

## FINITE DIFFERENCE OPERATORS FROM MOVING LEAST SQUARES INTERPOLATION

HENNADIY NETUZHYLOV<sup>1</sup>, THOMAS SONAR AND WARISA YOMSATIEANKUL

**Abstract.** In a foregoing paper [Sonar, *ESAIM: M2AN* **39** (2005) 883–908] we analyzed the Interpolating Moving Least Squares (IMLS) method due to Lancaster and Šalkauskas with respect to its approximation powers and derived finite difference expressions for the derivatives. In this sequel we follow a completely different approach to the IMLS method given by Kunle [*Dissertation* (2001)]. As a typical problem with IMLS method we address the question of getting admissible results at the boundary by introducing “ghost points”. Most interesting in IMLS methods are the finite difference operators which can be computed from different choices of basis functions and weight functions. We present a way of deriving these discrete operators in the spatially one-dimensional case. Multidimensional operators can be constructed by simply extending our approach to higher dimensions. Numerical results ranging from 1-d interpolation to the solution of PDEs are given.

**Mathematics Subject Classification.** 65M06, 65M60, 65F05.

Received February 10, 2004. Revised September 27, 2006 and May 30, 2007.

### 1. INTRODUCTION

The present paper is the sequel to [7] and was already announced there. Therefore we refer the reader to the aforementioned paper for questions of the history of the method and the place in the literature into which our research fits.

The plan of the present paper is as follows. We start with a natural derivation of the IMLS method in form of the normal equations of a certain optimization problem. The resulting coefficient matrices turn out to be singular, and instead of exploiting the Ansatz of Lancaster and Šalkauskas [3] we follow ideas of Kunle [2] in subdividing the coefficient matrix into a regular and a singular part and then applying an asymptotic analysis in order to actually do the inversion. As a result we get a representation of the IMLS interpolant in terms of so-called kernel functions. However, although the inverse can be computed theoretically it numerically represents an awkward operator with very large condition number. While the inverse can be handled inside the interpolation domain by means of fairly standard singular value decomposition the situation becomes much worse near the boundary of the domain. The condition number of the inverse coefficient matrix blows up and intolerable errors occur at the boundary which travel well inside the domain. It is this phenomenon which we address by introducing ghost points very similar to the usual treatment in finite difference methods. Furthermore, we compute derivatives of the IMLS interpolant in arbitrary dimensions. Although we show only the case of first and second derivatives it is obvious how derivatives of arbitrary order can be computed. Finally we derive finite

---

*Keywords and phrases.* Difference operators, moving least squares interpolation, order of approximation.

<sup>1</sup> TU Braunschweig, Computational Mathematics, Pockelstrasse 14, 38106 Braunschweig, Germany. [t.sonar@tu-bs.de](mailto:t.sonar@tu-bs.de)  
© EDP Sciences, SMAI 2007

difference formulae from 1-d IMLS interpolants by computing the derivatives of the kernel functions. We have used linear and quadratic polynomial basis functions only but our computation is applicable to any sensible basis functions. Having the derivatives of the kernel functions at hand we derive the actual finite difference operators as in the previous paper [7]. We conclude with numerical examples ranging from 1-d interpolation to the solution of the 2-d Poisson equation.

## 2. OVERVIEW OF WEIGHTED MLS METHODS

### 2.1. Structure of the approximating functions

Given  $n$  data pairs  $(x_i, u_i), i = 1, \dots, n$ , the moving least squares method (MLS) is defined by assigning weights  $w_i(x) := w(|x - x_i|), i = 1, \dots, n$  to each point and then asking for the minimum of the functional

$$E_x(\mathbf{a}) = \sum_{i=0}^n w(|x - x_i|)(p(x_i) - u_i)^2,$$

where we can think of  $p$  as being a polynomial  $p(x) = \sum a_k x^k$  of some fixed degree. However, as we shall see immediately, the function  $p$  minimizing the functional has  $x$ -dependent coefficients, *i.e.*  $p$  will not be a polynomial. On the other hand we want to be free to choose any set of basis functions and not only the monomials, so that instead of a polynomial we introduce  $p(x) := Gu(x) := \sum_{j=1}^k a_j b^{(j)}(x)$  as a function from a finite dimensional space of dimension  $k$  with basis  $\mathbf{b}^\top(x) := \{b_1(x), \dots, b_k(x)\}$  and coefficients  $\mathbf{a}^\top := (a_1, \dots, a_k) \in \mathbb{R}^k$ . Note that the weights can be chosen completely independent from the basis functions. In our case singular weights are used in order to enforce the interpolating property.

The minimum is given as the solution of the normal equations

$$\mathbf{B}\mathbf{W}(x)\mathbf{B}^\top \mathbf{a}(x) = \mathbf{B}\mathbf{W}(x)\mathbf{u}, \tag{1}$$

where

$$\mathbf{B} := \begin{pmatrix} b_1(x_1) & \cdots & b_1(x_n) \\ \vdots & \ddots & \vdots \\ b_n(x_1) & \cdots & b_n(x_n) \end{pmatrix}$$

and

$$\mathbf{W}(x) := \text{diag}(w_1(x), \dots, w_n(x)),$$

see [7]. Note that the matrix  $\mathbf{W}$  of weights depends on the point  $x$ . Hence, the coefficients  $\mathbf{a}$  will in general depend on  $x$  and, consequently, the moving least squares function

$$Gu(x) = \sum_{j=1}^k a_j(x)b_j(x) = \mathbf{a}^\top(x) \cdot \mathbf{b}(x) = \langle \mathbf{a}(x), \mathbf{b}(x) \rangle \tag{2}$$

will not be a polynomial even if the  $b_j$  are the monomials.

Since  $\mathbf{a}(x)$  is a solution of (1) we can write, at least formally,

$$\mathbf{a}(x) = (\mathbf{B}\mathbf{W}(x)\mathbf{B}^\top)^{-1} \mathbf{B}\mathbf{W}(x)\mathbf{u}. \tag{3}$$

On the other hand we would like to be able to write the interpolant in the form

$$Gu(x) = \sum_{i=1}^n u_i \varphi_i(x) = \mathbf{u}^\top \cdot \boldsymbol{\varphi}(x) \tag{4}$$

with some *kernel functions*  $\varphi_i, i = 1, \dots, n$ .

Knowing the kernel functions, the derivatives of the interpolating function are obtained as

$$g'(x) = \left\langle \mathbf{u}, \frac{d}{dx} \boldsymbol{\varphi}(x) \right\rangle \quad (5)$$

and

$$g''(x) = \left\langle \mathbf{u}, \frac{d^2}{dx^2} \boldsymbol{\varphi}(x) \right\rangle. \quad (6)$$

Comparing (2) with (4) yields the requirement

$$g(x) = \langle \mathbf{a}(x), \mathbf{b}(x) \rangle \stackrel{!}{=} \langle \mathbf{u}, \boldsymbol{\varphi}(x) \rangle,$$

*i.e.* with the help of (3)

$$\begin{aligned} \langle \mathbf{a}(x), \mathbf{b}(x) \rangle &= \left\langle (\mathbf{B}\mathbf{W}(x)\mathbf{B}^\top)^{-1} \mathbf{B}\mathbf{W}(x)\mathbf{u}, \mathbf{b}(x) \right\rangle \\ &= \left\langle \mathbf{u}, \left[ (\mathbf{B}\mathbf{W}(x)\mathbf{B}^\top)^{-1} \mathbf{B}\mathbf{W}(x) \right]^\top \mathbf{b}(x) \right\rangle \\ &= \left\langle \mathbf{u}, \mathbf{W}(x)\mathbf{B}^\top (\mathbf{B}\mathbf{W}(x)\mathbf{B}^\top)^{-1} \mathbf{b}(x) \right\rangle \end{aligned}$$

and hence

$$\boldsymbol{\varphi}(x) = \mathbf{W}(x)\mathbf{B}^\top (\mathbf{B}\mathbf{W}(x)\mathbf{B}^\top)^{-1} \mathbf{b}(x). \quad (7)$$

We are not interested in intrinsic properties of the moving least squares method but in the expression for derivatives of certain orders at the nodes. Our background is in meshless methods for evolutionary partial differential equations. We would like to construct finite difference operators from moving least squares functions which can then be used in numerical methods for PDEs.

## 2.2. Basis, weight, and kernel functions

We would like our basis functions to enjoy the following properties: ease of computation and completeness. The completeness means that any given order of accuracy can be achieved by taking the order of basis function large enough.

The simplest and widely used basis functions are the polynomial basis functions of degree  $k - 1$ :

$$\mathbf{b}(x) := (1, x, x^2, \dots, x^{k-1})^\top.$$

Different weights are reported in the literature, see [3, 4], and in case of approximation problems their choice mostly depends on personal tastes. In order to compare with [7] we again choose the singular weights

$$w_i(x) := \frac{1}{(x - x_i)^\alpha}, \quad \alpha = 2m, m \in \mathbf{N} \quad (8)$$

so that our method belongs to the class of Interpolating Moving Least Squares (IMLS) methods.

In (7) we have derived the expression for the kernel function  $\varphi(x)$ . Now let us look at its properties.

- **Consistency**

Interpolating data from the basis functions leads to

$$\mathbf{B} \cdot \boldsymbol{\varphi}(x) = \mathbf{B}\mathbf{W}(x)\mathbf{B}^\top (\mathbf{B}\mathbf{W}(x)\mathbf{B}^\top)^{-1} \mathbf{b}(x) = \mathbf{b}(x),$$

*i.e.* the basis functions are reproduced *exactly* by the IMLS method. This property is called *consistency* of the IMLS method.

• **Partition of unity**

As a special case of consistency we consider the basis function  $b_1(x) \equiv 1$  which leads to

$$\sum_{j=1}^n \varphi_j(x) = 1. \tag{9}$$

Hence, our kernel functions comprise a partition of unity.

Applying the derivative operator to equation (9) we obtain that the sum of the first derivatives of all kernel functions vanishes,

$$\sum_{j=1}^n \frac{d}{dx} \varphi_j(x) = 0. \tag{10}$$

3. REPRESENTATION OF THE KERNEL FUNCTIONS

In the non-interpolatory case the inversion of the matrix in (7) does not cause many problems. Since this is a small ( $k \times k$ )-matrix, the inversion could be done *via* the LU-decomposition with pivoting with relatively little effort.

Nonetheless, if the matrix is ill-conditioned, the LU-decomposition may lead to false results as was shown, *e.g.*, in [2]. One should therefore use QR- or Singular Value Decomposition (SVD).

In Interpolating Moving Least Squares the weight function matrix  $\mathbf{W}(x)$  contains a singularity in the  $\ell$ -th row:  $\lim_{x \rightarrow x_\ell} w_\ell(x) = \infty$ . Due to this fact the matrix  $\mathbf{B}\mathbf{W}(x)\mathbf{B}^\top$  in (7) may be singular. In order to compute the inverse of this singular matrix we exploit an idea given in [2].

In [2] the author already outlined the approach we follow here. However, Kunle avoided notations from linear algebra whenever possible and employed a clumsy sum notation which made conclusions concerning higher derivatives of the kernel functions impossible and which did not allow any structural insight. The consistent use of matrices and vectors which we follow here not only clearly reveals the *structure* of the kernel functions but also shows the structure of derivatives of arbitrary order.

In order to temporarily cure the singularity problem we introduce a regularization due to a small positive number  $\varepsilon$  in the form of the regularized weight function matrix  $\mathbf{W}(x + \varepsilon)$ . Then (7) reads as

$$\boldsymbol{\varphi}(x + \varepsilon) = \mathbf{W}(x + \varepsilon)\mathbf{B}^\top (\mathbf{B}\mathbf{W}(x + \varepsilon)\mathbf{B}^\top)^{-1} \mathbf{b}(x + \varepsilon). \tag{11}$$

We then proceed in proving the following:

**Theorem 3.1.** *The matrix  $\mathbf{B}\mathbf{W}(x)\mathbf{B}^\top$  is regular and its inverse at the point  $x_\ell$  is given by*

$$(\mathbf{B}\mathbf{W}(x_\ell)\mathbf{B}^\top)^{-1} = \mathbf{R}^{-1} - \frac{1}{\mathbf{b}^\top(x_\ell)\mathbf{R}^{-1}\mathbf{b}(x_\ell)} (\mathbf{R}^{-1}\mathbf{b}(x_\ell))(\mathbf{R}^{-1}\mathbf{b}(x_\ell))^\top,$$

with a regular and symmetric matrix

$$\mathbf{R} := \begin{bmatrix} \sum_{\substack{i=1 \\ i \neq \ell}}^n b_1(x_i)w_i(x_\ell)b_1(x_i) & \cdots & \sum_{\substack{i=1 \\ i \neq \ell}}^n b_1(x_i)w_i(x_\ell)b_k(x_i) \\ \vdots & \ddots & \vdots \\ \sum_{\substack{i=1 \\ i \neq \ell}}^n b_k(x_i)w_i(x_\ell)b_1(x_i) & \cdots & \sum_{\substack{i=1 \\ i \neq \ell}}^n b_k(x_i)w_i(x_\ell)b_k(x_i) \end{bmatrix}.$$

*Proof.* Regularizing with  $\varepsilon > 0$  we get

$$\begin{aligned} \mathbf{W}(x_\ell + \varepsilon) &= \text{diag}(w_1(x_\ell + \varepsilon), \dots, w_{\ell-1}(x_\ell + \varepsilon), w_\ell(\varepsilon), w_{\ell+1}(x_\ell + \varepsilon), \dots, w_n(x_\ell + \varepsilon)) \\ &= \text{diag}(w_1(x_\ell + \varepsilon), \dots, w_{\ell-1}(x_\ell + \varepsilon), 0, w_{\ell+1}(x_\ell + \varepsilon), \dots, w_n(x_\ell + \varepsilon)) \\ &\quad + \text{diag}(0, \dots, 0, \underbrace{w_\ell(\varepsilon)}_{\ell\text{-th position}}, 0, \dots, 0), \end{aligned}$$

where  $w_\ell(\varepsilon) := w_\ell(x_\ell + \varepsilon)$ .

Hence,

$$\begin{aligned} \mathbf{B}\mathbf{W}(x_\ell + \varepsilon)\mathbf{B}^\top &= \underbrace{\begin{bmatrix} \sum_{\substack{i=1 \\ i \neq \ell}}^n b_1(x_i)w_i(x_\ell + \varepsilon)b_1(x_i) & \cdots & \sum_{\substack{i=1 \\ i \neq \ell}}^n b_1(x_i)w_i(x_\ell + \varepsilon)b_k(x_i) \\ \vdots & \ddots & \vdots \\ \sum_{\substack{i=1 \\ i \neq \ell}}^n b_k(x_i)w_i(x_\ell + \varepsilon)b_1(x_i) & \cdots & \sum_{\substack{i=1 \\ i \neq \ell}}^n b_k(x_i)w_i(x_\ell + \varepsilon)b_k(x_i) \end{bmatrix}}_{=: \mathbf{R}_\varepsilon} \\ &\quad + w_\ell(\varepsilon) \begin{bmatrix} b_1(x_\ell)b_1(x_\ell) & \cdots & b_1(x_\ell)b_k(x_\ell) \\ \vdots & \ddots & \vdots \\ b_k(x_\ell)b_1(x_\ell) & \cdots & b_k(x_\ell)b_k(x_\ell) \end{bmatrix}. \end{aligned}$$

Note that  $\mathbf{R}_\varepsilon$  is regular even in the case  $\varepsilon = 0$ . Using the Sherman-Morrison formula [1] it then follows

$$\begin{aligned} (\mathbf{B}\mathbf{W}(x_\ell + \varepsilon)\mathbf{B}^\top)^{-1} &= \mathbf{R}_\varepsilon^{-1} - \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\mathbf{b}^\top(x_\ell)\mathbf{R}_\varepsilon^{-1}\mathbf{b}(x_\ell)} (\mathbf{R}_\varepsilon^{-1}\mathbf{b}(x_\ell)(\mathbf{R}_\varepsilon^{-1}\mathbf{b}(x_\ell))^\top) \\ &= \mathbf{R}_\varepsilon^{-1} - \frac{1}{\frac{1}{w_\ell(\varepsilon)} + \mathbf{b}^\top(x_\ell)\mathbf{R}_\varepsilon^{-1}\mathbf{b}(x_\ell)} (\mathbf{R}_\varepsilon^{-1}\mathbf{b}(x_\ell)(\mathbf{R}_\varepsilon^{-1}\mathbf{b}(x_\ell))^\top). \end{aligned} \quad (12)$$

Since  $\lim_{\varepsilon \rightarrow 0} w_\ell(\varepsilon) = \infty$  we have  $\lim_{\varepsilon \rightarrow 0} \frac{1}{w_\ell(\varepsilon)} = 0$  as well as  $\lim_{\varepsilon \rightarrow 0} \mathbf{R}_\varepsilon = \mathbf{R}$  and the desired result follows.  $\square$

In order to compute values of (7) at the nodes we consider the regularized version

$$\varphi(x_\ell + \varepsilon) = \mathbf{W}(x_\ell + \varepsilon)\mathbf{B}^\top (\mathbf{B}\mathbf{W}(x_\ell + \varepsilon)\mathbf{B}^\top)^{-1} \mathbf{b}(x_\ell + \varepsilon) \quad (13)$$

and note that

$$\mathbf{W}(x_\ell + \varepsilon)\mathbf{B}^\top = \begin{bmatrix} b_1(x_1)w_1(x_\ell + \varepsilon) & b_2(x_1)w_1(x_\ell + \varepsilon) & \cdots & b_k(x_1)w_1(x_\ell + \varepsilon) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(x_{\ell-1})w_{\ell-1}(x_\ell + \varepsilon) & b_2(x_{\ell-1})w_{\ell-1}(x_\ell + \varepsilon) & \cdots & b_k(x_{\ell-1})w_{\ell-1}(x_\ell + \varepsilon) \\ b_1(x_\ell)w_1(\varepsilon) & b_2(x_\ell)w_1(\varepsilon) & \cdots & b_k(x_\ell)w_1(\varepsilon) \\ b_1(x_{\ell+1})w_{\ell+1}(x_\ell + \varepsilon) & b_2(x_{\ell+1})w_{\ell+1}(x_\ell + \varepsilon) & \cdots & b_k(x_{\ell+1})w_{\ell+1}(x_\ell + \varepsilon) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(x_n)w_n(x_\ell + \varepsilon) & b_2(x_n)w_n(x_\ell + \varepsilon) & \cdots & b_k(x_n)w_n(x_\ell + \varepsilon) \end{bmatrix}, \quad (14)$$

so that we have to consider two different cases, namely

$$\varphi_\ell(x_\ell + \varepsilon)$$

and

$$\varphi_k(x_\ell + \varepsilon), \quad k \neq \ell.$$

**Theorem 3.2.** For  $\ell = 1, \dots, n$  the equations

$$\begin{aligned} \varphi_\ell(x_\ell) &= 1 \\ \varphi'_\ell(x_\ell) &= \frac{1}{\mathbf{b}^\top(x_\ell)\mathbf{R}^{-1}\mathbf{b}(x_\ell)} \mathbf{b}^\top(x_\ell)\mathbf{R}^{-1} \frac{d}{dx} \mathbf{b}(x_\ell) \end{aligned}$$

hold.

*Proof.* Inserting (12) into (13) and introducing the abbreviation

$$\sigma := \mathbf{b}^\top(x_\ell)\mathbf{R}^{-1}\mathbf{b}(x_\ell) \tag{15}$$

yields

$$\varphi(x_\ell + \varepsilon) = \mathbf{W}(x_\ell + \varepsilon)\mathbf{B}^\top \left( \mathbf{R}^{-1} - \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} (\mathbf{R}^{-1}\mathbf{b}(x_\ell))(\mathbf{R}^{-1}\mathbf{b}(x_\ell))^\top \right) \mathbf{b}(x_\ell + \varepsilon)$$

and with the help of (14) we get

$$\varphi_\ell(x_\ell + \varepsilon) = w_\ell(\varepsilon) \left( \mathbf{b}^\top(x_\ell)\mathbf{R}^{-1} - \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} \mathbf{b}^\top(x_\ell)(\mathbf{R}^{-1}\mathbf{b}(x_\ell))(\mathbf{R}^{-1}\mathbf{b}(x_\ell))^\top \right) \mathbf{b}(x_\ell + \varepsilon).$$

The central idea now is to exploit the Taylor series expansion

$$\mathbf{b}(x_\ell + \varepsilon) = \sum_{m=0}^{\infty} \frac{\varepsilon}{m!} \frac{d^m}{dx^m} \mathbf{b}(x_\ell).$$

Note that this series is finite if a polynomial basis is used. However, our analysis is valid for arbitrary bases. Working out the product with the Taylor series yields

$$\begin{aligned} \varphi_\ell(x_\ell + \varepsilon) &= w_\ell(\varepsilon) \left( \sigma - \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} \underbrace{\mathbf{b}^\top(x_\ell)(\mathbf{R}^{-1}\mathbf{b}(x_\ell))(\mathbf{R}^{-1}\mathbf{b}(x_\ell))^\top \mathbf{b}(x_\ell)}_{=\sigma^2} \right) + \varepsilon w_\ell(\varepsilon) \left( \mathbf{b}^\top(x_\ell)\mathbf{R}^{-1} \frac{d}{dx} \mathbf{b}(x_\ell) \right. \\ &\quad \left. - \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} \underbrace{\mathbf{b}^\top(x_\ell)(\mathbf{R}^{-1}\mathbf{b}(x_\ell))}_{=\sigma} \underbrace{(\mathbf{R}^{-1}\mathbf{b}^\top(x_\ell))}_{=\mathbf{b}^\top(x_\ell)\mathbf{R}^{-1}} \frac{d}{dx} \mathbf{b}(x_\ell) \right) + \sum_{m=2}^{\infty} \frac{\varepsilon^m}{m!} w_\ell(\varepsilon) \left( \mathbf{b}^\top(x_\ell)\mathbf{R}^{-1} \frac{d^m}{dx^m} \mathbf{b}(x_\ell) \right. \\ &\quad \left. - \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} \mathbf{b}^\top(x_\ell)(\mathbf{R}^{-1}\mathbf{b}(x_\ell))(\mathbf{R}^{-1}\mathbf{b}(x_\ell))^\top \frac{d^m}{dx^m} \mathbf{b}(x_\ell) \right) \\ &= \frac{w_\ell(\varepsilon)\sigma}{1 + w_\ell(\varepsilon)\sigma} + \varepsilon \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} \mathbf{b}^\top(x_\ell)\mathbf{R}^{-1} \frac{d}{dx} \mathbf{b}(x_\ell) + \sum_{m=2}^{\infty} \frac{\varepsilon^m}{m!} \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} \mathbf{b}^\top(x_\ell)\mathbf{R}^{-1} \frac{d^m}{dx^m} \mathbf{b}(x_\ell). \end{aligned}$$

For  $\varepsilon \rightarrow 0$  we have  $\frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} = \frac{1}{\frac{1}{w_\ell(\varepsilon)} + \sigma} \rightarrow \frac{1}{\sigma}$  and hence  $\varphi_\ell(x_\ell) = 1$ . Furthermore, we have

$$\varphi'_\ell(x_\ell) = \lim_{\varepsilon \rightarrow 0} \frac{\varphi_\ell(x_\ell + \varepsilon) - \varphi_\ell(x_\ell)}{\varepsilon}$$

and therefore from above

$$\varphi'_\ell(x_\ell) = \frac{1}{\sigma} \mathbf{b}^\top(x_\ell) \mathbf{R}^{-1} \frac{d}{dx} \mathbf{b}(x_\ell). \quad \square$$

**Theorem 3.3.** For  $k, \ell = 1, \dots, n$ ,  $k \neq \ell$  the equations

$$\begin{aligned} \varphi_k(x_\ell) &= 0 \\ \varphi'_k(x_\ell) &= w_k(x_\ell) \mathbf{b}^\top(x_k) \left( \mathbf{R}^{-1} - \frac{1}{\sigma} (\mathbf{R}^{-1} \mathbf{b}(x_\ell)) (\mathbf{R}^{-1} \mathbf{b}(x_\ell))^\top \right) \frac{d}{dx} \mathbf{b}(x_\ell) \end{aligned}$$

hold where  $\sigma$  is defined in (15).

*Proof.* We start with

$$\varphi_k(x_\ell + \varepsilon) = w_k(x_\ell + \varepsilon) \mathbf{b}^\top(x_k) \left( \mathbf{R}^{-1} - \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} (\mathbf{R}^{-1} \mathbf{b}(x_\ell)) (\mathbf{R}^{-1} \mathbf{b}(x_\ell))^\top \right) \mathbf{b}(x_\ell + \varepsilon).$$

Utilizing again the Taylor series expansion  $\mathbf{b}(x_\ell + \varepsilon) = \sum_{m=0}^{\infty} \frac{\varepsilon^m}{m!} \frac{d^m}{dx^m} \mathbf{b}(x_\ell)$  yields

$$\begin{aligned} \varphi_k(x_\ell + \varepsilon) &= w_k(x_\ell + \varepsilon) \left( \underbrace{\mathbf{b}^\top(x_k) \mathbf{R}^{-1} \mathbf{b}(x_\ell) - \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} \mathbf{b}^\top(x_k) (\mathbf{R}^{-1} \mathbf{b}(x_\ell)) (\mathbf{R}^{-1} \mathbf{b}(x_\ell))^\top \mathbf{b}(x_\ell)}_{= \frac{1}{1 + w_\ell(\varepsilon)\sigma} \mathbf{b}^\top(x_k) \mathbf{R}^{-1} \mathbf{b}(x_\ell)} \right) \\ &\quad + \varepsilon w_k(x_\ell + \varepsilon) \mathbf{b}^\top(x_k) \left( \underbrace{\mathbf{R}^{-1} - \frac{w_\ell(\varepsilon)}{1 + w_\ell(\varepsilon)\sigma} (\mathbf{R}^{-1} \mathbf{b}(x_\ell)) (\mathbf{R}^{-1} \mathbf{b}(x_\ell))^\top}_{=: \Theta(x_\ell, \varepsilon)} \right) \frac{d}{dx} \mathbf{b}(x_\ell) \\ &\quad + \sum_{m=2}^{\infty} \frac{\varepsilon^m}{m!} w_k(x_\ell + \varepsilon) \mathbf{b}^\top(x_k) \Theta(x_\ell, \varepsilon) \frac{d^m}{dx^m} \mathbf{b}(x_\ell). \end{aligned}$$

For  $\varepsilon \rightarrow 0$  we find a finite value of the weight function  $w_k(x_\ell)$ ,  $\frac{1}{1 + w_\ell(\varepsilon)\sigma} \rightarrow 0$  and  $\frac{w_\ell(\varepsilon)\sigma}{1 + w_\ell(\varepsilon)\sigma} \rightarrow \frac{1}{\sigma}$ , so that

$$\Theta(x_\ell) := \lim_{\varepsilon \rightarrow 0} \Theta(x_\ell, \varepsilon) = \mathbf{R}^{-1} - \frac{1}{\sigma} (\mathbf{R}^{-1} \mathbf{b}(x_\ell)) (\mathbf{R}^{-1} \mathbf{b}(x_\ell))^\top$$

and

$$\varphi_k(x_\ell) = 0.$$

Furthermore, from

$$\varphi'_k(x_\ell) = \lim_{\varepsilon \rightarrow 0} \frac{\varphi_k(x_\ell + \varepsilon) - \varphi_k(x_\ell)}{\varepsilon}$$

we immediately get

$$\varphi'_k(x_\ell) = w_k(x_\ell) \mathbf{b}^\top(x_k) \Theta(x_\ell) \frac{d}{dx} \mathbf{b}(x_\ell). \quad \square$$

Inspecting the proofs of Theorems 3.2 and 3.3 immediately yields:

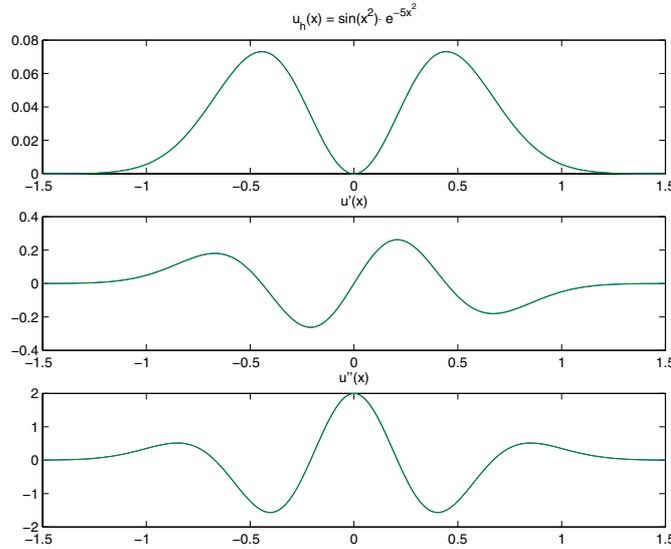


FIGURE 1. Function, its first and second derivative.

**Lemma 3.1.** *For the second derivatives it holds*

$$\begin{aligned} \varphi''_\ell(x_\ell) &= \frac{1}{\mathbf{b}^\top(x_\ell)\mathbf{R}^{-1}\mathbf{b}(x_\ell)} \mathbf{b}^\top(x_\ell)\mathbf{R}^{-1} \frac{d^2}{dx^2} \mathbf{b}(x_\ell) \\ \varphi''_k(x_\ell) &= w_k(x_\ell)\mathbf{b}^\top(x_k) \left( \mathbf{R}^{-1} - \frac{1}{\sigma} (\mathbf{R}^{-1}\mathbf{b}(x_\ell))(\mathbf{R}^{-1}\mathbf{b}(x_\ell))^\top \right) \frac{d^2}{dx^2} \mathbf{b}(x_\ell), \quad k \neq \ell. \end{aligned}$$

#### 4. AN EXAMPLE CONCERNING PROBLEMS AT THE BOUNDARIES

As was already pointed out in [7] and verified through computations the IMLS approximation suffers from problems at the boundaries of the domain. This undesired property is, of course, also present in the approach exploited here. The core of the problems with boundaries is best seen in the condition number of the matrix  $\mathbf{R}$  defined in Theorem 3.1. This matrix which plays a vital role is perfectly regular by construction and we have used the inverse in all of our formulae for the kernel functions. Numerically, however, this matrix is not very well conditioned.

Consider the following interpolation problem. Given the function

$$u(x) := \sin x^2 \cdot e^{-5x^2}$$

on  $[-\frac{3}{2}, \frac{3}{2}]$ . We have generated 121 randomly distributed points in that interval and employed the quadratic basis  $\mathbf{b}(x) = (1, x, x^2)^\top$ . In Figure 1 the IMLS function is shown on top and the first and second derivative, computed by means of the formulae in Theorems 3.2, 3.3 and 3.1.

Figure 2 gives the errors of the interpolation in terms of the supremum norm. At the boundaries one-sided weights were simply used as was the case in [7]. Computing the condition number of the matrix  $\mathbf{R}$  with the help of MATLAB for this case gives the plot as shown in Figure 3. At the boundaries the condition number blows up up to around  $2 \times 10^9$  which led us to the SVD methodology even in the case of this regular matrix  $\mathbf{R}$ .

In order to cure the boundary problems we arrived finally at a classical trick used in finite differences, namely so-called ghost points. Outside the domain auxiliary points are introduced so that symmetric weights can be used also at the boundaries. This approach led to much lower condition numbers of  $\mathbf{R}$  as can be seen in Figure 4.

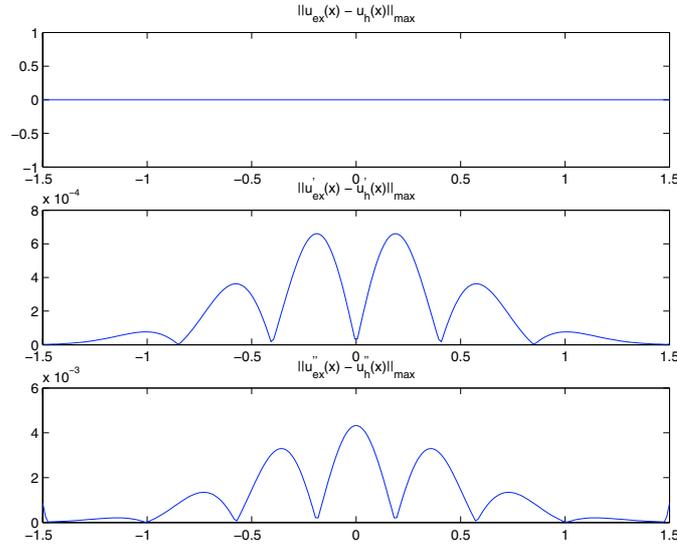


FIGURE 2. Error between the exact and interpolated function, its first and second derivative.

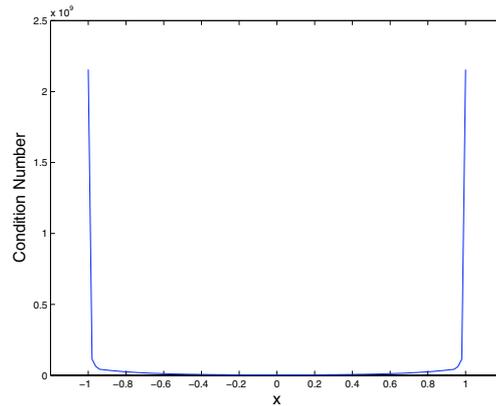


FIGURE 3. Condition number distribution of  $\mathbf{R}$  in the domain.

### 5. DIFFERENCE OPERATORS

In our previous work [7] the derivation of finite difference operators required a considerable work since the asymptotic behaviour of certain expressions had to be studied. In the approach followed here all the work was spent in order to arrive at representation formulae of the kernel functions. If we restrict the support of our weight functions to, say,  $2m + 1$  points centered at  $x_\ell$ , then from (4) and our results on the form of the kernel functions it follows

$$Gu(x_\ell) = u_{\ell-m}\varphi_{\ell-m}(x_\ell) + \dots + u_\ell\varphi_\ell(x_\ell) + \dots + u_{\ell+m}\varphi_{\ell+m}(x_\ell)$$

which leads to

$$\frac{d}{dx}Gu(x_\ell) = u_{\ell-m}\varphi'_{\ell-m}(x_\ell) + \dots + u_\ell\varphi'_\ell(x_\ell) + \dots + u_{\ell+m}\varphi'_{\ell+m}(x_\ell)$$

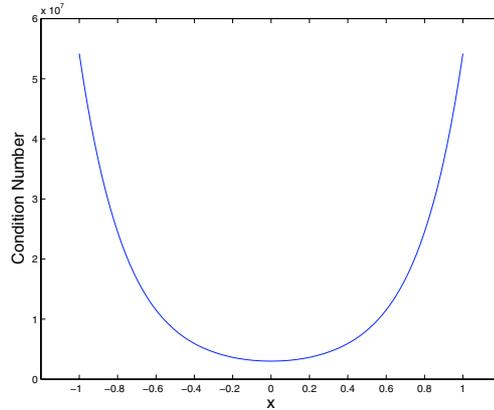


FIGURE 4. Condition number distribution in the domain using the ghost points.

and

$$\frac{d^2}{dx^2}Gu(x_\ell) = u_{\ell-m}\varphi''_{\ell-m}(x_\ell) + \dots + u_\ell\varphi''_\ell(x_\ell) + \dots + u_{\ell+m}\varphi''_{\ell+m}(x_\ell).$$

Thus, all finite difference operators follow directly by evaluating the derivatives of the kernel functions.

From now on we make the assumption that the underlying grid is regular, *i.e.*,  $x_{i+1} - x_i = \Delta x$  for  $i = 1, 2, \dots, n - 1$ .

**5.1. The constant case**

The IMLS interpolation is called *Shepard interpolation* if the basis

$$\mathbf{b}(x) := b_1(x) := 1$$

is employed. Hence, from Theorems 3.2, 3.3 and Lemma 3.1 it follows immediately

$$\frac{d}{dx}Gu(x_\ell) = \frac{d^2}{dx^2}Gu(x_\ell) = 0.$$

Thus, the Shepard interpolant is not suited to compute a derivative of any order.

**5.2. The linear case**

If we restrict ourselves to the basis  $\mathbf{b}(x) := (1, x)^\top$  we can infer from  $\frac{d^2}{dx^2}\mathbf{b}(x) = \mathbf{0}$  and hence from Lemma 3.1 immediately

$$\frac{d^2}{dx^2}Gu(x_\ell) = 0.$$

Thus, in the linear case it makes no sense to rely on second derivatives. For the case of the first derivative we restrict the support of our weights to a three-point stencil, so that

$$\frac{d}{dx}Gu(x_\ell) = u_{\ell-1}\varphi'_{\ell-1}(x_\ell) + u_\ell\varphi'_\ell(x_\ell) + u_{\ell+1}\varphi'_{\ell+1}(x_\ell). \tag{16}$$

**Lemma 5.1.** *The finite difference on an equi-spaced mesh given by the IMLS method with weights of three-point support is*

$$\frac{d}{dx}Gu(x_\ell) = \frac{u_{\ell+1} - u_{\ell-1}}{2\Delta x}.$$

*Proof.* Since  $\alpha$  is even we have

$$w_{\ell-1}(x_\ell) = \frac{1}{(x_\ell - x_{\ell-1})^\alpha} = \frac{1}{\Delta x^\alpha} = \frac{1}{(x_\ell - x_{\ell+1})^\alpha} = w_{\ell+1}(x_\ell)$$

and since  $\mathbf{b}(x_\ell) := (1, x_\ell)^\top$  we conclude therefore from Theorem 3.1 that

$$\begin{aligned} \mathbf{R} &= \begin{bmatrix} w_{\ell-1}(x_\ell) + w_{\ell+1}(x_\ell) & x_{\ell-1}w_{\ell-1}(x_\ell) + x_{\ell+1}w_{\ell+1}(x_\ell) \\ x_{\ell-1}w_{\ell-1}(x_\ell) + x_{\ell+1}w_{\ell+1}(x_\ell) & x_{\ell-1}^2w_{\ell-1}(x_\ell) + x_{\ell+1}^2w_{\ell+1}(x_\ell) \end{bmatrix} \\ &= \frac{2}{\Delta x^\alpha} \begin{bmatrix} 1 & x_\ell \\ x_\ell & x_\ell^2 + \Delta x^2 \end{bmatrix}, \end{aligned}$$

where we have exploited the fact that  $x_{\ell\pm 1} = x_\ell \pm \Delta x$ . The inverse is easily found to be

$$\mathbf{R}^{-1} = \frac{\Delta x^{\alpha-2}}{2} \begin{bmatrix} x_\ell^2 + \Delta x^2 & -x_\ell \\ -x_\ell & 1 \end{bmatrix}.$$

Now  $\frac{d}{dx}\mathbf{b}(x_\ell) = (0, 1)^\top$  and hence  $\mathbf{b}(x_\ell)^\top \mathbf{R}^{-1} \frac{d}{dx}\mathbf{b}(x_\ell) = 0$ , so that we conclude from Theorem 3.2 that

$$\varphi'_\ell(x_\ell) = 0.$$

With a view at Theorem 3.3 we compute

$$\sigma = \mathbf{b}(x_\ell)^\top \mathbf{R}^{-1} \mathbf{b}(x_\ell) = \frac{\Delta x^\alpha}{2}$$

and

$$\mathbf{R}^{-1} \mathbf{b}(x_\ell) = \frac{\Delta x^\alpha}{2} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and hence

$$(\mathbf{R}^{-1} \mathbf{b}(x_\ell))(\mathbf{R}^{-1} \mathbf{b}(x_\ell))^\top = \frac{\Delta x^{2\alpha}}{4} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

This, then, results in

$$\mathbf{M} := \mathbf{R}^{-1} - \frac{1}{\sigma} (\mathbf{R}^{-1} \mathbf{b}(x_\ell))(\mathbf{R}^{-1} \mathbf{b}(x_\ell))^\top = \frac{\Delta x^{\alpha-2}}{2} \begin{bmatrix} x_\ell^2 & -x_\ell \\ -x_\ell & 1 \end{bmatrix}$$

which leads to

$$\mathbf{b}(x_{\ell\pm 1})^\top \mathbf{M} \mathbf{b}(x_\ell) = \pm \frac{\Delta x^{\alpha-1}}{2}.$$

From Theorem 3.3 we conclude

$$\varphi'_{\ell\pm 1}(x_\ell) = w_{\ell\pm 1}(x_\ell) \mathbf{b}(x_{\ell\pm 1})^\top \mathbf{M} \mathbf{b}(x_\ell) = \pm \frac{1}{2\Delta x}.$$

Inserting our results into (16) yields the difference formula sought for.  $\square$

We could now proceed in computing further finite difference operators from the IMLS interpolant but will instead refer to [7] since all the results published there could of course be found with the approach explained in the present work. Instead, we will give some numerical examples.

## 6. THE IMLS AND THE SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS

### 6.1. A linear transport equation

We consider the simple linear transport equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$$

with initial condition

$$u(x, 0) = u_0(x) = e^{-55(x+0.5)^2}$$

on  $[-1, 1]$  where periodic boundary conditions are employed. In order to stabilize<sup>1</sup> the computation we exploit a Lax-Wendroff-type ansatz of the form

$$U_i^{n+1} = U_i^n - \Delta t \frac{d}{dx} Gu(x, t^n) + \frac{\Delta t^2}{2} \frac{d^2}{dx^2} Gu(x, t^n).$$

Here,  $U_i^n$  denotes the value of the IMLS interpolant  $Gu$  at point  $x_i$  and time  $t_n := n\Delta t$ , and the time step  $\Delta t$  is chosen so that the simple CFL condition

$$\text{CFL} = \frac{\Delta t}{\Delta x} < 1$$

is satisfied. We chose  $\text{CFL} = 0.9$ , and the spatial grid consisted of 101 equally spaced points. We used a quadratic monomial basis and restricted the weight functions to a five-point stencil.

The result of the computation for two different parameters  $\alpha$  of the weight function (8) after 56 time steps (corresponding to  $t = 1$ ) is shown in Figure 5.

### 6.2. Burgers' equation

We consider now a nonlinear transport equation, namely the Burgers' equation

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0$$

with  $f(u) := \frac{1}{2}u^2$  and initial condition

$$u(x, 0) = u_0(x) = e^{-55(x+0.5)^2}$$

on  $[-1, 1]$  where again periodic boundary conditions are employed. Employing again the Lax-Wendroff-type ansatz leads to

$$U_i^{n+1} = U_i^n - \Delta t \frac{d}{dx} \frac{1}{2} Gu^2(x, t^n) + \frac{\Delta t^2}{2} \frac{d}{dx} \left( Gu^2(x, t^n) \frac{d^2}{dx^2} Gu(x, t^n) \right).$$

The points are chosen as in the case of the linear transport equation described before and the CFL number is chosen to be 0.9,  $\alpha = 4$ . The result at time  $t = 0.144$  in Figure 6 shows the beginning of a pre-shock oscillation as the shocks starts to form. The oscillation is growing as can also be seen from Figure 6 which shows the solution at time  $t = 0.198$ .

If we increase the number of points to  $N = 801$  and leave all other parameters we get the solutions as shown in Figure 7.

Although we had hoped for a stabilizing effect of the Lax-Wendroff-type ansatz the present numerical results show clearly that this effect is not present. After a few more time steps the oscillations grow beyond all limits

---

<sup>1</sup>It is well known that the discretization of the second spatial derivative in the Lax-Wendroff method acts as a stabilizing artificial diffusivity.

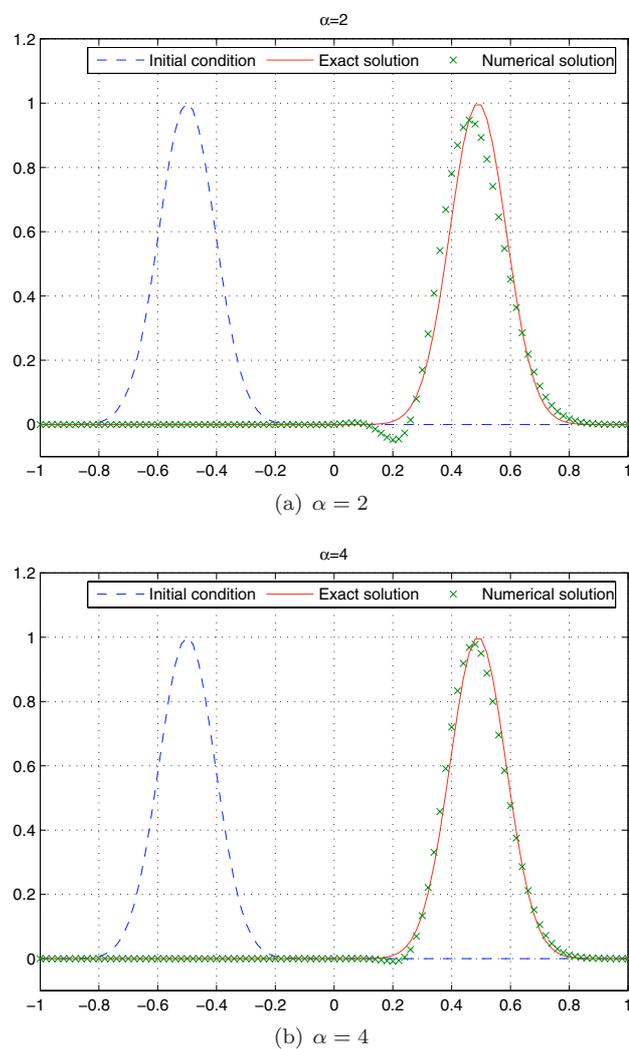


FIGURE 5. Solutions of the linear transport problem.

and the computation stops. We shall come back to these problems and give a thorough analysis in the third part of this series of papers [6].

### 6.3. A multi-dimensional problem

It is easily seen that all of our results can be directly generalized to the multi-dimensional case, if the weights are defined by  $w_k(\mathbf{x}) := (\mathbf{x} - \mathbf{x}_k)^{-\alpha}$  and the one-dimensional derivative is replaced by the gradient.

We solved the boundary value problem

$$\Delta u = 0$$

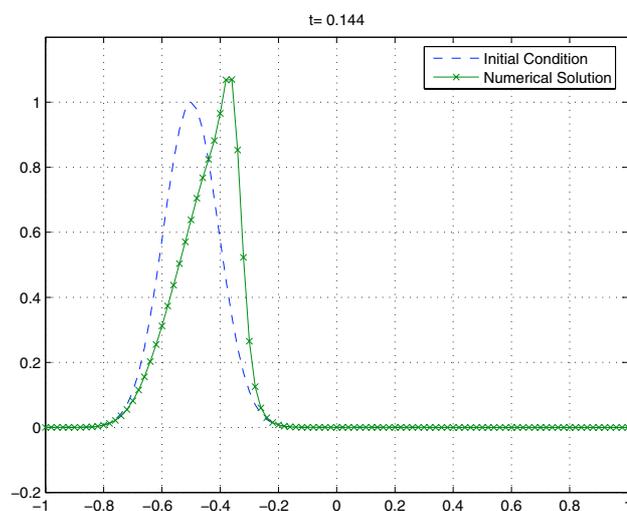
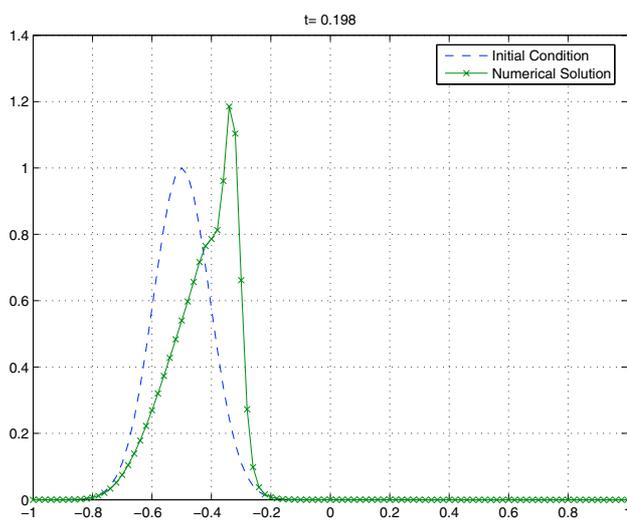
(a)  $N = 101, t = 0.144$ (b)  $N = 101, t = 0.198$ 

FIGURE 6. Solutions of the nonlinear problem.

on  $0 < x_1, x_2 < 1$  with boundary values

$$\begin{aligned} u(x_1, 0) &= \sin(\pi x_1) \\ u(0, x_2) &= \sin(\pi x_2) \\ u(1, 0) &= u(0, 1) = 0. \end{aligned}$$

The exact solution to this problem is given by

$$u(\mathbf{x}) = \frac{\sinh(\pi(1-x_1))}{\sinh \pi} \sin(\pi x_2) + \frac{\sinh(\pi(1-x_2))}{\sinh \pi} \sin(\pi x_1).$$

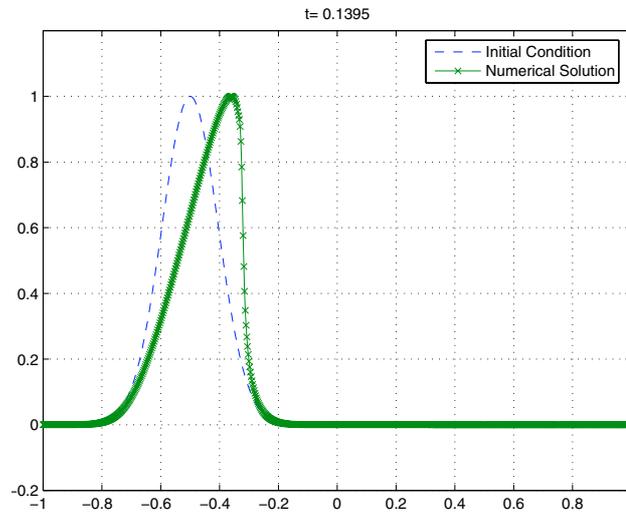
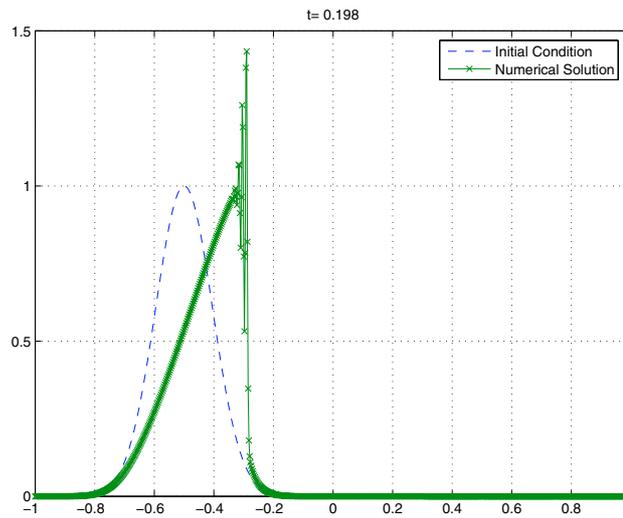
(a)  $N = 801, t = 0.144$ (b)  $N = 801, t = 0.198$ 

FIGURE 7. Solutions of the nonlinear problem.

We used a quadratic basis and  $N$  uniformly distributed points in the domain with a meshsize  $h$ . We have computed relative errors

$$\varepsilon := \left\| \frac{u^{appx} - u^{ex}}{u^{ex}} \right\|_{\infty},$$

where  $u^{appx}$  is the numerical solution, and  $u^{ex}$  the exact solution, and the *experimental order of convergence*

$$\text{EOC} := \frac{\log[\varepsilon_{h_1}/\varepsilon_{h_2}]}{\log[h_1/h_2]}.$$

The results are presented in Table 1. We obtained the third order of convergence.

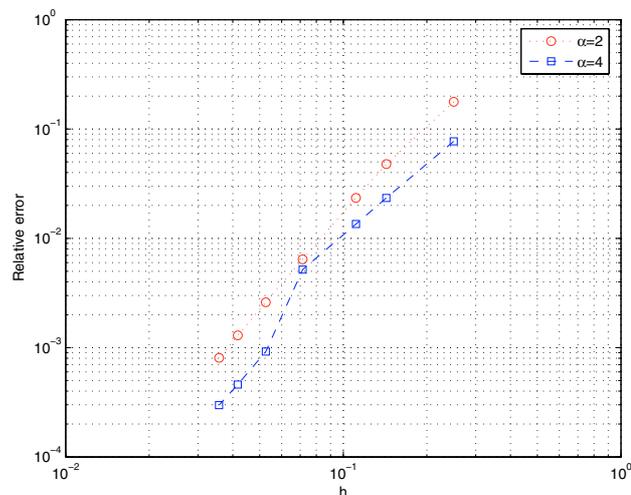
FIGURE 8. Relative error of the solution *vs.* meshsize  $h$ .

TABLE 1. Solution of boundary value problem.

$N$	$h$	$\varepsilon_{\alpha=2}$	$\text{EOC}_{\alpha=2}$	$\varepsilon_{\alpha=4}$	$\text{EOC}_{\alpha=4}$
25	0.2500 e+00	1.7725 e-1	—	7.7221 e-02	—
64	1.4286 e-01	4.7782 e-02	2.3425	2.3404 e-02	2.1332
100	1.1111 e-01	2.3430 e-02	2.8356	1.3525 e-02	2.1820
225	7.1429 e-02	6.4265 e-03	2.9278	5.1794 e-03	2.1724
400	5.2632 e-02	2.5979 e-03	2.9659	9.2167 e-04	5.6528
625	4.1667 e-02	1.2955 e-03	2.9784	4.5974 e-04	2.9772
841	3.5714 e-02	8.0568 e-04	3.0812	2.9784 e-04	2.8161

We refer the interested reader to the paper [5] for a detailed description of the solution of boundary value problems using interpolating moving least squares.

*Acknowledgements.* The authors gratefully acknowledge the financial support of the DFG for H.N. through the project SO 363/9-1.

We thank Matthias Kunle for his willingness to openly share with us his experience with MLS and IMLS methods for transport equations.

## REFERENCES

- [1] G.H. Golub and C.F. Van Loan, *Matrix Computations*. Johns Hopkins Univ. Press (1996).
- [2] M. Kunle, *Entwicklung und Untersuchung von Moving Least Square Verfahren zur numerischen Simulation hydrodynamischer Gleichungen*. Dissertation, Fakultät für Physik, Eberhard-Karls-Universität zu Tübingen (2001).
- [3] P. Lancaster and K. Šalkauskas, Surfaces generated by moving least square methods. *Math. Comp.* **37** (1981) 141–158.
- [4] P. Lancaster and K. Šalkauskas, *Curve and Surface Fitting - An Introduction*. Academic Press (1986).
- [5] H. Netuzhylov, Meshfree collocation solution of Boundary Value Problems via Interpolating Moving Least Squares. *Comm. Num. Meth. Engng.* **22** (2006) 893–899.
- [6] O. Nowak and T. Sonar, *Upwind and ENO strategies in Interpolating Moving Least Squares methods* (in preparation).
- [7] T. Sonar, Difference operators from interpolating moving least squares and their deviation from optimality. *ESAIM: M2AN* **39** (2005) 883–908.