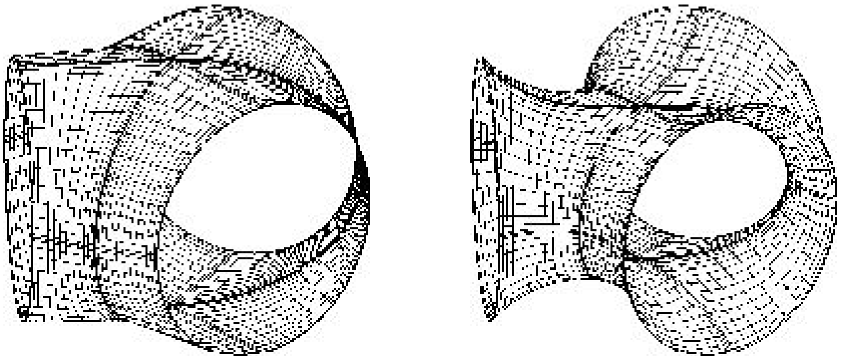
Figure 14: The three minimal surfaces in the Enneper case for $R = 1.2$. The unstable solution (a) is known analytically (see (4.2)), and is depicted by interpolating this known parametrization. The two other solutions are stable and global-area minimizers. The surfaces (b) and (c) are here obtained using the restricted Lagrangian mesh update algorithm, starting from slightly perturbed versions of (a) (by adding random perturbations on the order of 10^{-10}).

4.4 The Trinoid

We conclude with two examples of geometrically more challenging minimal surfaces which we are able to obtain. The first is the Jorge-Meeks trinoid [9], which has three circles as boundary curves and looks like three catenoids attached at the end. Similarly to the catenoid, it collapses when the radius of the boundary circles is too small compared to the distance between them. The initial surface is *not* three cylinders, but rather three conical cylinders that interpolate linearly between the boundary curves. Figure 15 shows the minimal surface obtained using the restricted Lagrangian update algorithm.

4.5 The Costa surface

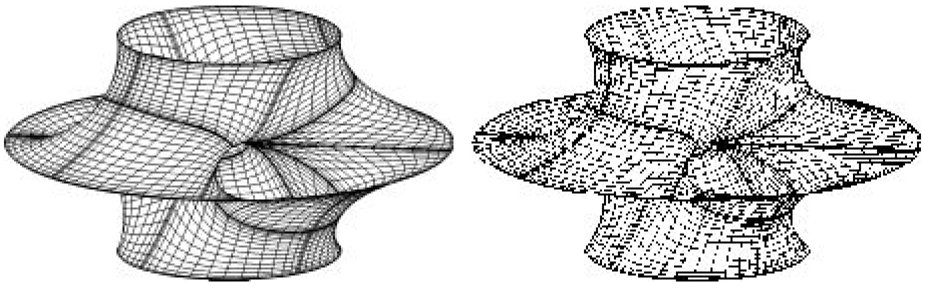
The last surface is a variation (or a simplification) of the Costa surface [9], which is one of the more exotic minimal surfaces and discovered as late as 1982. Here, the radius of the two small boundary circles is $R = 1$, while the radius of the large circle in the middle is $R = 2$. The vertical distance between the two small



(a) Initial surface **(b)** Minimal surface

Figure 15: The trinoid represented using 12 spectral elements.

boundary circles is $H = 2$. We remark that this is not the exact Costa surface, since the large circle is a plane curve in our case, while the exact Costa surface stretches to infinity in the horizontal plane. Figure 16 shows the minimal surface obtained using the restricted Lagrangian update algorithm.



(a) Initial surface **(b)** Minimal surface

Figure 16: The modified Costa surface represented using 16 spectral elements.

5 Conclusions

An algorithm for finding a high order polynomial approximation of a minimal surface with a given boundary has been introduced. It is based on a weak formulation of the Laplace-Beltrami problem. The problem is nonlinear, so a linearization and iterative scheme is applied to solve the discrete problem. The algorithm does not handle topological changes.

Minimal surfaces are smooth provided that the boundary curve is smooth. Spectral (or spectral element) methods based on high order polynomials should therefore be very suitable numerical methods for such problems. Indeed, our numerical results show that we are able to achieve exponential convergence as the polynomial degree increases. The convergence rate, however, depends strongly on the structure of the mesh points on the surface. Our iterative algorithm for computing minimal surfaces allows freedom in how the mesh points are moved during the iterations. A good mesh update algorithm is needed to retain a smooth mapping between the two-dimensional reference domain and the three-dimensional surface; this is essential in order to obtain rapid convergence.

We have compared three different mesh update algorithms for computing minimal surfaces. Our focus on minimal surfaces is partially motivated by the fact that there exist nontrivial minimal surfaces with analytical solutions which we can use for evaluation purposes. Numerical results show that purely Lagrangian updates are not always optimal, and neither are updates purely in the direction of the surface normal. For the Lagrangian update algorithm we always observe stability problems for sufficiently many iterations. A restricted Lagrangian update algorithm (where the displacement in the normal direction dominates the tangential displacement) seems to combine the good properties from a purely Lagrangian update and a purely normal update approach. The numerical results also show that a customized mesh update algorithm for the particular problem at hand often yields the best result.

The work presented here points to several important aspects related to using high order discretization methods for problems where the geometry is an unknown. The present work focuses on computing minimal surfaces, but the work is also highly relevant for fluid flow problems with free surfaces, in particular, when formulated in an Arbitrary Lagrangian Eulerian setting. For example, the normal update scheme is commonly used for such problems. Our numerical results demonstrate that high order (exponential) convergence rates can be realized, but great care has to be shown when selecting a mesh update algorithm. The results demonstrate the sensitivity in the error to the choice of interpolation points along the curved three-dimensional surface. The results also suggest that the restricted Lagrangian update scheme is a better choice than the normal update scheme.

Finding a general interface tracking algorithm that works better than the existing ones is a very challenging task. One such algorithm has been suggested in [1], but much work remains to be done in this direction.

Better algorithms are needed in order for high order methods to reach their full potential for computing minimal surfaces or for tracking time-dependent interfaces. We feel that the sensitivity to the choice of interpolation points has not been properly addressed in the literature before, and we stress that these challenges are particular for high order methods. More work is required in order to make high order methods robust and optimal for this class of problems.

Acknowledgment

The work has been supported by the Research Council of Norway under contract 185336/V30. The authors would like to thank Alf Emil Løvgren for helpful discussions on the derivation of the weak form of the Laplace-Beltrami operator.

Bibliography

[1] T. Bjøntegaard and E. M. Rønquist. Accurate interface-tracking for arbitrary Lagrangian-Eulerian schemes. *J. Comput. Phys.*, 228(12):4379–4399, 2009.

[2] T. Bjøntegaard, E. M. Rønquist, and Ø. Tråsdahl. High order polynomial interpolation of parameterized curves. In J. S. Hesthaven and E. M. Rønquist, editors, *Spectral and High Order Methods for Partial Differential Equations*, volume 76 of *Lecture Notes in Computational Science and Engineering*, pages 365–372. Springer, 2011.

[3] K. A. Brakke. The surface evolver. *Experimental Mathematics*, 1(2):141–165, 1992.

[4] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods, Fundamentals in Single Domains.* Springer, 2006.

[5] D. L. Chopp. Computing minimal surfaces via level set curvature flow. *J. Comput. Phys.*, 106(1):77–91, 1993.

[6] P. Concus. Numerical solution of the minimal surface equation. *Math. Comput.*, 21(99):340–350, 1967.

[7] C. Coppin and D. Greenspan. A contribution to the particle modeling of soap films. *Appl. Math. Comput.*, 26(4):315–331, 1988.

[8] P.-G. De Gennes, F. Brochard-Wyart, and D. Quéré. *Capillarity and Wetting Phenomena: Drops, Bubbles, Pearls, Waves.* Springer Verlag, 2004.

[9] U. Dierkes, S. Hildebrandt, A. Küster, and O. Wohlrab. *Minimal surfaces. I. Boundary value problems*, volume 295 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, 1992.

[10] J. Douglas. Solution of the problem of Plateau. *Trans. Amer. Math. Soc.*, 33(1):263–321, 1931.

[11] G. Dziuk. An algorithm for evolutionary surfaces. *Numerische Mathematik*, 58(1):603–611, 1991.

[12] G. Dziuk and J. E. Hutchinson. The discrete Plateau problem: algorithm and numerics. *Math. Comp.*, 68(225):1–23, 1999.

[13] G. Dziuk and J. E. Hutchinson. The discrete Plateau problem: convergence results. *Math. Comp.*, 68(226):519–546, 1999.

[14] A. R. Elcrat and K. E. Lancaster. On the behavior of a non-parametric minimal surface in a non-convex quadrilateral. *Arch. Rational Mech. Anal.*, 94(3):209–226, 1986.

[15] A. T. Fomenko. *The Plateau Problem.* Gordon and Breach Science Publishers, 1990.

[16] D. Greenspan. On approximating extremals of functionals–II theory and generalizations related to boundary value problems for nonlinear differential equations. *Int. J. Eng. Sci.*, 5(7):571–588, 1967.

[17] M. Hinata, M. Shimasaki, and T. Kiyono. Numerical solution of Plateau's problem by a finite element method. *Math. Comput.*, 28(125):45–60, 1974.

[18] L. W. Ho and A. T. Patera. Variational formulation of three-dimensional viscous free-surface flows: Natural imposition of surface tension boundary conditions. *Int. J. Numer. Methods Fluids*, 13(6):691–698, 1991.

[19] R. H. W. Hoppe. Multigrid algorithms for variational inequalities. *SIAM J. Numer. Anal.*, 24(5):1046–1065, 1987.

[20] A. Huerta and A. Rodríguez-Ferran (eds.). The Arbitrary Lagrangian-Eulerian Formulation. *Comput. Methods Appl. Mech. Engrg.*, 193(39-41):4073–4456, 2004.

[21] C. Isenberg. *The Science of Soap Films and Soap Bubbles.* Dover Publications Inc., 1992.

[22] E. Kreyszig. *Differential Geometry.* Dover Publications Inc., 1991.

[23] L. D. Landau and E. M. Lifshitz. *Fluid Mechanics.* Course of Theoretical Physics, Volume 6. Butterworth-Heinemann, 1987.

[24] R. Osserman. *A Survey of Minimal Surfaces.* Dover Publications Inc., 2002.

[25] J. A. F. Plateau. *Statique expérimentale et théorique des liquides soumis aux seules forces moléculaires.* Gauthier-Villars, 1873.

[26] T. Radó. On Plateau's problem. *Ann. of Math.*, 31(3):457–469, 1930.

[27] M. Struwe. *Plateau's Problem and the Calculus of Variations.* Princeton University Press, 1988.

[28] H. J. Wagner. A contribution to the numerical approximation of minimal surfaces. *Computing*, 19(1):35–58, 1977.

[29] T. J. Willmore. *Riemannian geometry.* Oxford University Press, 1993.

PAPER 5

ADAPTIVE SPECTRAL METHODS

ØYSTEIN TRÅSDAHL AND EINAR M. RØNQUIST

NTNU Preprint Numerics No. 1/2012

ADAPTIVE SPECTRAL METHODS

ØYSTEIN TRÅSDAHL AND EINAR M. RØNQUIST

Department of Mathematical Sciences,
Norwegian University of Science and Technology,
Trondheim, Norway

Abstract

This paper discusses numerical solution of boundary value problems using spectral methods combined with nonlinear and adaptive mappings between the reference domain and the physical domain. A brief review of existing methods for adaptive mesh generation is given, and a method for finding close to optimal mappings for boundary value problems in $\mathbb{R}^1$ is presented. The method exploits the link between high order numerical solutions of PDEs and approximation of parametric curves. Also, other adaptive methods for boundary value problems in $\mathbb{R}^1$ are proposed, based either on minimizing the discrete L^2-norm of the residual, or interpolating the residual as a parametric curve. The adaptive methods are constructed with the aim of finding optimal mappings, however, this turns out to be a very difficult task. Still, significant improvement from standard (non-adaptive) high order methods is achieved in some cases.

Keywords: Spectral methods, optimal mesh, advection-diffusion equation

1 Introduction

Spectral and pseudo-spectral (PS) methods are attractive methods for the numerical solution of partial differential equations (PDEs) for which the solution, the geometry and the source terms have a high degree of regularity. In particular, for analytic solution and data, the error in the numerical solution decreases exponentially fast as the dimension of the approximation space increases [6, 10]. This convergence rate is related to the global approach of the spectral and PS methods: the basis functions (trigonometric or algebraic polynomials) have global support, and the convergence rate is inherited from the convergence rate of classical trigonometric or polynomial approximation. Compared with finite difference and finite element methods, which are based on local approximations of fixed (low) order, this can lead to significantly smaller errors for a given number of degrees-of-freedom.

However, there are problems where spectral and PS methods are considered less suitable. If the solution has a boundary or interior layer with steep gradients, then small errors will only be achieved when the polynomial degree is high enough to resolve the localized phenomena. One such class of problems is singularly perturbed boundary value problems. For example, considering the advection-diffusion equation, a high Peclet number may yield a solution with a thin boundary layer, and the numerical solution may be corrupted by oscillations, spreading globally over the computational domain (unless the resolution is high enough). The thinner the boundary layer is, the larger the oscillations are for a fixed resolution (or polynomial degree). These problems are common for most numerical methods, including finite difference and finite element methods. However, these are cheaper and easier to refine locally to the required resolution [2, 26]. For this reason, adaptivity is much more developed in the context of low order methods than for spectral methods.

There have been a number of strategies proposed for overcoming these difficulties in the context of spectral and PS methods. One option is post-processing the solution through filtering, which can be used to dampen oscillations. This requires a modal representation of the numerical solution. Another option is the addition of artificial viscosity. Tadmor introduced the vanishing viscosity method for shock capturing [22, 23]. Brezzi et. al. [8] introduced bubble stabilization in a finite element context, in which the space of test functions is augmented by a set of "bubble functions". The ideas were applied to spectral methods by Canuto [9] and Pasquarelli and Quarteroni [20].

Adaptive methods can also be based on modifying the mesh on which the spectral solution is represented. A spectral Galerkin approach typically involves applying a coordinate transformation, solving the PDE in a *reference domain* using a standard mesh, and then mapping the numerical (polynomial) solution back to the *physical domain* to approximate the exact solution. A key idea with adaptive methods is that, to achieve sufficient resolution in the interior or boundary layers, mesh nodes are clustered in these critical regions in the physical domain. For low order methods this always leads to a better approximation, but for high order methods based on polynomials, the issue is a bit more complicated. A high mesh node density may not necessarily mean better accuracy, since the numerical solution may display wild oscillations *between* the nodes. The method may not even converge as the number of nodes increases. The key to constructing good adaptive meshes in the context of high order methods is regularity: the reference domain must be mapped to the physical domain by a *smooth coordinate transformation* in order to avoid oscillations. Enforcing a smooth and yet suitably adaptive mapping is difficult, and few authors address the regularity of the mapping explicitly. The adaptive methods are often based on some other requirement, for example equidistribution of some function over the computational domain, or minimization of some norm of the numerical solution. An overview over existing adaptive methods in this category is given in Section 3.

The objectives of this paper are to investigate optimal or close to optimal mappings for the numerical solution of boundary value problems and to test adaptive methods that are constructed with the aim of finding such mappings. The paper is structured as follows: in Section 2 the potential of adaptive methods is demonstrated through the numerical solution of a one-dimensional advection-diffusion model problem, and the role of the coordinate transformation is briefly discussed. We then proceed in Section 3 with a review of existing adaptive methods in the high order context. In Sections 4 and 5 we present methods for constructing close to optimal and approximate coordinate transformations, respectively; these methods are based on interpolation of parametric curves, and the link between interpolation and adaptive high order methods is explained. In Sections 6 and 7 we introduce alternative adaptive methods, and in Section 8 some additional numerical results are presented. Finally, in Section 9, we present some conclusions and remarks.

2 A motivational example

To show the potential of adaptivity in the context of spectral methods, we begin with an example. The problems we will consider in this paper are advection-diffusion boundary value problems that can be written on the form

$$-\varepsilon\frac{\mathrm{d}^2u}{\mathrm{d}x^2}+\frac{\mathrm{d}u}{\mathrm{d}x}=f, \qquad x\in\Omega, \tag{2.1}$$

accompanied by suitable boundary conditions. Here, ε is a constant, f is a smooth function, and Ω is a bounded interval on the real axis.

Consider a particular model problem on the form (2.1) where $\Omega=(0,1)$, $\varepsilon=0.01$, $f(x)=1$, and with homogeneous Dirichlet boundary conditions. The problem has the exact solution

$$u(x)=x-\frac{e^{x/\varepsilon}-1}{e^{1/\varepsilon}-1}, \tag{2.2}$$

which features a boundary layer with a width of order $\mathcal{O}(\varepsilon)$ near $x = 1$.

A pure spectral method is applied to solve the problem numerically. It is based on the equivalent weak form of (2.1), and can be stated as follows: find $u \in X = H_0^1(\Omega)$ such that

$$a(u, v) = (f, v), \qquad \forall v \in X, \tag{2.3}$$

where

$$a(u, v) = \varepsilon \int_\Omega \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\ \mathrm{d}x + \int_\Omega \frac{\mathrm{d}u}{\mathrm{d}x} v\ \mathrm{d}x, \tag{2.4}$$

and

$$(f, v) = \int_\Omega f v\ \mathrm{d}x. \tag{2.5}$$

Prior to discretization both integrals are transformed to integrals over a reference domain $\widehat{\Omega} = (-1, 1)$. The coordinate transformation is given by $x = \mathcal{F}(\xi)$, with $x \in \Omega$, $\xi \in \widehat{\Omega}$. Discretization is based on high order polynomials over $\widehat{\Omega}$, and the discrete space $X_N \subset X$ can be expressed as

$$X_N = \{v \in X, v \circ \mathcal{F} \in \mathbb{P}_N(\widehat{\Omega})\}. \tag{2.6}$$

Exact integration of the bilinear and linear forms is replaced by quadrature at the Gauss-Lobatto Legendre (GLL) points. The numerical solution $\hat{u}_N = u_N \circ \mathcal{F}$ is a polynomial of degree N over $\widehat{\Omega}$, and it can be represented by the nodal basis

$$\hat{u}_N(\xi) = \sum_{j=0}^{N} u_j \ell_j(\xi). \tag{2.7}$$

Here, ℓ_j is a Lagrangian interpolant through the GLL points $\xi_0, \ldots, \xi_N$, i.e., it is the unique polynomial of degree N satisfying $\ell_j(\xi_k) = \delta_{jk}$. The functions $\ell_0, \ldots, \ell_N$ make up a basis for the space $\mathbb{P}_N(\widehat{\Omega})$ of polynomials of degree less than or equal to N over $\widehat{\Omega}$. Note that in (2.7) the basis coefficients $u_0 = u_N = 0$ due to the homogeneous boundary conditions, and $u_j = \hat{u}_N(\xi_j) = u_N(x_j)$, with $x_j = \mathcal{F}(\xi_j)$.

The difference between the exact solution and the numerical solution measured in the L^2 norm is given by

$$||u - u_N||^2_{L^2(\Omega)} = \int_\Omega \Big(u(x) - u_N(x)\Big)^2\ \mathrm{d}x. \tag{2.8}$$

Again, the integral can be transformed to an integral over $\widehat{\Omega}$ and exact integration can be replaced by GLL quadrature. In practice, overintegration in $M \gg N$ quadrature points is used in order to ensure that the quadrature error is subdominant the discretization error.

Solving the boundary value problem over a reference domain requires a bijective mapping between $\widehat{\Omega}$ and Ω. Of course, the easiest, and certainly the most common, option is to use a linear (or affine) mapping, $\mathcal{F} = \overline{\mathcal{F}}$, given explicitly as

$$x = \overline{\mathcal{F}}(\xi) = \frac{\xi + 1}{2}. \tag{2.9}$$

However, we may also consider more general (nonaffine) mappings $\mathcal{F} : \widehat{\Omega} \to \Omega$. There is nothing in the spectral method that requires $\mathcal{F}$ to be linear; the only requirement is that it is bijective. Note that the numerical solution is not constructed as a polynomial approximation of u, but rather of the mapped solution

$$u(\mathcal{F}(\xi)) = (u \circ \mathcal{F})(\xi) = \hat{u}(\xi). \tag{2.10}$$

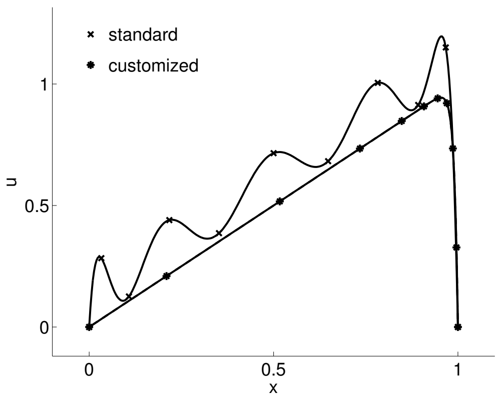
The standard error estimates apply to $\hat{u}$, so using $\mathcal{F}$ to increase the smoothness of $\hat{u}$ may increase the convergence rate.

A consequence of using a nonlinear mapping is that the test functions are no longer polynomials when viewed as functions on the physical domain. This is discussed in [21], where optimal error estimates are derived.

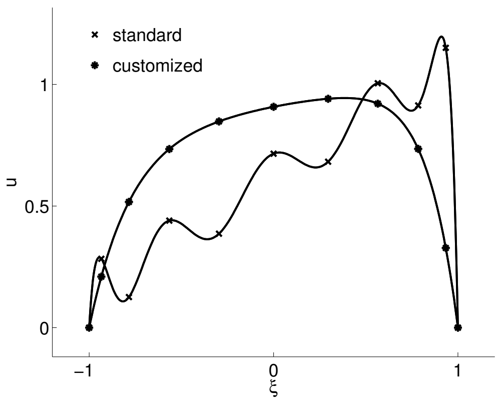
Let us now solve the model problem using a polynomial degree $N = 10$ and two different mappings $\mathcal{F}$: the affine mapping (2.9), and a customized, nonlinear mapping determined by the ET method described in Section 4. Figure 1 shows the two numerical solutions over both the physical domain and the reference domain. The affine mapping results in an oscillatory numerical solution due to the failure of resolving the boundary layer. On the other hand, the nonlinear $\mathcal{F}$ gives a numerical solution that cannot be distinguished from the exact solution. Figure 2 shows that this mapping is highly nonlinear, and that it moves all the GLL points in the direction of the boundary layer.

3 A review of adaptive high order methods

Adaptive mesh generation in the context of spectral and pseudo-spectral (PS) methods has been investigated for several decades. Although some progress has been made, a lot remains to be done in order for this to represent a practical and efficient tool. This is in contrast to adaptive low order methods (e.g., hp-FEM), which is a mature field and also widely used in commercial applications.



(a) Exact and numerical solutions in Ω



(b) Numerical solutions in $\widehat{\Omega}$

Figure 1: Two numerical solutions of an advection-diffusion problem with exact solution (2.2), solved using a pure Legendre spectral method. Using an affine mapping $\mathcal{F}$ (`standard`) from the reference domain $\widehat{\Omega}$ to the physical domain Ω gives poor resolution of the boundary layer near $x = 1$ and therefore unwanted oscillations in the numerical solution u_N. Using a customized, nonlinear $\mathcal{F}$ (`customized`) smooths out the variation in $\hat{u}$ over $\widehat{\Omega}$ and gives sufficient resolution of the localized effects, resulting in a vastly better numerical solution.

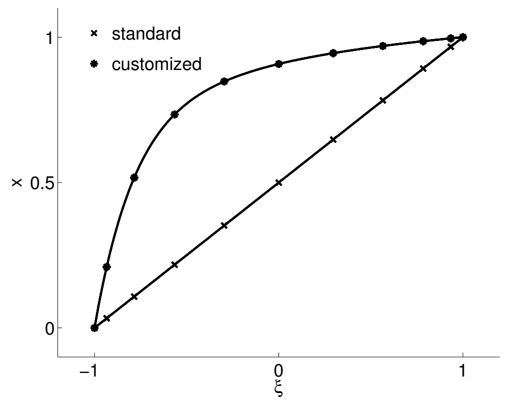


Figure 2: The two mappings $x = \mathcal{F}(\xi)$ used to produce the two numerical solutions shown in Figure 1. The affine (`standard`) mapping corresponds to the oscillatory solution, and the nonlinear (`customized`) mapping corresponds to the solution that cannot be distinguished from the exact solution.

The reason for the relative lack of breakthrough is probably the complexity and the cost associated with constructing a global mapping which will minimize the discretization error for a fixed order of approximation.

In the literature, adaptivity for high order methods is most often applied to problems with interior or boundary layers. This is the area where such methods have the biggest potential and where the effect of adaptivity is the most striking. Standard high order methods may yield exponential convergence for such problems, but small error levels are typically achieved only after the interior or boundary layer has been properly resolved, and then often at a very low (exponential) rate [27]. An adaptive mapping will typically cluster (physical) grid points in the region(s) in Ω where the exact solution u changes rapidly, effectively "stretching out" the corresponding region in $\widehat{\Omega}$, yielding slower variation and less step gradients in the mapped solution $\hat{u}$.

Most adaptive methods can be applied to time-independent problems, but they are more often applied to time-dependent problems. This is because such problems lend themselves well to most of the existing adaptive algorithms. The numerical solution from the current time step contains valuable information about the variation that will occur in the solution at the next time step. In time-independent problems, adaptivity either involves an *a priori* asymptotic analysis to disclose the position (and width) of interior and boundary layers, or an iterative method is employed to gradually discover these features.

The analysis of adaptive spectral and PS methods can essentially be done in two ways: on the reference domain or on the physical domain. With the former approach, the basis functions are not mapped and standard tools can be applied. The difficulty lies in the PDE, which must be mapped from the physical domain by the inverse of $\mathcal{F}(\xi)$ and can be very complex. The latter approach allows us to consider the original PDE, but now the basis functions are mapped, and specialized tools must be developed. Wang and Shen used this approach when they derived optimal error estimates for mapped Legendre spectral methods in [21] and for mapped Jacobi spectral methods in [27].

The simplest adaptive mesh generation methods are not really adaptive, in that they simply *choose* the structure of $\mathcal{F}$ without using any information about the exact solution. In [17] the mapping

$$\mathcal{F}(\xi;\lambda) = -1 + \sigma_\lambda \int_{-1}^{y} (1-t^2)^\lambda \,\mathrm{d}t \tag{3.1}$$

is used, where σ_λ is a constant depending on λ, and λ is a parameter to be determined. For $-1 < \lambda < 0$ the mesh nodes are moved towards the origin, and for $\lambda > 0$ they are clustered near the end points. The authors choose $\lambda = 1$ in their numerical experiments and solve various boundary value problems with a modified Legendre spectral method. They also extend their method to two-dimensional problems.

Similarly, in [24] Tang and Trummer use a transformation $\mathcal{F}_m(\xi)$ consisting of iterated sine functions, defined recursively by

$$\mathcal{F}_m(\xi) = \sin(\frac{\pi}{2}\mathcal{F}_{m-1}(\xi)), \qquad \mathcal{F}_0(\xi) = \xi. \tag{3.2}$$

As m increases, the mesh nodes are clustered more densely at the end points of the physical interval. The authors test their mappings on one-dimensional boundary value problems for different $m \leq 3$.

A common approach in adaptive high order methods is to restrict $\mathcal{F}$ to be a function on a particular, *a priori* chosen form, with one or more free parameters. Adaptivity is then simplified to the task of determining suitable values for these parameters. This is the approach adopted by Kosloff and Tal-Ezer [16], who use a mapping

$$\mathcal{F}(\xi;\lambda) = \frac{\arcsin(\lambda\xi)}{\arcsin(\lambda)}, \qquad 0 < \lambda < 1, \tag{3.3}$$

which stretches a Gauss-Lobatto Chebyshev (GLC) grid toward a uniform grid as $\lambda \to 1^-$. They discuss various considerations for choosing the parameter λ:

enabling resolution of the largest possible wave number, or optimizing for interpolation of a general trigonometric function. An adaptive approach is to choose the parameters such that the function to be approximated can be represented by a Chebyshev expansion with the minimal number of terms. The authors show that a suitable choice leads to a differentiation matrix D whose eigenvalues are all $\mathcal{O}(N)$, where N is the polynomial degree, reducing the time step restriction in explicit time integration methods.

In the adaptive methods proposed in [1, 4, 5] the parameters in $\mathcal{F}$ are chosen so that the numerical solution minimizes some chosen functional. The mappings used are two-parameter functions composed of trigonometric and inverse trigonometric functions, e.g.,

$$\mathcal{F}(\xi;\lambda_1,\lambda_2) = 1 + \frac{4}{\pi}\arctan(\lambda_1\tan(\frac{4}{\pi}(\frac{\lambda_2 - x}{\lambda_2 x - 1} - 1))), \quad \lambda_1 > 0, \quad -1 < \lambda_2 < 1, \tag{3.4}$$

where λ_1 relates to the width and λ_2 to the position of an interior or boundary layer in the exact solution. The functional to be minimized is typically a measure of the total variation in the solution, for example the Sobolev norm

$$\mathcal{J} = \int_\Omega (u^2 + u_x^2 + u_{xx}^2)\,\mathrm{d}x. \tag{3.5}$$

The idea is that minimizing $\mathcal{J}$ means reducing unwanted oscillations, since these can give large contributions to $\mathcal{J}$ through the second derivative of u. A good mapping yields a solution with few (or no) unwanted oscillations, and hence a low value of $\mathcal{J}$. The interpolation and integration in the minimization procedure introduces some significant overhead, but on the other hand, vast improvements in convergence rate may be achieved.

Tee and Trefethen [25] choose a particular two-parameter mapping $\mathcal{F}$ based on how it maps singularities of the exact solution in the complex plane. The parameters are chosen to enlarge the *ellipse of analyticity* of $\hat{u}$, which is the largest ellipse with foci ± 1 in which $\hat{u}$ has no singularities. This immediately increases the convergence rate of the rational spectral collocation method, since the error is of order $\mathcal{O}((L+l)^{-N})$, where L and l are the semi-axes of the ellipse [3]. The method in [25] is limited to cases where u has one pair of complex conjugated singularities, but it is extended to problems with multiple pairs of singularities in [14].

Another family of adaptive mesh methods are based on equidistribution of some monitor or weight function over the physical domain. Such methods must consider general $\mathcal{F}$, since it is not given that equidistribution can be achieved by an $\mathcal{F}$ on a particular functional form. One weight function that has been considered is the arc length of the numerical solution. Creating a mesh such that the arc length is distributed equally between the mesh points will for example ensure a clustering of mesh nodes in regions with steep gradients. Equidistribution has been shown to work well with different weight functions in FEM and FD methods [11, 12, 15], but it has not been explored much in the context of high order methods. It is usually not a suitable adaptive method in itself, since it does not ensure regularity in $\mathcal{F}$, as demonstrated in [19]. However, the same paper shows that equidistribution in conjunction with filtering can give very good results. The authors also reduce the cost of the method by solving the equidistribution problem with a low order method.

Funaro [13] proposed an adaptive collocation method where the collocation points are determined by inserting the Legendre polynomial of degree N into the given PDE. For an advection-diffusion model problem this results in a staggered mesh where the nodes are actually slightly shifted *away* from the boundary layer. Still, the numerical solution on the staggered mesh displays smaller oscillations than that obtained with the standard collocation method, and it is a better approximation of the exact solution. A favorable feature of this method is that it is applicable also in higher space dimensions, and the author shows an example by solving a boundary value problem in $\mathbb{R}^2$.

4 Constructing an optimal mapping

The adaptive high order methods mentioned so far are all constructed with the objective to adjust the mapping $\mathcal{F}$ from $\widehat{\Omega}$ to Ω in such a way that the discretization error decreases for a fixed number of degrees-of-freedom (i.e., for a fixed polynomial degree, N). A natural question to ask is then: to what extent can the mapping be used to reduce the discretization error? Let us by *optimal mapping* define the mapping $\mathcal{F}$ that, when used in the numerical solution of a given boundary value problem with a spectral method as described in Section 2, results in the smallest numerical error, measured in the L^2-norm. This particular numerical solution will be defined as the *optimal solution* of the given boundary value problem for the given method and the given polynomial degree N. Note that the optimal mapping depends on all of these factors; even changing the norm in which we measure the error may give a (slightly) different optimal $\mathcal{F}$.

The optimal $\mathcal{F}$ is not unique since information about the mapping is only required at a finite number of points. For example, using a spectral method to solve (2.1) with Dirichlet boundary conditions only requires the value of the Jacobian (the first derivative of $\mathcal{F}$) at the quadrature points. Any two mappings that have the same values of the derivatives at these points are not discerned by the numerical method. In the following, we will therefore assume that $\mathcal{F} \in \mathbb{P}_N(\widehat{\Omega})$ and we will denote the mapping as $\mathcal{F}_N$ to emphasize this; this corresponds to an isoparametric approach.

Finding optimal mappings adaptively is very difficult. We therefore start by considering the simpler problem of finding optimal mappings based on the exact solution. Consider an advection-diffusion problem (2.1) with a smooth right hand side f and homogeneous Dirichlet boundary conditions. The exact solution is a function $u(x)$ defined on Ω. As any other function it can be viewed as a *parametric curve*, using the trivial parametrization

$$\boldsymbol{u}(x) = (x, u(x)), \qquad x \in \Omega. \tag{4.1}$$

The curve can also be *reparametrized* by a change of variable $x = \mathcal{F}_N(\xi)$, yielding the new representation

$$\hat{\boldsymbol{u}}(\xi) = (\mathcal{F}_N(\xi), u(\mathcal{F}_N(\xi))) = (\mathcal{F}_N(\xi), \hat{u}(\xi)), \qquad \xi \in \widehat{\Omega}. \tag{4.2}$$

The numerical solution of the PDE, obtained with a spectral method using the (isoparametric) mapping $\mathcal{F}_N$, can be represented by

$$\hat{\boldsymbol{u}}_N(\xi) = (\mathcal{F}_N(\xi), \hat{u}_N(\xi)), \qquad \xi \in \widehat{\Omega}, \tag{4.3}$$

and this can be viewed as an approximation of the vector-valued function in (4.2).

A simple way to approximate the exact solution is through interpolation. In order to exploit the flexibility that the change of variable $\mathcal{F}_N$ offers, we consider interpolation of parametric curves. The parametric curve interpolation operator $\boldsymbol{I}_N$ is defined by applying the standard function interpolation operator I_N to each parametric function. The functions are interpolated in the (affinely mapped) GLL points. Applying the operator to (4.1) produces the interpolant

$$\boldsymbol{I}_N\boldsymbol{u}(x) = (I_N x, I_N u(x)) = (x, I_N u(x)), \qquad x \in \Omega, \tag{4.4}$$

and interpolating (4.2) yields

$$\boldsymbol{I}_N\hat{\boldsymbol{u}}(\xi) = (I_N\mathcal{F}_N(\xi), I_N\hat{u}(\xi)) = (\mathcal{F}_N(\xi), I_N\hat{u}(\xi)), \qquad \xi \in \widehat{\Omega}. \tag{4.5}$$

We will refer to these curves as *parametric interpolants.* Now, even though $\boldsymbol{u}$ and $\hat{\boldsymbol{u}}$ describe the same curve, the interpolants $\boldsymbol{I}_N\boldsymbol{u}$ and $\boldsymbol{I}_N\hat{\boldsymbol{u}}$ are generally not the same. If we let $\overline{\mathcal{F}}$ denote the affine mapping from $\widehat{\Omega}$ to Ω, and $\overline{x}_m = \overline{\mathcal{F}}(\xi_m)$, we remark that $(\overline{x}_m, u(\overline{x}_m))$ and $(\mathcal{F}_N(\xi_m), \hat{u}(\xi_m))$ are generally different points on the exact curve (unless $\mathcal{F}_N = \overline{\mathcal{F}}$).

The interpolant (4.4) is equivalent to classical interpolation of the function $u(x)$ in the GLL points. The interpolant (4.5) also corresponds to classical interpolation in the special case that $\mathcal{F}_N = \overline{\mathcal{F}}$, i.e., when $\mathcal{F}_N$ just varies linearly with ξ. However, if $\mathcal{F}_N$ is chosen wisely, then $\hat{u}$ is better suited for polynomial interpolation than u. For example, it may be a function of higher regularity, or it may have slower variation and wider boundary layers. Reparametrization can be used to "move" some of the complexity from one parametric function to the other. One can in principle consider a strongly nonlinear change of variable $x = \mathcal{F}_N(\xi)$ that yields little or no variation in $\hat{u}(\xi)$, however, this may only be possible for a very high N. A balance in complexity between the two parametric functions is most likely better in order to achieve a small interpolation error for a fixed N.

The optimal reparametrization is the one that results in minimization of the L^2-norm of the interpolation error[1]. Finding the optimal change of variable is a problem with $N+1$ free variables, since the reparametrization is uniquely determined by the value of $\mathcal{F}_N$ at the GLL points. The mapping $\mathcal{F}_N : \widehat{\Omega} \to \Omega$ such that $\mathcal{F}_N \in \mathbb{P}_N(\widehat{\Omega})$ can be represented explicitly as

$$\mathcal{F}_N(\xi) = \sum_{j=0}^{N} x_j^* \ell_j(\xi), \tag{4.6}$$

with $x_j^* = \mathcal{F}_N(\xi_j)$. Choosing basis coefficients x_j^* *uniquely* determines $\mathcal{F}_N$, which in turn uniquely determines the interpolant $I_N \hat{\boldsymbol{u}}$, with

$$I_N \hat{u}(\xi) = \sum_{j=0}^{N} u_j^* \ell_j(\xi), \tag{4.7}$$

and with $u_j^* = I_N \hat{u}(\xi_j) = u(x_j^*)$, $j = 0, \dots, N$.

The free variables $x_0^*, \dots, x_N^*$ represent $N+1$ *degrees-of-freedom* in the construction of the interpolant. Note how this differs from classical interpolation of functions: these degrees-of-freedom are available *after* we have chosen the GLL points as interpolation points in the reference domain. If exploited wisely,

[1]Since the exact curve can be represented by a single function, we measure the interpolation error between this function and the function representation of the interpolant. This is possible as long as $\mathcal{F}_N$ is invertible.

they represent a potential for parametric interpolants to be more accurate than classical interpolants.

The variables $x_0^*, \dots, x_N^*$ can be also expressed through the affine mapping $\overline{\mathcal{F}}(\xi)$. In particular, we can write

$$x_j^* = \mathcal{F}_N(\xi_j) = \overline{\mathcal{F}}(\xi_j^*), \qquad j = 0, \dots, N, \tag{4.8}$$

and thus redefine the problem of finding an optimal mapping to a problem of choosing the values $\xi_0^*, \dots, \xi_N^*$. The choice should be restricted to values that satisfy

$$\xi_0^* < \xi_1^* < \dots < \xi_N^*, \tag{4.9}$$

since $\mathcal{F}_N$ should be invertible. The monotonicity restriction (4.9) does not guarantee that $\mathcal{F}_N$ is invertible, since it can oscillate between the nodes, but this is a small practical problem and will not be discussed here.

The problem of optimal choice of $\xi_0^*, \dots, \xi_N^*$ was investigated in [7], where the *equal-tangent* (ET) method was proposed. The method reduces the number of degrees-of-freedom by two by requiring that the end points be interpolation points. This was motivated by the numerical solution of PDEs in deformed quadrilaterals in $\mathbb{R}^2$, but it also makes sense here: it ensures that the end points are nodes on the computational mesh in the physical domain, which makes imposition of the boundary conditions easy. The remaining $N-1$ free variables are determined by the condition that the interpolant also matches the tangent directions at the internal interpolation points. This can also be viewed as Hermite interpolation when considering the function representation of the curves [18]. It can be achieved by solving the system of non-linear equations

$$\frac{\mathrm{d}u}{\mathrm{d}x}(\mathcal{F}_N(\xi_j))\,\frac{\mathrm{d}\mathcal{F}_N}{\mathrm{d}\xi}(\xi_j) - \frac{\mathrm{d}I_N(u \circ \mathcal{F}_N)}{\mathrm{d}\xi}(\xi_j) = 0, \qquad j = 1, \dots, N-1, \tag{4.10}$$

with respect to the $N-1$ free variables $\xi_1^*, \dots, \xi_{N-1}^*$. The dependency of each term on the free variables is trough the coordinate transformation $\mathcal{F}_N$; see (4.6) and (4.8). We use $u \circ \mathcal{F}_N$ instead of $\hat{u}$ since we do not assume a priori knowledge of $\hat{u}$; in fact, it depends on $\mathcal{F}_N$, which in essence is the unknown here. The last term is given by

$$\frac{\mathrm{d}I_N(u \circ \mathcal{F}_N)}{\mathrm{d}\xi}(\xi) = \sum_{j=0}^{N} u(\mathcal{F}_N(\xi_j))\ell_j'(\xi). \tag{4.11}$$

The equations in (4.10) can be viewed in two ways: (i) in the reference domain they represent the difference in the derivative between of $\hat{u}$ and $I_N\hat{u}$ at the internal GLL points; (ii) in the physical domain, the equations represent the dot product between the tangent vector $(\mathcal{F}_N'(\xi_j), (I_N\hat{u})'(\xi_j))$ to the interpolant and the normal vector $(u'(\mathcal{F}_N(\xi_j)), -1)$ to the exact curve at the interpolation points. Hence, by solving the system (4.10) for $\xi_1^*, \ldots, \xi_{N-1}^*$, we are implicitly finding a coordinate transformation $\mathcal{F}_N$ such that the interpolant $I_N\hat{\boldsymbol{u}}$ matches both function values and derivatives of $\hat{\boldsymbol{u}}$ at the internal GLL points $\xi_1, \ldots, \xi_{N-1}$.

The system (4.10) is solved using Newton's method. Exact expressions for the partial derivatives of the objective function with respect to the free variables can be derived by standard techniques. In order to make the interpolation method more robust, Newton's method is run several times with different initial values, and the solution that results in the smallest interpolation error, measured in the discrete L^2-norm, is chosen. Implementation is discussed in more detail in [7].

When the ET method has produced a solution to the interpolation problem, the associated mapping $\mathcal{F}_N$ can be used in a pure spectral method for solving the given boundary value problem. Assume that the exact solution u to this problem belongs to $H^\sigma(\Omega)$ and that we construct the classical interpolant I_Nu associated with the GLL points. The standard interpolation error is then bounded by [6]

$$||u - I_Nu||_{L^2(\Omega)} \leq cN^{-\sigma}||u||_{H^\sigma(\Omega)}, \tag{4.12}$$

where c is a constant. If the exact solution u is analytic $(\sigma \to \infty)$, we can expect the classical interpolation error to decrease exponentially fast as the polynomial degree, N, increases. It is now of interest to consider the following two questions: (i) what difference does it make to use a non-affine mapping instead of the standard mapping? and (ii) what is the difference between the interpolation error $||u - I_Nu||$ and the discretization error $||u - u_N||$ in the spectral solution of the given boundary value problem?

We illustrate these issues by revisiting the numerical example from Section 2. Figure 3 shows that a standard spectral method (i.e., using a linear mapping $\overline{\mathcal{F}}$) results in exponential convergence, but that the convergence rate can be increased dramatically by using the ET method to construct a more appropriate mapping $\mathcal{F}_N$. Figure 3 also shows the interpolation error when interpolating the exact solution as a parametric curve, for two different parametrizations, one based on the non-affine mapping $\mathcal{F}_N$ and one based on the affine mapping $\overline{\mathcal{F}}$. As expected, the results indicate a close relationship between the discretization error and the interpolation error.

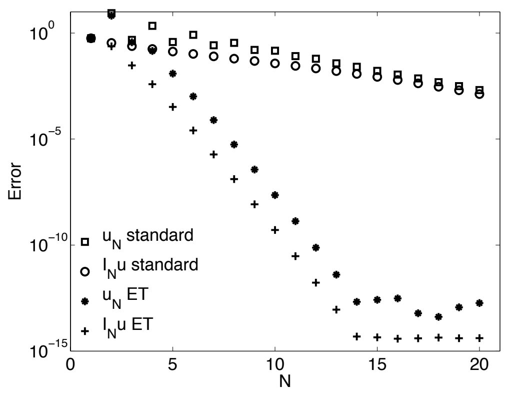


Figure 3: The error $||u - u_N||_{L^2}$ in the spectral Galerkin solution of (2.1) where (2.2) is the exact solution, and the error $||u - I_N u||_{L^2}$ in the parametric interpolation of the exact solution. The non-affine mapping $\mathcal{F}_N$ produced by the ET method gives much faster convergence than the standard (affine) mapping $\overline{\mathcal{F}}$, both when considering interpolation and numerical solution of the boundary value problem.

5 Constructing an approximate mapping

Let us now change the way we construct our non-affine mapping. In the previous section we applied the ET method to the exact solution of the original problem to construct a very good mapping $\mathcal{F}_N$. Let us now instead apply the ET method to the exact solution of a modified problem. In particular, we first change the diffusivity ε to $\tilde{\varepsilon}$ (with $\tilde{\varepsilon} \geq \varepsilon$) in the advection-diffusion problem (2.1). The exact solution of this modified problem has a thicker boundary layer compared to the original problem. We now apply the ET method to the exact solution of the modified problem and construct a corresponding non-affine mapping $\mathcal{F}_N(\xi; \tilde{\varepsilon})$ (the argument is added to remind us that the mapping constructed this way depends on the diffusivity $\tilde{\varepsilon}$ chosen in the approximate problem). Finally, we use the mapping $\mathcal{F}_N(\xi; \tilde{\varepsilon})$ in the spectral Galerkin solution of the original problem. In Figure 4 we show the discretization error as a function of the polynomial degree, N, for $\tilde{\varepsilon} = 0.1$, $\tilde{\varepsilon} = 0.02$ and $\tilde{\varepsilon} = \varepsilon = 0.01$ (i.e., corresponding to the results of the previous section). We notice that significant improvement from using the standard affine mapping is achieved. However, the results also show a high degree of sensitivity to the quality of the mapping used.

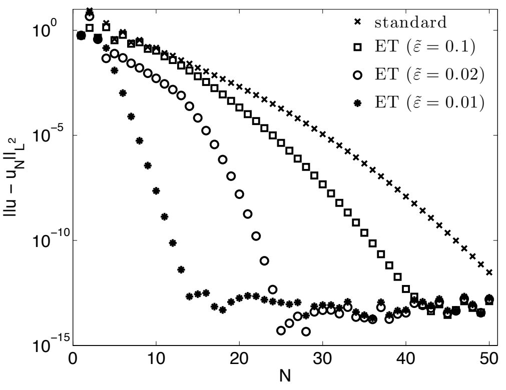


Figure 4: The error $||u - u_N||_{L^2}$ in the spectral Galerkin solution of (2.1) where (2.2) is the exact solution. The non-affine mapping $\mathcal{F}_N(\xi; \tilde{\varepsilon})$ used here is constructed by applying the ET method to the exact solution of a modified advection-diffusion problem with a diffusivity $\tilde{\varepsilon} \geq \varepsilon$.

6 Adaptivity through minimization of the residual

6.1 Residual of the strong form

If we want to construct an adaptive method to find the optimal solution of a given boundary value problem, we immediately face a problem: our definition of the optimal solution as the L^2-minimizer of the error in the numerical solution of the PDE does not lend itself to adaptivity, since the evaluation of the error involves the exact solution; an adaptive method cannot assume any knowledge of the exact solution.

The most valuable information we have about the approximation properties of a numerical solution of a boundary value problem is the residual. It may give information about about the magnitude of the numerical error, as well as the location of the areas where the numerical solution represents a poor (or good) approximation of the exact solution. An adaptive method can in principle be constructed by minimizing the residual.

Consider an advection-diffusion problem on the form (2.1) with Dirichlet boundary conditions, and assume that we apply a spectral method as shown in Section 2, using a polynomial mapping $\mathcal{F}_N$. The residual of the strong form is then given by

$$r_N(x) = f(x) - f_N(x), \tag{6.1}$$

where $f(x)$ is the right hand side of the PDE and

$$f_N(x) = -\varepsilon \frac{\mathrm{d}^2 u_N}{\mathrm{d}x^2} + \frac{\mathrm{d}u_N}{\mathrm{d}x} \tag{6.2}$$

is the numerical solution (2.2) inserted into the left hand side of the PDE.

Figure 5 shows some examples of the residual (6.1) when solving the model problem from Section 2. In the particular case of using a standard spectral Galerkin method (including an affine mapping), all the integrals in the weak form are computed without any quadrature error using standard GLL quadrature, including the right hand side since f is only a constant. The numerical solution is therefore also a solution of the strong form of the problem, and hence the residual is zero at the mesh nodes. This is not the case for the ET solution, which gives zero residual at other points throughout the domain. The standard method gives poor approximation and large residual in the entire domain for small N, whereas the ET method has managed a good approximation of the outer solution even for relatively small N. The residual is relatively large in the boundary layer, but here it decreases very rapidly as N increases.

In the following, we will only consider the strong form of the residual when discussing potential ways to construct adaptive spectral methods. This choice has been made in order to simplify as much as possible the implementation and assessment of such an approach.

6.2 Unconstrained minimization of the residual

A measure of the quality of the numerical solution of the boundary value problem can be given by the residual of the strong form measured in the L^2-norm,

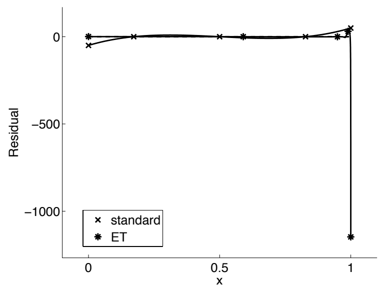
$$||f - f_N||^2_{L^2(\Omega)} = \int_\Omega r_N(x)^2 \, \mathrm{d}x, \tag{6.3}$$

which is most conveniently evaluated over the reference domain. Mapping the residual to the reference domain yields

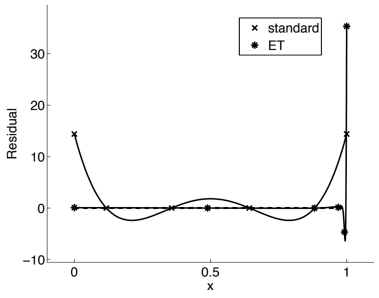
$$\hat{r}_N(\xi) = \hat{f}(\xi) - \hat{f}_N(\xi), \tag{6.4}$$

where $\hat{f}(\xi) = f(\mathcal{F}_N(\xi))$ and

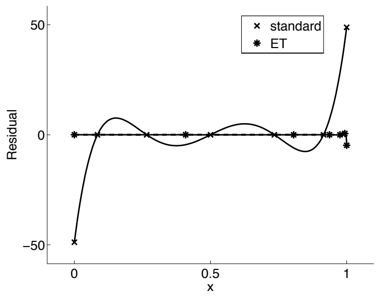
$$\hat{f}_N(\xi) = -\varepsilon \frac{\hat{u}''_N(\xi)\mathcal{F}'_N(\xi) - \mathcal{F}''_N(\xi)\hat{u}'_N(\xi)}{(\mathcal{F}'_N(\xi))^3} + \frac{\hat{u}'_N(\xi)}{\mathcal{F}'_N(\xi)} \tag{6.5}$$



(a) $N = 4$



(b) $N = 5$



(c) $N = 6$

Figure 5: The residual (6.1) of the strong form after computing the spectral Galerkin solution of the advection-diffusion problem (2.1) with exact solution (2.2). The standard method (i.e., using an affine mapping) results in a large residual over the entire domain, whereas a non-affine mapping produced by the ET method confines the inaccuracy to a very thin boundary layer.

Applying quadrature with over-integration in $M \gg N$ points to ensure subdominant quadrature error yields the discrete norm

$$||f - f_N||_N = \Big(\sum_{\alpha=0}^{M} \rho_\alpha \hat{r}_N(\xi_\alpha)^2 \mathcal{F}'_N(\xi_\alpha) \Big)^{1/2}. \tag{6.6}$$

An adaptive method can now in principle be defined by minimizing the functional

$$\mathcal{J} = ||f - f_N||_N^2. \tag{6.7}$$

At this point, both $\mathcal{F}_N$ and $\hat{u}_N$ are unknown. Similar to the case of interpolating the exact solution (see Section 4), the mapping $\mathcal{F}_N$ is given by (4.6) and (4.8). The numerical solution $\hat{u}_N$ is also expressed through a nodal basis similar to (4.7). The simplest implementation of the minimization procedure is achieved when letting both $\xi_1^*, \dots, \xi_{N-1}^*$ and the basis coefficients $u_1^*, \dots, u_{N-1}^*$ be free variables. This means that we do not solve (2.1) numerically at this point, but rather let the minimizer of $\mathcal{J}$ define the mapping $\mathcal{F}_N$ that is subsequently used in a spectral method to compute $\hat{u}_N$. However, we keep a link with the boundary value problem by letting $\xi_0^*, \xi_N^*, u_0^*, u_N^*$ be determined by the boundary conditions.

The minimum of $\mathcal{J}$ occurs at a stationary point, which is a solution of the system

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial \xi_i^*} &= 0, \qquad i = 1, \dots, N-1, \\ \frac{\partial \mathcal{J}}{\partial u_i^*} &= 0, \qquad i = 1, \dots, N-1. \end{aligned} \tag{6.8}$$

Of course, $\mathcal{J}$ may have several stationary points, corresponding to different *local* minima (and maxima). A crude *global* minimization procedure is constructed by solving (6.8) several times, using Newton's method, but with different initial values. The solution that gives the smallest value of $\mathcal{J}$ is then chosen. As the polynomial degree N increases, an exhaustive search for the global minimizer will be infeasible (as the number of free variables becomes too large), but with good initial values this minimization method may be sufficient to find close to optimal solutions.

The result of the minimization procedure is set of variables that define $\mathcal{F}_N$ and $\hat{u}_N$. We discard $\hat{u}_N$ and use a spectral method based on $\mathcal{F}_N$ to solve (2.1) numerically. This completes the adaptive method, which we will refer to as the *unconstrained minimum residual* (UMR) method.

Let us write out the terms in (6.8). The partial derivative of $\mathcal{J}$ with respect to ξ_i^* is

$$\frac{\partial \mathcal{J}}{\partial \xi_i^*} = \sum_{\alpha=0}^{M} \rho_\alpha \left(2\, \hat{r}_N(\xi_\alpha)\, \frac{\partial \hat{r}_N}{\partial \xi_i^*}(\xi_\alpha) \mathcal{F}_N'(\xi_\alpha) + \hat{r}_N(\xi_\alpha)^2\, \frac{\partial \mathcal{F}_N'}{\partial \xi_i^*}(\xi_\alpha) \right), \tag{6.9}$$

where prime means derivative with respect to the free variable ξ on $\widehat{\Omega}$. The partial derivative of the residual is

$$\begin{aligned} \frac{\partial \hat{r}_N}{\partial \xi_i^*} =& f'(\mathcal{F}_N(\xi)) \frac{\partial \mathcal{F}_N}{\partial \xi_i^*} + \hat{u}_N'(\xi) \mathcal{F}_N'(\xi)^{-2} \frac{\partial \mathcal{F}_N'}{\partial \xi_i^*} \\ &+ \varepsilon \mathcal{F}_N'(\xi)^{-4} \left(3\hat{u}_N'(\xi) \mathcal{F}_N''(\xi) \frac{\partial \mathcal{F}_N'}{\partial \xi_i^*} - \hat{u}_N'(\xi) \mathcal{F}_N'(\xi) \frac{\partial \mathcal{F}_N''}{\partial \xi_i^*} - 2\hat{u}_N''(\xi) \mathcal{F}_N'(\xi) \frac{\partial \mathcal{F}_N'}{\partial \xi_i^*} \right). \end{aligned} \tag{6.10}$$

Note that $\hat{u}_N$ and its derivatives do not depend on ξ_i^* since $\hat{u}_N$ is not a numerical solution of the boundary value problem at this point; it is one of the unknowns. From the representation (4.6) of $\mathcal{F}_N$ we see that

$$\frac{\partial \mathcal{F}_N}{\partial \xi_i^*} = c\, \ell_i(\xi), \qquad \frac{\partial \mathcal{F}_N'}{\partial \xi_i^*} = c\, \ell_i'(\xi) \qquad \text{and} \qquad \frac{\partial \mathcal{F}_N''}{\partial \xi_i^*} = c\, \ell_i''(\xi), \tag{6.11}$$

where $c = \overline{\mathcal{F}}'(\xi)$ is a constant. The partial derivative of $\mathcal{J}$ with respect to u_i^* is simply

$$\frac{\partial \mathcal{J}}{\partial u_i^*} = 2 \sum_{\alpha=0}^{M} \rho_\alpha \hat{r}_N(\xi_\alpha) \frac{\partial \hat{r}_N}{\partial u_i^*}(\xi_\alpha) \mathcal{F}_N'(\xi_\alpha), \tag{6.12}$$

since $\mathcal{F}_N$ does not depend on any of the u_i^*. The partial derivative of the residual is

$$\frac{\partial \hat{r}_N}{\partial u_i^*} = -\frac{\varepsilon}{\mathcal{F}_N'(\xi)^2} \frac{\partial \hat{u}_N''}{\partial u_i^*} + \left(\frac{\varepsilon \mathcal{F}_N''(\xi)}{\mathcal{F}_N'(\xi)^3} + \frac{1}{\mathcal{F}_N'(\xi)} \right) \frac{\partial \hat{u}_N'}{\partial u_i^*}, \tag{6.13}$$

where

$$\frac{\partial \hat{u}_N'}{\partial u_i^*} = \ell_i'(\xi) \qquad \text{and} \qquad \frac{\partial \hat{u}_N''}{\partial u_i^*} = \ell_i''(\xi). \tag{6.14}$$

The expressions above are needed just to evaluate (6.8). Newton's method additionally requires the partial derivatives of each equation in (6.8) with respect to each free variable. The resulting $2(N-1) \times 2(N-1)$ Hessian matrix is quite complicated, but can be found explicitly through repeated partial differentiation.

We apply the UMR method in the numerical solution of the advection-diffusion problem (2.1) with $f = 1$ and exact solution (2.2). Figure 6 shows that the UMR method almost keeps up with the ET method (as discussed in Section 4) for small N, but that the convergence rate decreases as N increases. This is most likely due to the difficulty of solving the global minimization problem.

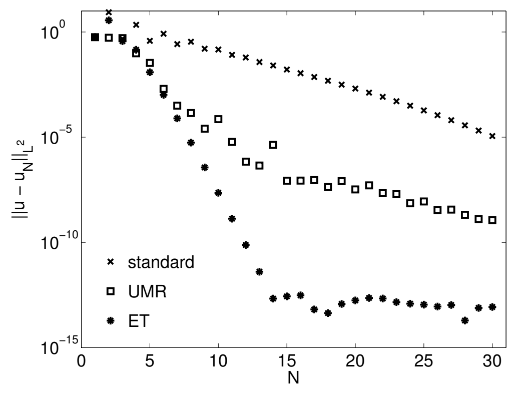


Figure 6: The error in the numerical solution of the advection-diffusion problem (2.1) with exact solution (2.2). The UMR method gives an improvement compared to a standard spectral method (i.e., using an affine mapping), and it keeps up with the solution obtained by the ET-method (as discussed in Section 4) for low values of N. However, as N increases the convergence rate decreases due to the complexity of the system of nonlinear equations that must be solved.

6.3 Constrained minimization of the residual

One possible way to improve the UMR method is to solve the boundary value problem in conjunction with the minimization problem. In the UMR method, the minimization procedure does not take into account which numerical method is used to ultimately solve the boundary value problem. The minimization procedure would be the same if we switched from a spectral method to a PS method or another method.

Solving the minimization problem and the boundary value problem simultaneously means minimizing $\mathcal{J}$ subject to the *constraint* that $\hat{u}_N$ is a numerical solution of (2.1). The resulting adaptive method will be referred to as the *minimum residual* (MR) method, and it is most conveniently implemented by considering a PS method for the numerical solution of the boundary value problem. This means that the constraints are $\hat{r}_N(\xi_j) = 0$, $j = 1, \ldots, N-1$, i.e., vanishing residual (of the strong form) in the internal GLL points. The constrained minimization problem can be solved by considering the Lagrange function

$$\Lambda = \mathcal{J} + \sum_{k=1}^{N-1} \lambda_k \hat{r}_N(\xi_k), \tag{6.15}$$

where λ_k are the Lagrange multipliers. Solutions of the constrained minimization problem are stationary points for Λ, which is a function of $\xi_1^*, \ldots, \xi_{N-1}^*$, $u_1^*, \ldots, u_{N-1}^*$ and $\lambda_1, \ldots, \lambda_{N-1}$. Finding a stationary point means solving a system of $3(N-1)$ nonlinear equations,

$$\begin{aligned} \frac{\partial \Lambda}{\partial \xi_i^*} &= \frac{\partial \mathcal{J}}{\partial \xi_i^*} + \sum_{k=1}^{N-1} \lambda_k \frac{\partial \hat{r}_N}{\partial \xi_i^*}(\xi_k) &&= 0, \qquad && i = 1, \ldots, N-1, \\ \frac{\partial \Lambda}{\partial u_i^*} &= \frac{\partial \mathcal{J}}{\partial u_i^*} + \sum_{k=1}^{N-1} \lambda_k \frac{\partial \hat{r}_N}{\partial u_i^*}(\xi_k) &&= 0, \qquad && i = 1, \ldots, N-1, \\ \frac{\partial \Lambda}{\partial \lambda_i} &= \hat{r}_N(\xi_i) &&= 0, \qquad && i = 1, \ldots, N-1. \end{aligned} \tag{6.16}$$

Note that even if this system is larger than (6.8), it is not much more complicated. The partial derivatives of $\mathcal{J}$ are the same as (6.9) and (6.12), and due to our choice of using a PS method we can reuse the partial derivatives (6.10) and (6.13) of $\hat{r}_N$.

In order to find solutions of (6.16) we again employ Newton's method. The implementation is basically the same as for the UMR method.

The MR method should, by construction, give the optimal solution in terms of the residual, which again is very close to the optimal solution as defined in Section 4. However, this requires us to find the best of all possible solutions to (6.16). Figure 7 shows that we get the same convergence rate as the ET method for low values of N; this is an indication that the solution may be close to optimal. However, the convergence rate decreases as N increases, just as in Figure 6, most likely for the same reason.

7 Adaptivity through residual-based interpolation

The excellent results achieved with the ET method applied to the exact solution in Section 4 motivates an investigation of how it may be used in an adaptive method, but now without using any information about the exact solution. We

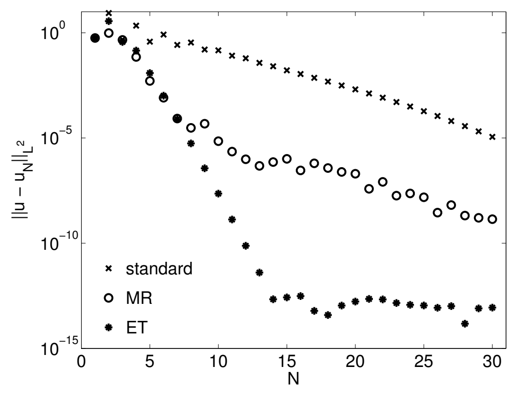


Figure 7: The error in the numerical solution of the advection-diffusion problem (2.1) with exact solution (2.2). The MR method almost keeps up with the ET method (as discussed in Section 4) for small values of N, but the convergence rate decreases as N increases since the minimization problem becomes more difficult to solve.

first recall that the ET method is based on using the available degrees-of-freedom to ensure that the error in the derivative is zero at the internal interpolation points. In the context of solving PDEs, the collocation method gives zero residual at the grid points, but has typically nonzero derivatives at these points, e.g., see the results in Figure 5 for the standard method. A parallel to the ET method would be to use the available degrees-of-freedom to also make the derivative of the residual zero at the internal grid points. This means finding a mapping $\mathcal{F}_N(\xi)$ such that

$$\hat{r}'_N(\xi_j) = \hat{f}'(\xi_j) - \hat{f}'_N(\xi_j) = 0, \qquad j = 1, \ldots, N-1. \tag{7.1}$$

To make sure that zero residual is also satisfied, we must solve these equations together with the collocation equations

$$\hat{r}_N(\xi_j) = \hat{f}(\xi_j) - \hat{f}_N(\xi_j) = 0, \qquad j = 1, \ldots, N-1. \tag{7.2}$$

As before, we use Newton's method for the solution of the coupled system of nonlinear equations. The Jacobian matrix of the system can be derived using (6.10) and (6.13) and normal differentiation rules. Since the system may have more than one solution, we repeat our strategy of solving the system several times with different initial values, and choose the solution with the smallest

L^2-error. We will refer to this method as the *equal-tangent residual* (ETR) method.

Figure 8 shows a convergence plot for the same test problem as before. There is some improvement from the standard method, but the new method is not able to find the optimal solution.

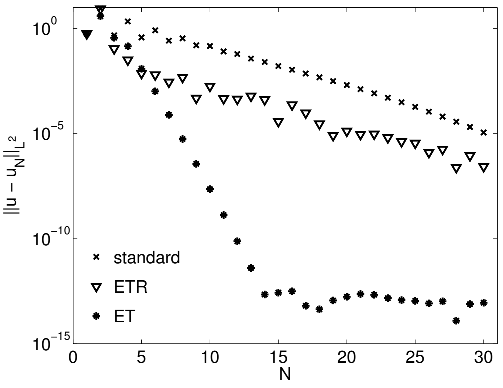


Figure 8: The error in the numerical solution of the model problem (2.1) with exact solution (2.2). The ETR method gives a certain improvement from the standard method, but does not match the convergence rate of the ET method.

8 Additional numerical results

Let us now consider a few more numerical examples in order to compare the various methods proposed earlier. Again, we consider the advection-diffusion boundary value problem (2.1) with $\varepsilon = 0.01$, but now for different choices of f.

Figure 9 shows a convergence plot when the exact solution is given as

$$u(x) = \frac{1}{2}\left(1 - \frac{\operatorname{erf}(x/\sqrt{\varepsilon})}{\operatorname{erf}(1/\sqrt{\varepsilon})}\right), \qquad x \in [-1, 1]. \tag{8.1}$$

This solution has an interior layer of width $\mathcal{O}(\varepsilon)$ around $x = 0$. The ETR and the UMR method only keep up with the ET method for very low N; the error then decreases very slowly as N increases. The MR method gives better results, since it is able to follow the ET convergence rate a little longer.

When the exact solution is given as

$$u(x) = \frac{1}{2}(1-x)(\arctan(\frac{1+x}{(2-a)\varepsilon}) + \arctan(\frac{a}{\varepsilon})), \qquad x \in [-1,1], \qquad (8.2)$$

where $a = 0.35$, we get an interior layer of width $\mathcal{O}(\varepsilon)$ near $x = 2a - 1 = -0.3$. Figure 10 shows that the UMR method is not very successful in this case, although it suddenly finds a better solution after $N = 20$. This may be caused by a lucky choice of initial value for the Newton iterations. The ETR method, on the other hand, gives markedly better performance, and the MR method gives an even better convergence rate.

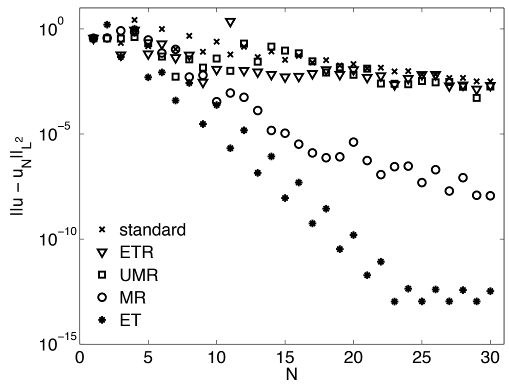


Figure 9: The error in the numerical solution of the advection-diffusion problem (2.1) with exact solution (8.1), which has an interior layer near $x = 0$. Of the adaptive methods, only the MR method gives a clear improvement from the standard method.

9 Conclusions

We have proposed several methods for constructing coordinate transformations for spectral and PS methods, and these have sometimes yielded significant improvement compared to standard methods. Improvement here means smaller discretization error for a fixed polynomial degree, N, used in the approximation.

The methods that are based on a direct minimization of the residual of the strong form should, by construction, give close to optimal solutions. However, these methods require proper global minimization procedures, something which

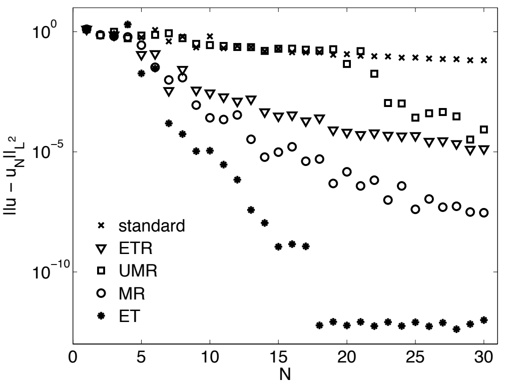


Figure 10: The error in the numerical solution of the advection-diffusion problem (2.1) with exact solution (8.2), which has an interior layer near $x = -0.3$. Again the adaptive methods lose track of the ET convergence rate very early, but they maintain a good convergence rate as N increases, making them far more accurate than the standard method.

is very difficult and computationally expensive to achieve. The minimization method used in this work sometimes yields close to optimal convergence rate for low N, but the convergence rate often decreases as N increases; this is primarily due to the increasing difficulty of solving the global minimization problem. One conclusion from this effort is therefore that it is difficult to construct efficient and robust adaptive methods based on minimization of the residual since good global minimization procedures are often computationally expensive, and the added cost typically outweigh the increased accuracy for a given N.

We have also proposed a way to construct close to optimal nonlinear mappings for spectral and PS methods based on information about the exact solution. The approach uses the equal-tangent (ET) method investigated in [7] to first construct an interpolant of the exact solution, viewed as a parametric curve, and then employs this interpolant to define the mapping used in a subsequent spectral Galerkin method to solve the given boundary value problem. The method does not qualify as an adaptive method since it requires knowledge of the exact solution, but it is useful for providing benchmark solutions to model problems.

An adaptive method based on the same idea as the ET method has also been proposed; this method is based on requiring zero derivative of the residual of the strong form in the nodal points of the solution. The method often gives an improvement from the standard spectral method, but the convergence rate is rarely optimal.

The methods proposed and tested in this paper are relatively expensive computationally. A cost-benefit test has not been performed, since efficiency has not been the main focus of this work. The problem of finding optimal adaptive solutions of even the simplest boundary value problems is so hard that even brute-force solutions do not always succeed. Hence, just finding solutions has taken precedence over efficiency in this preliminary study. A lot of work remains to be done in order to find efficient and robust adaptive methods for the numerical solutions of PDEs using spectral methods.

Acknowledgment

The authors would like to thank Professor Yvon Maday of Paris VI for many helpful comments and suggestions. This work has been supported by the Research Council of Norway under contract 185336/V30. The support is gratefully acknowledged.

Bibliography

[1] J. M. Augenbaum. An adaptive pseudospectral method for discontinuous problems. *Appl. Num. Math.*, 5(6):459–480, 1989.

[2] I. Babuška and W. C. Rheinboldt. Analysis of optimal finite-element meshes in $\mathbf{R}^1$. *Math. Comp.*, 33(146):435–463, 1979.

[3] R. Baltensperger, J.-P. Berrut, and B. Noël. Exponential convergence of a linear rational interpolant between transformed Chebyshev points. *Math. Comp.*, 68(227):1109–1120, 1999.

[4] A. Bayliss, D. Gottlieb, B. J. Matkowsky, and M. Minkoff. An adaptive pseudo-spectral method for reaction diffusion problems. *J. Comput. Phys.*, 81(2):421–443, 1989.

[5] A. Bayliss and B. J. Matkowsky. Fronts, relaxation oscillations, and period doubling in solid fuel combustion. *J. Comput. Phys.*, 71(1):147–168, 1987.

[6] C. Bernardi and Y. Maday. Spectral methods. *In: P. G. Ciarlet, J. L. Lions (eds.), Handbook of Numerical Analysis, Volume V: Techniques of Scientific Computing (Part 2)*, pages 209–485, 1997.

[7] T. Bjøntegaard, E. M. Rønquist, and Ø. Tråsdahl. Spectral approximation of partial differential equations in highly distorted domains. *J. Sci. Comput.*, 2012. To appear. DOI: 10.1007/s10915-011-9561-8.

[8] F. Brezzi, M. O. Bristeau, L. P. Franca, M. Mallet, and G. Rogé. A relationship between stabilized finite element methods and the Galerkin method with bubble functions. *Comput. Methods Appl. Mech. Engrg.*, 96(1):117–129, 1992.

[9] C. Canuto. Stabilization of spectral methods by finite element bubble functions. *Comput. Methods Appl. Mech. Engrg.*, 116(1-4):13–26, 1994. ICOSAHOM'92 (Montpellier, 1992).

[10] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods, Fundamentals in Single Domains.* Springer, 2006.

[11] G. F. Carey and H. T. Dinh. Grading functions and mesh redistribution. *SIAM J. Numer. Anal.*, 22(5):1028–1040, 1985.

[12] T.-F. Chen, G. J. Fix, and H. D. Yang. Numerical studies of optimal grid construction. *Numer. Methods Partial Differential Equations*, 12(2):191–206, 1996.

[13] D. Funaro. A new scheme for the approximation of advection-diffusion equations by collocation. *SIAM J. Numer. Anal.*, 30(6):1664–1676, 1993.

[14] N. Hale and T. W. Tee. Conformal maps to multiply slit domains and applications. *SIAM J. Sci. Comput.*, 31(4):3195–3215, 2009.

[15] W. Z. Huang and D. M. Sloan. A simple adaptive grid method in two dimensions. *SIAM J. Sci. Comput.*, 15(4):776–797, 1994.

[16] D. Kosloff and H. Tal-Ezer. A modified Chebyshev pseudospectral method with an $O(N^{-1})$ time step restriction. *J. Comput. Phys.*, 104(2):457–469, 1993.

[17] W. B. Liu and J. Shen. A new efficient spectral Galerkin method for singular perturbation problems. *J. Sci. Comput.*, 11(4):411–437, 1996.

[18] K. Mørken. On geometric interpolation of parametric surfaces. *Comput. Aided Geom. Design*, 22(9):838–848, 2005.

[19] L. S. Mulholland, W.-Z. Huang, and D. M. Sloan. Pseudospectral solution of near-singular problems using numerical coordinate transformations based on adaptivity. *SIAM J. Sci. Comput.*, 19(4):1261–1289, 1998.

[20] F. Pasquarelli and A. Quarteroni. Effective spectral approximations of convection-diffusion equations. *Comput. Methods Appl. Mech. Engrg.*, 116(1-4):39–51, 1994. ICOSAHOM'92 (Montpellier, 1992).

[21] J. Shen and L.-L. Wang. Error analysis for mapped Legendre spectral and pseudospectral methods. *SIAM J. Numer. Anal.*, 42(1):326–349, 2004.

[22] E. Tadmor. Convergence of spectral methods for nonlinear conservation laws. *SIAM J. Numer. Anal.*, 26(1):30–44, 1989.

[23] E. Tadmor. Shock capturing by the spectral viscosity method. *Comput. Methods Appl. Mech. Engrg.*, 80(1-3):197–208, 1990. Spectral and high order methods for partial differential equations (Como, 1989).

[24] T. Tang and M. R. Trummer. Boundary layer resolving pseudospectral methods for singular perturbation problems. *SIAM J. Sci. Comput.*, 17(2):430–438, 1996.

[25] T. W. Tee and L. N. Trefethen. A rational spectral collocation method with adaptively transformed Chebyshev grid points. *SIAM J. Sci. Comput.*, 28(5):1798–1811, 2006.

[26] R. Verfürth. *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques.* John Wiley & Sons Inc, 1996.

[27] L.-L. Wang and J. Shen. Error analysis for mapped Jacobi spectral methods. *J. Sci. Comput.*, 24(2):183–218, 2005.