Weierstraß-Institut
für Angewandte Analysis und Stochastik

im Forschungsverbund Berlin e.V.

Preprint ISSN 0946 – 8633

Regular polynomial interpolation and approximation of global solutions of linear partial differential equations

Jörg Kampen

submitted: June 5, 2007

Weierstrass Institute
for Applied Analsis and Stochastics
Mohrenstrasse 39
10117 Berlin, Germany
E-Mail: kampen@wias-berlin.de

No. 1230
Berlin 2007

2000 *Mathematics Subject Classification.* 65D05, 35G05.

Key words and phrases. Extended Newtonian interpolation, linear systems of partial differential equations, error estimates.

This work was completed with the support of DFG (Matheon).

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Mohrenstraße 39
10117 Berlin
Germany

Fax: + 49 30 2044975
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Abstract

We consider regular polynomial interpolation algorithms on recursively defined sets of interpolation points which approximate global solutions of arbitrary well-posed systems of linear partial differential equations. Convergence of the 'limit' of the recursively constructed family of polynomials to the solution and error estimates are obtained from a priori estimates for some standard classes of linear partial differential equations, i.e. elliptic and hyperbolic equations. Another variation of the algorithm allows to construct polynomial interpolations which preserve systems of linear partial differential equations at the interpolation points. We show how this can be applied in order to compute higher order terms of WKB-approximations of fundamental solutions of a large class of linear parabolic equations. The error estimates are sensitive to the regularity of the solution. Our method is compatible with recent developments for solution of higher dimensional partial differential equations, i.e. (adaptive) sparse grids, and weighted Monte-Carlo, and has obvious applications to mathematical finance and physics.

1 Introduction

This work shows how multivariate interpolation techniques can be combined with analytic information of linear partial differential equations (i.e. a priori estimates and/or WKB representations of solutions) in order to design efficient and accurate numerical schemes for solving (systems) of linear partial differential equations. These schemes are nothing but sequences of multivariate polynomials which are constructed recursively such that they solve a given linear system of partial differential equations on a finite discrete set of interpolation points. However, additional information is needed in order to ensure that the sequence of interpolation polynomials converges to a (or, if uniqueness is proved, the) global solution of a given linear system of partial differential equations. As we shall see, this information can be provided by a priori estimates which in turn lead us to error estimates in regular norms dependent on the regularity of the solution. We examine the situation in the case of linear elliptic equations with variable coefficients. Another possibility is that (more or less) explicit representations of solutions are known which lead to problems which are easier to solve. A prominent example is the WKB-expansion which was investigated in [6]. The recursive structure of WKB coefficient functions and the error analysis lead us to the problem of regular polynomial approximation. In this introductionary Section we our method on an abstract level.

1. 1. Regular polynomial interpolation

Since we are interested in the relationship between multivariate polynomial interpolation and approximation of solutions of partial differential equations, our focus will be on multivariate polynomial interpolation. However, in order to make basic ideas more accessible we shall describe algorithms in the univariate case first and then generalize to the multivariate case. It is well known that polynomial interpolation in the multivariate case is quite different from the univariate case in general. However, in our approach which aims at solving linear systems of partial differential equations or aims at supplementing certain strategies of solving partial differential equations many features are already present in the univariate framework. In order to avoid misunderstandings, we dwell a little on this point. Classically, the problem of multivariate interpolation can be stated as follows (cf. [11]):

Given a set of interpolation points $\Theta = \{x_1, \cdots, x_N\}$ and an N-dimensional

space P_Θ of polynomials find, for given values $y_1, \cdots, y_N$, a unique polynomial $f \in P$ such that

$$f(x_j) = y_j,\ j \in 1, \cdots, N. \tag{1.1}$$

In this form it turns out that there is an intricate relation between sets of interpolation points and interpolation spaces that must be satisfied in order that the problem can be considered to be well-posed. Either we have to make some restrictions concerning the set of interpolation points Θ (cf. [11]) or we consider Θ to be fixed and consider the problem of constructing the polynomial space P_θ (cf.[1]). This amounts to a construction of the map

$$\Theta \rightarrow P_\Theta \tag{1.2}$$

with additional constraints such as minimality of degree (cf. [11, 1]) or monotonicity (cf. [1]). In this paper we are interested in interpolation algorithms with the following features

- there are no essential restriction on the discrete set Θ of interpolation points except that $\Theta \subset D$, where D is the domain of the function to be interpolated.
- the map $\Theta \rightarrow P_\Theta$ is monoton (indeed our basic algorithm is an extension of multivariate versions of Newton's interpolation algorithm).
- the algorithm can be extended to vector valued interpolation functions $g : D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^k$ and if g satisfies a system of linear partial differential equations, then the interpolation polynomial p solves the same system of linear partial differential equations on the given set Θ of interpolation points.
- the algorithm is numerically stable and practical with respect to the problem that the interpolation function f and arbitrary set of partial derivatives of f are to be interpolated simultaneously. For the application of higher order approximation of the fundamental solution of linear parabolic equations we comute accurate approximations of derivatives of smooth functions up to order 10 in order to obtain an approxmation of order 5 of the WKB-expansion of the fundamental solution.
- the algorithm can be refined in order to solve well-posed linear systems of partial differential equations directly.
- the algorithm can be combined with collocation methods in an efficient way; it can be partially parallelized.
- the algorithm allows for error estimates which depend on the regularity of the solution such that the algorithm is compatible with methods for higher dimensional problems of linear systems of partial differential equations such as sparse grids, adaptive sparse grids, and weighted Monte-Carlo.

First we consider the problem of polynomial approximation p of a regular (i.e smooth or finitely many times differentiable) function

$$f : D \subseteq \mathbb{R}^n \rightarrow \mathbb{R} \tag{1.3}$$

defined on discrete subset of $\Theta \subset D$ where for m given linear partial differential operators

$$L_i = \sum_{|\alpha| \leq \beta_i} a^i_\alpha(x) \partial_\alpha, \tag{1.4}$$

we require that

$$L_i f(x_j) = L_i p(x_j) \text{ for } 1 \leq i \leq m \tag{1.5}$$

for some finite set of points $x_j \in \Theta \subset D$. As indicated above we shall allow that the interpolation set Θ can be constructed recursively (and, hence, extended arbitrarily within the domain of the interpolation function). Investigations of specific instances of this problem can be found in the literature on polynomial interpolation (cf. the survey paper of [10] for the development up to the year 2001). Note that other algorithms of natural interpolation of C^k-functions have been proposed (cf.[5] for hints at the history and further references).

The paper is organized as follows. In Section 1.2 we introduce the partial differential equations for which we seek global regular interpolation polynomials of their global solutions. All basic types of partial differential equations, i.e. elliptic equations, parabolic equations, and hyperbolic equations are considered. While the basic algorithm is quite similar for each type of partial differential equation, we shall see, however, that the convergence of the scheme of recursively defined interpolation polynomials depends on very different a priori estimates for different type of equations. In case of second order elliptic equations classical Schauder boundary estimates can be used, while in the case of hyperbolic equations energy estimates are considered. In the case of parabolic equations we refer back to Safanov-Krylov estimates considered in the context of the truncation error analysis of WKB-expansions. In Section 2.1 we introduce first an extension of Newton's polynomial algorithm which interpolates a given function and its derivatives up to some given order k simultaneously. Section 2.2. describes a variation of this algorithm which interpolates a given function such that a given set of partial differential equations is preserved. Section 3 discusses the extension to the multvariate case. In Section 4 we refine the algorithm and construct polynomials which satisfy a given linear (i.g. partial) differential equation on a given set of interpolation points, i.e. there is no given function to be interpolated. In Section 5 we consider refinements which show how polynomials constructed on disjoint sets of interpolation points can be synthesized in order to get one polynomial which interpolates on the union of sets of interpolation points. Naturally, parallelization is consideredin this context. In Section 6 we show how a priori estimates of elliptic equations (standard Schauder boundary estimates) and hyperbolic equations (energy estimates) lead to convergent schemes implied by error estimates. Section 7 discusses a special use of regular polynomial interpolation for parabolic equations where the global solution is given in the form of a WKB- expansion. Section 8 provides a numerical example of global regular polynomial interpolation of a locally analytic function up to the third derivative. In Section 9 we provide a summary and give an outlook on current research and research in the near future. Before we start with the description of the algorithm, we state the typical linear partial differential equations and indicate the different types of approximations and error estimates which we aim at.

1. 2. Regular interpolation and partial differential equations

We consider the three standard types of linear partial differential equations, namely elliptic equations, parabolic equations, and hyperbolic equations, and exemplify different types of application and extension.

- The most popular examples of elliptic partial differential equations are of the second order form, i.e.

$$\sum_{j,k}^{n} a_{jk}(x)\frac{\partial^2 u}{\partial x_j \partial x_k} + \sum_{l} b_l(x)\frac{\partial u}{\partial x_l} + c(x)u = f(x), \tag{1.6}$$

  to be solved on a domain $\Omega \subseteq \mathbb{R}^n$ with the boundary condition

$$u\Big|_{\partial\Omega} = g \tag{1.7}$$

  for some function $f : \partial\Omega \to \mathbb{R}$ which is usually assumed to be Lipschitz continuous at least. Here, a_{jk} are (at least) measurable coefficient functions satisfying for some constant c, and ellipticity means that

$$\sum_{jk} a_{jk}(x)\xi_i\xi_j \geq c > 0 \text{ (uniformly in } x). \tag{1.8}$$

  We construct an extension of the polynomial interpolation algorithm which produces a multivariate polynomial solving this elliptic equation on an arbitrary grid of interpolation points. In order to obtain error estimates b standard boundary Schauder estimates in this paper we shall make some regularity assumptions. We derive convergence of the family of multivariate polynomials constructed by our our interpolation scheme to the global solution of the linear elliptic equation on a bounded domain and we derive error estimates from a priori estimates.

- Parabolic equations of the form

$$\frac{\partial u}{\partial t} - Lu = 0, \tag{1.9}$$

  on $D := \Omega \times (0, T)$, ($\Omega \subseteq \mathbb{R}^n$, with

$$u(0, x) = \delta_y(x) := \delta(x - y),\ y \in \mathbb{R}^n, \tag{1.10}$$

  where δ is the Dirac delta distribution, and where

$$Lu \equiv \frac{1}{2}\sum_{ij} a_{ij}(x)\frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i} b_i(x)\frac{\partial u}{\partial x_i} \tag{1.11}$$

  is an elliptic operator. The solution of this equation is called fundamental solution, because solutions of standard parabolic initial-value boundary problems can be represented by convolution integrals of data functions with the fundamental solution. The standard assumptions for such a fundamental solution to exist are

(A) The operator L is uniformly parabolic in $\mathbb{R}^n$, i.e. there exists $0 < \lambda < \Lambda < \infty$ such that for all $\xi \in \mathbb{R}^n \setminus \{0\}$

$$0 < \lambda|\xi|^2 \leq \sum_{i,j=1}^{n} a_{ij}(x)\xi_i\xi_j \leq \Lambda|\xi|^2.$$

(B) The coefficients of L are bounded functions in $\mathbb{R}^n$ which are uniformly Hölder continuous of exponent α ($\alpha \in (0,1)$).

If some regularity assumptions on the coefficients hold in addition, then it can be shown that the fundamental solution p is of the form

$$p(t,x,y) = \frac{1}{\sqrt{2\pi t}^n} \exp\left(-\frac{d^2(x,y)}{2t} + \sum_{k\geq 0} c_k(x,y)t^k\right), \tag{1.12}$$

with some regular coefficient functions d^2 and c_k. We shall show how our regular polynomial interpolation algorithm can be used to compute the fundamental solution in terms of this representation.

Remark 1.1 *The algorithm designed in the case of elliptic equations can be applied to the parabolic case directly, of course. However, it turns out that the convergence is better if the special representation* (1.12) *is used.*

- As an example of a hyperbolic equation we consider an equation of the form

$$Lu = f \text{ in } \Omega, \tag{1.13}$$

where

$$Lu \equiv \sum_{ij} h_{ij} \frac{\partial u}{\partial x_i \partial x_j} + \sum_i \frac{\partial}{\partial x_j} + c(x)u \tag{1.14}$$

and (h_{ij}) is a symmetric matrix of signature $(n,1)$, if $\dim\Omega = n+1$. We assume that some $O \subset \Omega$ is bounded by two spacelike surfaces Σ_i and Σ_e and swept out by a family of spacelike surfaces $\Sigma_e(s)$. We assume the initial conditions

$$u = g \text{ and } du = \omega \tag{1.15}$$

where g is a function on Ω and Ω is a 1-form.

2 Interpolation algorithm (univariate case)

We start with the description of the algorithm which produces polynomials which satisfy some given requirements on interpolation points. Our starting point is an extension of Newton's polynomial interpolation method such that the interpolation polynomial and its derivatives up to a given order k (an integer) equal a given function and its derivatives up to order k at the interpolation points. For simplicity of representation and since the essential features of the algorithm can be demonstrated for one dimensional functions, we describe our ideas first in the univariate case and then generalize to the multivariate case in the next section.

2. 1. Extension of Newton's method

Let us recall the Newtonian interpolation for an univariate function

$$f : [a, b] \subset \mathbb{R} \to \mathbb{R}. \tag{2.1}$$

Given a discrete set of interpolation points $D = \{x_0, x_1 \cdots , x_N\} \subset [a, b]$ we want to construct a polynomial

$$\begin{aligned} &p : [a, b] \subset \mathbb{R} \to \mathbb{R} \text{ such that} \\ &f(x_i) = p(x_i) \text{ for all } x_i \in D. \end{aligned} \tag{2.2}$$

The idea of the basic Newton interpolation algorithm is that instead of looking for some polynomial of form $\sum_{i=1}^{N} b_i x^i$ for some constants b_i we may write

$$\sum_{l=0}^{N} a_l \Phi_l(x) \tag{2.3}$$

with

$$\Phi_0(x) = 1 \text{ and } \Phi_l(x) = \Pi_{i=0}^{l}(x - x_i) \text{ for } l \geq 1. \tag{2.4}$$

In order to determine $a_0, \cdots a_N$ we then may solve the system

$$R_0 a := \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & \phi_1(x_1) & 0 & \cdots & 0 \\ 1 & \phi_1(x_2) & \phi_2(x_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & \phi_1(x_N) & \phi_2(x_N) & \cdots & \phi_N(x_N) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix} \tag{2.5}$$

This leads to an L^2-approximation of the function f similar to the Gaussian algorithm. Note however, that the matrix R_0 is a lower diagonal. Hence the linear system can be solved easily. Moreover the matrix condition number is much better than that of the Vandermonde matrix used in the classical Gaussian interpolation. We extend this idea to a C^k-norm interpolation, i.e. we design an algorithm that approximates f up to the k-th derivative, i.e. we construct a polynomial

$$\begin{aligned} &q : [a, b] \subset \mathbb{R} \to \mathbb{R} \text{ such that} \\ &f^{(l)}(x_i) = q^{(l)}(x_i) \text{ for all } x_i \in D \text{ and all } l \leq k, \end{aligned} \tag{2.6}$$

where for a function $g : [a, b] \subset \mathbb{R} \to \mathbb{R}$ $g^{(l)}$ denotes the derivative of order l while $g = g^0$. We consider the polynomial

$$\sum_{m=0}^{(N+1)(k+1)-1} a_m \Phi_{m,k}(x) \tag{2.7}$$

where

$$\Phi_{m,k}(x) = (x - x_{m \text{ div}(k+1)})^{m \text{ mod}(k+1)} \Pi_{l=0}^{m\text{div}(k+1)-1}(x - x_l)^{k+1}, \tag{2.8}$$

where, by convention, we understand

$$\Pi_{l=0}^{-1}(x - x_l)^{k+1} := 1. \tag{2.9}$$

For simplicity of notation we sometimes use the abbreviations

$$p(m) = m\mathrm{div}(k+1) \text{ and } q(m) = m\mathrm{mod}(k+1). \tag{2.10}$$

Next we define

$$\Phi^{(l)}_{m,k}(x) := \frac{d}{dx^l}\Phi_{m,k}(x), \tag{2.11}$$

and for each $k \geq 1$ the linear system

$$R_k \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{(k+1)(N+1)-1} \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f'(x_0) \\ \vdots \\ f^{(k)}(x_0) \\ f(x_1) \\ \vdots \\ f^{(k)}(x_{(k+1)(N+1)-1}) \end{bmatrix} \tag{2.12}$$

where R_k is a $(N+1)(k+1) \times (N+1)(k+1)$-matrix determined by $(k+1) \times (k+1)$ matrices A^{lm}_k as follows:

$$R_k := \begin{bmatrix} A^{00}_k & Z_k & Z_k & Z_k & \cdots & Z_k \\ A^{10}_k & A^{11}_k & Z_k & Z_k & \cdots & Z_k \\ A^{20}_k & A^{21}_k & A^{31}_k & Z_k & \cdots & Z_k \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A^{N0}_k & A^{N1}_k & A^{N2}_k & A^{N3}_k & \cdots & A^{NN}_k \end{bmatrix}, \tag{2.13}$$

where Z_k is the $(k+1) \times (k+1)$ matrix with 0 entries, and

$$A^{ij}_k = A^i_k(x_j) \tag{2.14}$$

with

$$A^{ij}_k := \begin{bmatrix} \Phi_{(k+1)p(i),k}(x_j) & \Phi_{(k+1)p(i)+1,k}(x_j) & \Phi_{(k+1)p(i)+2,k}(x_j) & \cdots & \Phi_{(k+1)p(i)+k,k}(x_j) \\ \Phi^{(1)}_{(k+1)p(i),k}(x_j) & \Phi^{(1)}_{(k+1)p(i)+1,k}(x_j) & \Phi^{(1)}_{(k+1)p(i)+2,k}(x_j) & \cdots & \Phi^{(1)}_{(k+1)p(i)+k,k}(x_j) \\ \Phi^{(2)}_{(k+1)p(i),k}(x_j) & \Phi^{(2)}_{(k+1)p(i)+1,k}(x_j) & \Phi^{(2)}_{(k+1)p(i)+2,k}(x_j) & \cdots & \Phi^{(2)}_{(k+1)p(i)+k,k}(x_j) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \Phi^{(k)}_{(k+1)p(i),k}(x_j) & \Phi^{(k)}_{(k+1)p(i)+1,k}(x_j) & \Phi^{(k)}_{(k+1)p(i)+2,k}(x_j) & \cdots & \Phi^{(k)}_{(k+1)p(i)+k,k}(x_j) \end{bmatrix}. \tag{2.15}$$

Note that

$$A^{00}_k := \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & k! \end{bmatrix}. \tag{2.16}$$

This leads to a system which can be solved row by row. It is therefore very easy to implement and numerically well-conditioned.

Remark 2.1 *In order to avoid large entries in the matrices A^{lm}_k one may consider basis functions of form $\frac{1}{l!}\Phi^{(l)}_{(k+1)p(i),k}$, but we do not deal with the peculiar niceties of computation here.*

2. 2. Interpolation preserving linear systems of differential equations

The preceding algorithm can be adapted it in order to construct a polynomial approximation p of f where the k differential operators

$$L_i f(x) = \sum_{j\leq q_i} a^i_j(x)\frac{d}{dx^j}f(x), i = 1,\cdots,k \tag{2.17}$$

are preserved on a discrete set of points $\Theta = \{x_0,\cdots,x_N\}$ in the sense that

$$L_i f(x_j) = L_i p(x_j) \text{ for } x_j \in \Theta. \tag{2.18}$$

At this point the linear system of the operators $\{L_i|1\leq i\leq k\}$ is quite arbitrary; we just assume that the operators are defined pointwise, i.e. $x\rightarrow a^i_j(x)$ are classical functions which can be evaluated pointwise (at least on the set of interpolation points). Note that we do not ask about convergence of a family of interpolation polynomials to at this point. There are several possibilities to extend our preceding algorithm. One is the following. Let

$$Q_i := \left\{j|a^i_j\neq 0\right\} \tag{2.19}$$

and define

$$L^m_i = \sum_{j\in Q_i, j\leq m} a^1_{i_j}(x)\frac{d^{i_j}}{dx^{i_j}}. \tag{2.20}$$

We start with

$$Q_1 = \{i_{11},\cdots,i_{1r_1}\}, \tag{2.21}$$

and assume that

$$i_{11} < \cdots < i_{1r_1} \tag{2.22}$$

We consider first the interpolation point x_0 and start with the following ansatz for the interpolation polynomial

$$p_{10}(x) = \sum_{i_{1j}\in Q_1} b^{10}_{i_j}(x-x_0)^{i_{1j}}. \tag{2.23}$$

We assume $f(x_0) = p_{10}(x_0) = 0$ w.l.o.g. ; we shall see later how we interpolate values of f different from zero at the other interpolation points $x_1,\cdots,x_N$. First we apply the operator

$$L^{i_1}_1 \equiv a^1_{i_1}(x)\frac{d^{i_1}}{dx^{i_1}} \tag{2.24}$$

to f and p_{10} at x_0. This leads to

$$i_1! b^{10}_{i_1} = a^1_{i_1}(x_0)\frac{d^{i_1}f}{dx^{i_1}}(x_0) \quad\Rightarrow\quad b^{10}_{i_1} = \frac{1}{i_1!}a^1_{i_1}\frac{d^{i_j}f}{dx^{i_1}}(x_0) \tag{2.25}$$

Inductively we assume that the coefficients $b^{10}_{i_j}$ have been defined up to the index i_m for some $m < r_1$ and that the operator $L^{i_m}_1$ has been defined accordingly. We apply the operator

$$L^{i_{m+1}}_1 \equiv L^{i_m}_1 + a^1_{i_{m+1}}(x)\frac{d^{i_{m+1}}}{dx^{i_{m+1}}} \tag{2.26}$$

to f and p_{10} at x_0. For an integer s with $m+1 \leq s \leq r_1$ define

$$p_{10}^s(x) = \sum_{j=1}^{s} b_{i_j}^{10}(x-x_0)^{i_j}. \tag{2.27}$$

Then we have

$$\begin{aligned} &L_1^{i_{m+1}} p_{10}(x_0) = L_1^{i_m} p_{10}(x_0) + a_{i_{m+1}}^1(x_0) \tfrac{d^{i_{m+1}}}{dx^{i_{m+1}}} p_{10}(x_0) = \\ &L_1^{i_m} p_{10}^{i_m}(x_0) + i_{m+1}! a_{i_{m+1}}^1(x_0) b_{i_{m+1}}^{10} = L_1^{i_{m+1}} f(x_0). \end{aligned} \tag{2.28}$$

This gives $b_{i_{m+1}}^{10}$. Next inductively assume that an interpolation polynomial p_{1k} has been constructed which interpolates f on the set of interpolation points $\{x_0, \cdots, x_k\}$ for some positive integer k with $k < N$ subject to the condition

$$L_1 f(x_i) = p_{1k}(x_i) \text{ for } 1 \leq i \leq k. \tag{2.29}$$

First we extend that polynomial in order to interpolate f at the point x_{k+1}. We consider the ansatz

$$p_{1(k+1)}^0(x) = p_{1k}(x) + b_0^{1(k+1)} \Pi_{l=0}^k (x-x_l)^{q_1}. \tag{2.30}$$

We then get $b_0^{1(k+1)}$ from the equation

$$p_{1(k+1)}^0(x_{k+1}) = f(x_{k+1}). \tag{2.31}$$

The ansatz for $p_{1(k+1)}$ (i.e. the interpolation polynomial which preserves $L_1 f$ on the set of interpolation points $\{x_1, \cdots, x_{k+1}\}$) is

$$p_{1(k+1)}(x) = p_{1(k+1)}^0(x) + \sum_{i_j \in Q_1} b_{i_j}^{1(k+1)} (x-x_{k+1})^{i_j} \Pi_{l=0}^k (x-x_l)^{q_1+1} \tag{2.32}$$

and the determination of coefficient constants $b_{i_j}^{1(k+1)}$ is similar to the procedure for the interpolation point x_0 described above. Proceeding inductively, we are lead to the polynomial p_1 which interpolates f at the interpolation points of $\Theta = \{x_0, \cdots, x_N\}$ such that

$$L_1 p_1(x_j) = L_1 f(x_j) \text{ for all } x_j \in \Theta. \tag{2.33}$$

Finally assuming that for some integer $s < k$ the polynomial p_s satisfies the condition that

$$\begin{aligned} &p_s(x_j) = f(x_j) \text{ for } x_j \in \Theta \\ &L_i p_s(x_j) = L_i f(x_j) \text{ for } x_j \in \Theta \text{ and } i \leq s, \end{aligned} \tag{2.34}$$

it is clear that we only need to consider the reduced operator

$$L_{s+1} \equiv \sum_{i_j j \in Q_{s+1} \setminus \cup_{i=1}^s Q_i} a_{i_j}^{s+1}(x) \frac{d^{i_j}}{dx^{i_j}}. \tag{2.35}$$

and proceed analogously.

3 Extension to the multivariate case

Next we consider generalizations to the multivariate case. There are several possibilities but the most simple seems to be the following. First we formulate the problem in a way that will turn out to be useful in the context of polynomial interpolation of global solutions of linear systems of partial differential equations. In its most simple form it is a form of multivariate Newton interpolation: given a function

$$f : S \subset \mathbb{R}^n \to \mathbb{R} \tag{3.1}$$

we want to construct a polynomial

$$\begin{aligned} &p : S \subset \mathbb{R}^n \to \mathbb{R} \text{ such that} \\ &f(x_i) = p(x_i) \text{ for all } x_i \in D \subseteq S, \end{aligned} \tag{3.2}$$

where $D = \{x_0, x_1, \cdots, x_n\}$ is some discrete sets of points in $\mathbb{R}^n$ whose coordinates will be denoted by superscript indices as x_i^j, $j = 1, \cdots, n$. This is done then by recursive definition of polynomials $p_0, p_1, \cdots$. First, define

$$p_0(x) \equiv f(x_0). \tag{3.3}$$

Next, ansatz and equation

$$p_1(x) \equiv f(x_0) + a_1 \Pi_{i=1}^n (x^i - x_0^i) = f(x_1) \tag{3.4}$$

leads to the determination of p_1 by

$$a_1 = \tfrac{f(x_1) - f(x_0)}{\Pi_{i=1}^n (x^i - x_0^i)} \tag{3.5}$$

Next assume that $p_0, p_1, \cdots, p_q$ have been defined. Then ansatz and equation

$$p_{q+1}(x_{q+1}) \equiv p(x_{q+1}) + a_{q+1} \Pi_{k=0}^q \Pi_{i=1}^n (x^i - x_k^i) = f(x_{q+1}) \tag{3.6}$$

leads to the determination of p_{q+1} by

$$a_{q+1} = \tfrac{f(x_{q+1}) - p_q(x_{q+1})}{\Pi_{k=0}^q \Pi_{i=1}^n (x^i - x_k^i)} \tag{3.7}$$

3. 1. Extension of Newton's method

Next we extend a multivariate version of Newton's method, i.e. we design an algorithm that approximates f up to the β-th derivative ($\beta = (\beta_1, \cdots, \beta_n)$ being some multiindex) where we construct a polynomial

$$\begin{aligned} &q : S \subset \mathbb{R} \to \mathbb{R} \text{ such that} \\ &\tfrac{\partial f}{\partial x^\gamma}(x_i) = \tfrac{\partial q}{\partial x^\gamma}(x_i) \text{ for all } x_i \in D \subseteq S \text{ and all } \gamma \leq \beta. \end{aligned} \tag{3.8}$$

where β is given (i.e. the multivariate substitute for k in the univariate case described above), and ordering is in the following sense:

Definition 3.1 *Let x^α and x^β be monomials in $\mathbb{R}[x_1, \cdots, x_n]$. We say that $x^\alpha > x^\beta$ (lexicographical order) if $\sum_i \alpha^i > \sum_i \beta^i$ or $\sum_i \alpha^i = \sum_i \beta^i$, and in the difference $\alpha - \beta \in \mathbb{Z}^n$ the left-most non zero entity is positive.*

Now, let $\alpha_0, \alpha_1, \cdots, \alpha_m, \cdots$ an enumeration of multiindices with respect to this ordering. We define a sequence of polynomials $p_{\alpha_0}, p_{\alpha_1}, \cdots, p_{\alpha_m}, \cdots$ recursively. First, let

$$p_{\alpha_0}(x) = a_{\alpha_0} + \sum_{\gamma \leq \beta} a_{\alpha_0 \gamma} \Pi_{i=1}^n (x^i - x^i_{\alpha_0})^{\gamma_i}. \tag{3.9}$$

If $p_{\alpha_0}, \cdots, p_{\alpha_{m-1}}$ have been defined, then we define

$$\begin{aligned} &p_{\alpha_m}(x) = p_{\alpha_{m-1}}(x) + \\ &\textstyle\sum_{\gamma \leq \beta} a_{\alpha_{m-1}\gamma} \Pi_{i=1}^n (x^i - x^i_{\alpha_{m-1}})^{\gamma^i} \Pi_{j=0}^{m-1} \Pi_{i=1}^n (x^i - x^i_{\alpha_j})^{\beta^i + 1}. \end{aligned} \tag{3.10}$$

This leads to a linear system to be solved for a vector $(a_{\alpha_0}, \cdots, a_{\alpha_N \beta})$ of length $(N + 1)\left(\sum_i \beta^i + 1\right)$

$$R_\beta \begin{bmatrix} a_{\alpha_0} \\ \vdots \\ a_{\alpha_0 \beta} \\ a_{\alpha_1} \\ \vdots \\ a_{\alpha_N \beta} \end{bmatrix} = \begin{bmatrix} f(x_{\alpha_0}) \\ \vdots \\ f^{(\beta)}(x_{\alpha_0}) \\ f(x_{\alpha_1}) \\ \vdots \\ f^{(\beta)}(x_{\alpha_N}) \end{bmatrix} \tag{3.11}$$

with

$$R_\beta := \begin{bmatrix} A_\beta^{00} & Z_\beta & Z_\beta & Z_\beta & \cdots & Z_\beta \\ A_\beta^{10} & A_\beta^{11} & Z_\beta & Z_\beta & \cdots & Z_\beta \\ A_\beta^{20} & A_\beta^{21} & A_\beta^{31} & Z_\beta & \cdots & Z_\beta \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_\beta^{N0} & A_\beta^{N1} & A_\beta^{N2} & A_\beta^{N3} & \cdots & A_\beta^{NN} \end{bmatrix} \tag{3.12}$$

We abbreviate $\sum \beta = \sum_i (\beta^i + 1)$ and defining $p(m) = m \div \sum \beta$ we have

$$A_k^{ij} := \begin{bmatrix} \Phi_{\sum \beta p(i), \beta}(x_j) & \Phi_{\sum \beta p(i) + \beta_1, \beta}(x_j) & \Phi_{\sum \beta p(i) + \beta_2, \beta}(x_j) & \cdots & \Phi_{\sum \beta p(i) + \beta, \beta}(x_j) \\ \Phi^{(\beta_1)}_{\sum \beta p(i), \beta}(x_j) & \Phi^{(1)}_{\sum \beta p(i) + \beta_1, \beta}(x_j) & \Phi^{(\beta_1)}_{(k+1) p(i) + \beta_2, \beta}(x_j) & \cdots & \Phi^{(\beta_1)}_{\sum \beta p(i) + \beta, \beta}(x_j) \\ \Phi^{(\beta_2)}_{\sum \beta p(i), \beta}(x_j) & \Phi^{(\beta_2)}_{\sum \beta p(i) + \beta_1, \beta}(x_j) & \Phi^{(\beta_2)}_{\sum \beta p(i) + \beta_2, k}(x_j) & \cdots & \Phi^{(\beta_2)}_{\sum \beta p(i) + \beta, \beta}(x_j) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \Phi^{(\beta)}_{\sum \beta p(i), \beta}(x_j) & \Phi^{(\beta)}_{\sum \beta p(i) + \beta_1, \beta}(x_j) & \Phi^{(\beta)}_{\sum \beta p(i) + \beta_2, \beta}(x_j) & \cdots & \Phi^{(\beta)}_{\sum \beta p(i) + \beta, \beta}(x_j) \end{bmatrix}. \tag{3.13}$$

3. 2. Multivariate Interpolation preserving linear systems of PDEs

Similar to the univariate case one can adapt the preceding algorithm to the interpolation of multivariate functions, i.e. interpolate f by a polynomial p such that $f = p$, and

$$L_i f(x) = L_i p(x) \text{ for } x \in \Theta. \tag{3.14}$$

where $\Theta = \{x_0, \cdots, x_N\}$ is the set of interpolation points, and the partial differential operators are defined by

$$L_i f(x) = \sum_{|\alpha| \leq q_i} a^i_\alpha(x) \partial^\alpha f(x), \ = 1, \cdots, k. \tag{3.15}$$

The procedure is analogue to that described in Section 2.2. (cf.also [7]).

4 Approximation of global solutions of linear partial differential equations

We refine the algorithm further in order to solve linear partial differential equations globally. In this case the function u to be approximated is not known. In this section we shall simply describe an algorithm which constructs a polynomial which satifies a linear system of partial differential equations on an arbitrary set of interpolation points. It is not clear, however, if this polynomial approximation converges to the solution of the system. To ensure that and in order to estimate the rate of convergence we shall need the a priori estimates and regularity results. Note however, that the regularity constraints on the solution maybe low for problems on compact domains as any continuous solution functions u can be approximated by a families of polynomial functions approximating u. Therefore, principally, the families of polynomial functions constructed here may approximate continuous global solutions in viscosity sense. An investigation of this problem will be considered elsewhere in a more general framework where we include some class of nonlinear problems. In order to make the basic ideas transparent we consider first scalar linear problems. We exemplify our algorithm first in the case of dimension $n = 1$ and then generalize to the case $n > 2$. What we have in mind here are elliptic equations but we need the ellipticity condition only when we wan to prove that the family of polynomials construxted converges to the global solutions. Then we exemplify our method in the case of a typical linear first order system. It is then clear how to generalize to systems of linear equations of any order.

4. 1. The case scalar second order equations of dimension $n = 1$

We consider the simple boundary value problem

$$L_1 u \equiv a(x)\frac{d^2u}{dx^2} + b(x)\frac{du}{dx} + c(x)u = f(x) \text{ on } (d, e) \subset \mathbb{R}, \tag{4.1}$$

with the boundary condition $u(d) = c_d$ and $u(e) = c_e$ (actually an ordinary differential equation). If $a(x) \geq \lambda > 0$ for all $x \in \mathbb{R}$, then we have an elliptic operator, but this is not an assumption which we need to construct an univariate polynomial which satisfies the boundary problem on the interpolation points.

We start with the point d. We construct a list of polynomial $q_m, m \geq 0$. We define the q_m in substeps. Let $p_0 = a_0$. In order that p_0 satisfies the boundary condition at $x = d$ we impose

$$p_0 = a_0 = c_d \tag{4.2}$$

Next we define

$$p_1(x) = a_0 + a_1(x - d) \tag{4.3}$$

In order to satisfy the second boundary condition we get

$$p_1(e) = a_0 + a_1(e-d) = c_d + a_1(e-d) = c_e \Rightarrow a_1 = \frac{c_e - c_d}{e-d}. \tag{4.4}$$

It is clear that p_1 preserves the boundary conditions, i.e. $p(d) = u(d) = c_d$ and $p(e) = u(e) = c_e$. Next let x_0 be the first interpolation point (any point in the interval (d,e). We want to ensure that

$$a(x_0)\frac{d^2p}{dx^2}(x_0) + b(x_0)\frac{dp}{dx}(x_0) + c(x_0)p(x_0) = f(x_0). \tag{4.5}$$

In order to ensure this, we define a polynomial which is an extension of p_0 in three steps. First, define

$$p_2(x) = a_0 + a_1(x-d) + a_4(x-x_0)^2(x-d)(x-e) \tag{4.6}$$

Plugging in and evaluating at $x = x_0$ we get

$$a(x_0)2a_4(x_0-d)(x_0-e) + b(x_0)a_1 + c(x_0)(a_0 + a_1(x_0-d)) = f(x_0) \tag{4.7}$$

Since a_0, a_1 are known we get (recall that $x_0 \neq d$ and $x_0 \neq e$)

$$a_4 = \frac{f(x_0) - c(x_0)(a_0 + a_1(x_0-d)) - b(x_0)a_1}{2a(x_0)(x_0-d)(x_0-e)}. \tag{4.8}$$

Next define

$$p_3(x) = p_2(x) + a_3(x-x_0)(x-x_d)(x-x_e). \tag{4.9}$$

Plugging in and evaluating at $x = x_0$ we get (assuming that)

$$\begin{aligned} L_1p_3(x_0) = \quad & L_1p_2(x_0) + a(x_0)a_3(2(x_0-d) \\ & +2(x_0-x_e)) + b(x_0)a_3(x_0-d)(x_0-e) = f(x_0). \end{aligned} \tag{4.10}$$

Hence, (provided that $x_0 \neq d$ and $x_0 \neq e$),

$$a_3 = \frac{f(x_0) - L_1p_2(x_0)}{a(x_0)(2(x_0-d) + 2(x_0-e)) + b(x_0)(x_0-d)(x_0-e)} \tag{4.11}$$

Finally, finishing the first inductive step of recursive definition of the polynomial family $(q_m)_{m\in\mathbb{N}}$

$$p_4(x) = p_3(x) + a_2(x-d)(x-e). \tag{4.12}$$

Plugging in and evaluating at $x = x_0$ we get (assuming that)

$$L_1p_4(x_0) = L_1p_3(x_0) + 2a(x_0)a_2 + b(x_0)((x_0-d) + (x_0-e)) = f(x_0). \tag{4.13}$$

Hence, (recall again that $x_0 \neq d$ and $x_0 \neq e$),

$$a_2 = \frac{f(x_0) - L_1p_3(x_0) - b(x_0)((x_0-d) + (x_0-e))}{2a(x_0)((x_0-d) + 2(x_0-e))} \tag{4.14}$$

Now we can define

$$q_1(x) = p_4(x) \tag{4.15}$$

Next assume that the polynomials $q_1, \cdots, q_k$ have been defined. This means that we have computed the polynomial coefficients $a_0, a_1, \cdots, a_{2+3k}$. Then q_{k+1} is defined via

$$q_{k+1}(x) = q_k(x) + (x-d)^3(x-e)^3\Pi_{l=0}^k (x-x_l)^3 z_k(x), \tag{4.16}$$

where z_k is a polynomial function which will be defined in three substeps. First, let

$$q_{k+1,1}(x) = q_k(x) + a_{2+3(k+1)}(x-x_{k+1})^2(x-d)^3(x-e)^3\Pi_{l=0}^k (x-x_l)^3 \tag{4.17}$$

Plugging in leads to

$$\begin{aligned} L_1 q_{k+1,1}(x_{k+1}) = \quad & L_1 q_k(x_{k+1}) + a(x_{k+1})2a_{2+3(k+1)}(x_{k+1}-d)^3 \times \\ & (x_{k+1}-e)^3\Pi_{l=0}^k (x_{k+1}-x_l)^3 = f(x_{k+1}). \end{aligned} \tag{4.18}$$

Hence,

$$a_{2+3(k+1)} = \frac{f(x_{k+1}) - L_1 q_k(x_{k+1})}{a(x_{k+1})2(x_{k+1}-d)^3(x_{k+1}-e)^3\Pi_{l=0}^k (x_{k+1}-x_l)^3} \tag{4.19}$$

Next, let

$$q_{k+1,2}(x) = q_{k+1,1}(x) + a_{2+3k+2}(x-x_{k+1})(x-d)^3(x-e)^3\Pi_{l=0}^k (x-x_l)^3 \tag{4.20}$$

We define

$$R(x) = (x-d)^3(x-e)^3\Pi_{l=0}^k (x-x_l)^3. \tag{4.21}$$

Plugging in leads to

$$\begin{aligned} & L_1 q_{k+1,2}(x_{k+1}) = L_1 q_{k+1,1}(x_{k+1}) + \\ & a(x_{k+1})2a_{2+3k+2}\tfrac{d^2}{dx^2}R(x_{k+1}) + b(x_{k+1})a_{2+3k+2}\tfrac{d}{dx}R(x_{k+1}) = f(x_{k+1}). \end{aligned} \tag{4.22}$$

Hence,

$$a_{2+3k+2} = \frac{f(x_{k+1}) - L_1 q_{k+1,1}(x_{k+1})}{a(x_{k+1})\frac{d^2}{dx^2}R(x_{k+1}) + b(x_{k+1})\frac{d}{dx}R(x_{k+1})} \tag{4.23}$$

Finally, let

$$\begin{aligned} q_{k+1,3}(x) \quad & = q_{k+1,2}(x) + a_{2+3k+1}(x-d)^3(x-e)^3\Pi_{l=0}^k (x-x_l)^3 \\ & = a_{2+3k+1}R(x) \end{aligned} \tag{4.24}$$

Plugging in leads to

$$\begin{aligned} & L_1 q_{k+1,3}(x_{k+1}) = L_1 q_{k+1,2}(x_{k+1}) + a(x_{k+1})a_{2+3k+1}\tfrac{d^2}{dx^2}R(x_{k+1}) \\ & + b(x_{k+1})a_{2+3k+1}\tfrac{d}{dx}R(x_{k+1}) + c(x_{k+1})a_{2+3k+1}R(x_{k+1}) = f(x_{k+1}). \end{aligned} \tag{4.25}$$

Hence,

$$a_{2+3k+2} = \frac{f(x_{k+1}) - L_1 q_{k+1,2}(x_{k+1})}{a(x_{k+1})\frac{d^2}{dx^2}R(x_{k+1}) + b(x_{k+1})\frac{d}{dx}R(x_{k+1}) + c(x_{k+1})R(x_{k+1})}. \tag{4.26}$$

It is clear how to proceed inductively in order to get a family of interpolation polynomials which satisfy the differential equation on an increasing set of interpolation points. Note,however,that we have not used any structural information about the coefficients at this point. This means that the equation may be ill-posed,and convergence cannot be guaranteed.

4. 2. The case of scalar linear partial differential equations

For a positive integer k consider an equation of form

$$L_k u \equiv \sum_{|\alpha|\leq k} a_\alpha(x)\frac{\partial^\alpha u}{\partial x^\alpha} = g(x), \tag{4.27}$$

to be solved on the domain Ω where

$$u\Big|_{\partial\Omega} = f \tag{4.28}$$

What we have in mind is an elliptic equation f order k, but ellipticity is not required in order to describe the algorithm which produces a family of multivariate polynomials which satisfy the equation on a set of interpolation points in Ω. Ellipticity becomes important when we want to show that the family of polynomial converges to the solution of the equation (assuming that there is an unique global solution). For simplicity of notation we consider the case $k = 2$, i.e. the situation of (1.9). Assume that $f \in C^k$ and choose a discrete interpolation set $\Theta_b \subset \partial\Omega$. Then we can apply the extended Newton algorithm of Section 3 in order to produce a polynomial $p_b : \mathbb{R}^n \to \mathbb{R}$ such that

$$\begin{aligned} &p_b(x) = f(x) \text{ for all } x \in \Theta_b \\ &\tfrac{\partial p_b}{\partial x^\alpha} = \tfrac{\partial p_b}{\partial x^\alpha} \text{ for all } \alpha \text{ with } |\alpha| \leq l \text{ and } x \in \Theta_b \end{aligned} \tag{4.29}$$

We assume that $\Theta_b = \{x_{0b}, \cdots, x_{Mb}\}$ with $x_{ib} = (x^1_{ib}, \cdots, x^n_{ib})$ and define

$$\Phi_b(x) = \Pi_{i=0b}^{Mb}\Pi_{j=1}^{n}(x^j - x^j_i)^{l+1}. \tag{4.30}$$

Next let $\theta_{int} \subset \Omega \setminus \partial\Omega$ be a set of interpolation points in the interior of Ω. Let

$$\Theta_{int} = \{x_0, \cdots, x_N\}. \tag{4.31}$$

We enumerate (case $k = 2$) the $q := \frac{(n+1)n}{2}$ diffusion coefficients $a_{\alpha_1}, \cdots, a_{\alpha_q}$ (arbitrary order), where we assume $\alpha_l = (\alpha_{l1}, \alpha_{l2})$ and define first q polynomials $p_0^{\mathrm{diff},l}(x), l = 1, \cdots, q$. Let

$$p_0^{\mathrm{diff},1}(x) = p_b(x) + \Phi_b(x)a_{\alpha_1}(x^{\alpha_{11}} - x_0^{\alpha_{11}})(x^{\alpha_{12}} - x_0^{\alpha_{12}}). \tag{4.32}$$

Then we have

$$L_2 p_0^{\mathrm{diff},1}(x_0) = L_2 p_b(x_0) + \Phi_b(x_0)(1 + \delta_{\alpha_{11}\alpha_{12}})a_{\alpha_1} = f(x_0), \tag{4.33}$$

which leads to

$$a_{\alpha_1} = \frac{f(x_0) - L_2 p_b(x_0)}{(1 + \delta_{\alpha_{11}\alpha_{12}})\Phi_b(x_0)} \tag{4.34}$$

Having defined $p_0^{\mathrm{diff},1}(x), \cdots, p_0^{\mathrm{diff},l}(x)$ (and therefore computed $a_{\alpha_1}, \cdots, a_{\alpha_l}$) we define

$$p_0^{\mathrm{diff},l+1}(x) = p_0^{\mathrm{diff},l}(x) + \Phi_b(x)a_{\alpha_{l+1}}(x^{\alpha_{(l+1)1}} - x_0^{\alpha_{(l+1)1}})(x^{\alpha_{(l+1)2}} - x_0^{\alpha_{(l+1)2}}), \tag{4.35}$$

and evaluation leads to

$$a_{\alpha_{l+1}} = \frac{f(x_0) - L_2 p_0^{\mathrm{diff},l}(p_(x_0)}{(1 + \delta_{\alpha_{(l+1)1}\alpha_{(l+1)2}})\Phi_b(x_0)}. \tag{4.36}$$

Proceeding inductively we get a $p_0^{\mathrm{diff},q}(x)$ which equals together with its derivatives up to order l the function f and such that the diffusion part of the operator applied to $p_0^{\mathrm{diff},q}(x)$ equals g at x_0. It is now clear how this procedure can be extended such that an extended polynomial $p_0(x)$ equals together with its derivatives up to order l the function f and such that the total operator applied to $p_0(x)$ equals g at x_0. As in Section 3 the ansatz for the interpolation polynomial p_Θ which satisfies the linear equation on the set of interpolation points $\Theta = \{x_0, \cdots, x_N\}$ then is

$$p_\Theta(x) = \sum_{i=0}^{N} \Pi_{j=1}^{i} \Pi_{k=1}^{n} (x^k - x_{j-1}^k)^3 p_i(x), \tag{4.37}$$

where p_i for $i \geq 2$ are then constructed as p_0 above.

4. 3. The case of a linear hyperbolic equation

We consider the hyperbolic equation mentioned above of the form

$$Lu = f \text{ in } \Omega, \tag{4.38}$$

where

$$Lu \equiv \sum_{ij} h_{ij} \frac{\partial u}{\partial x_i i \partial x_j} + \sum_{i} \frac{\partial}{\partial x_j} + c(x)u \tag{4.39}$$

and (h_{ij}) is a symmetric matrix of signature $(n,1)$, if $\dim\Omega = n+1$. Note that the operator L can be transformed into the form

$$Lu \equiv \Box u + L_1 u, \tag{4.40}$$

where $L_1 u$ is some first order differential operator on Ω. We assume the initial conditions

$$u = g \text{ and } du = \omega, \tag{4.41}$$

where g and ω $(1 - form)$ are initial data. It is clear that the algorithm described in the preceding section can be used in the present situation. Later we shall see that energy estimates imply convergence of the scheme.

5 Further refinements: collocation and parallelization

Numerical experiments show that the coefficients of the recursively computed polynomials have to be computed with increasing accuracy in order to control effects of the truncation error of the coefficients of the polynomials. In the numerical example below, where we computed a polynomial approximation of degree 74 of the locally analytic function

$$x \to \frac{1}{1+x} \tag{5.1}$$

and its derivatives up to order 3 on the interval $[0, 5.4]$ such effects are not observed. However, if we increase the number of derivatives to be approximated up to order $k = 10$ and increase the number of interpolation points, effects of truncation errors can be observed for polynomials of degrees larger than 200. The error increases as $|x|$ becomes large and truncation errors increase. This error can be reduced by a more precise representation of the computational approximation of the real numbers involved in the computation. However, as we point out in this section, we can compute m polynomials $p_1^{\Theta_1}, \cdots, p_m^{\Theta_m}$ of degree $N_1, N_2 \cdots N_m$ parallel which interpolate a given linear system of partial differential equations on some interpolation sets $\Theta_1, \cdots, \Theta_m$ using our basic algorithm, and then compute one polynomial $p_{\sum \Theta}$ which interpolates the same linear system of partial differential equations on the set $\sum \Theta = \Theta_1, \cup \cdots, \cup \Theta_m$. It turns out that this can be in such a way that the truncation error of the resulting polynomial $p_{\sum \Theta}$ is much smaller than in case of a direct extension of one polynomial p_{Θ_i} using the basic algorithm. We call this method the collocation extension of our basic algorithm. We shall assume that the sets of interpolation points are mutually disjunct, i.e.

$$\Theta_i \cap \Theta_j = \oslash \text{ iff } i \neq j. \tag{5.2}$$

It is clear that the computation of the polynomials $p_1^{\Theta_1}, \cdots, p_m^{\Theta_m}$ can be done parallel and only the step of synthesizing has to be done non-parallel. Next we describe that step in case of two polynomials for simplicit of notation. Extension to $m > 2$ polynomials will be clear from that description. So let $\Theta_1, \Theta_2 \subset \Omega \subset \mathbb{R}^n$ be two discrete finite sets of interpolation points of a linear system of partial differential equations $Lu = f$ to be solved on a domain Ω and such that $\Theta_1 \cap \Theta_2 = \oslash$. We write down the polynomial in the univariate case because this simplifies the notation, and the multivariate case is quite similar. Then we define a regular polynomial interpolation formula on $\Theta_1 \cup \Theta_2$ by

$$\begin{array}{l}
\sum_{j=1}^{N} \Pi_{k\neq j,} \frac{(x-x_k^{\Theta_1})^{k+1}}{(x_j^{\Theta_1}-x_k^{\Theta_1})^{k+1}} \Pi_{i=1}^{M} \frac{(x-x_i^{\Theta_2})^{k+1}}{(x_j^{\Theta_1}-x_i^{\Theta_2})^{k+1}} p_{\Theta_1}(x) \\[2ex]
-\sum_j \Pi_{p\in\{1,2\}\ l\neq j} (x-x_l^{\Theta_p})^{k+1} a_{i,1}^j (x-x_j^{\Theta_1})^i \\[2ex]
+\sum_{j=1}^{M} \Pi_{k\neq j} \frac{(x-x_k^{\Theta_1})^{k+1}}{(x_j^{\Theta_1}-x_k^{\Theta_1})^{k+1}} \Pi_{i=1}^{N} \frac{(x-x_i^{\Theta_2})^{k+1}}{(x_j^{\Theta_1}-x_i^{\Theta_2})^{k+1}} p_{\Theta_2}(x) \\[2ex]
-\sum_j \Pi_{p\in\{1,2\}\ l\neq j} (x-x_l^{\Theta_p})^{k+1} a_{i,2}^j (x-x_j^{\Theta_2})^i \\[2ex]
=: \sum_j q_{\Theta_2,\Theta_1}^{1j}(x) p_{\Theta_1}(x) + h_1^a(x) \\[2ex]
+\sum_j q_{\Theta_1,\Theta_2}^{2j}(x) p_{\Theta_2}(x) + h_2^a(x),
\end{array} \tag{5.3}$$

where the constants $a_{i,p}^j, p \in \{1,2\}$ are computed recursively as follows: For each j we can define $a_{0,p}^j = 0$. If $a_{1,p}^j, \cdots, a_{l-1,p}^j$ are determined, then compute $a_{1,p}^j$ via

$$\sum_{1\leq r\leq l} \left(\begin{array}{c} l \\ r \end{array} \right) D_x^r q_{\Theta_1\Theta_2}^{pj}(x_j) D_x^{l-r} p_{\Theta_p}(x_j) = D_x^l h_p^a(x_j) \tag{5.4}$$

for each j. Note that this 'synthesis of polynomials' improves the computational power of our method dramatically. In the example below, where we approximate a simple locally analytic function

$$x \rightarrow \frac{1}{1+x} \tag{5.5}$$

(with convergence radius 1) and its derivatives up to the third derivative on the interval $[0, 5.4]$ with 19 interpolation points $\Theta_1 = \{k0.3|k = 0, \cdots 18\}$ we compute a polynomial of degree 74 in half a minute on a modest laptop machine. If we want to compute a polynomial which gives the same kind of approximation on the interval $[0, 5836, 8]$ it will take several weeks. However, using parallelization and synthesis, and using the rough estimate that synthesis takes in average the same time as building the 1024 basis polynomials of degree 74 on the intervals $[0, 5.4]$ and $[k5.7, (k + 1)5.7], k = 1, \cdots 1023$ we need 10 steps of parallel synthesis of pairs of polynomials of cost of a less than a minute to get a regular approximation polynomial which is at least of degree 75776! It is clear fromthe preceding remarks how to extend this to the multivariate case (cf. also [7]).

6 Convergence of polynomial approximations of global solutions of linear elliptic PDE and error estimates by a priori estimates

Up to now we just considered (regular) polynomial interpolation on given sets of interpolation points. In this section we consider standard problems in the theory of linear partial differential equations and derive the convergence of our algorithm and error estimates (as the mesh size of the sets of interpolation points converges to zero). We start with elliptic equations and then consider hyperbolic problems. Similar results can be obtained for initial-value boundary problems for parabolic equations (since analogous error estimates can be obtained). In this case, however, it turns out that (at least for regular data) a WKB-expansion of the fundamental solution has better convergence properties and error estimates can be obtained by Safanov a priori estimates (cf. [8] and [6]). We shall consider application of our algorithm to this case in the next section. Note that Since to get an error from simple Taylor expansion in genera, because the interpolated function is unknown.

6. 1. Convergence for elliptic equations with regular data

We consider the Dirichlet problem for elliptic equations, i.e. an equation of the form

$$Lu = \sum_{|\alpha| \leq k} a_\alpha(x) \frac{\partial u}{\partial x^\alpha} = f(x) \tag{6.1}$$

on a domain $\Omega \subseteq \mathbb{R}^n$. coefficient functions

$$x \to a_\alpha(x), \tag{6.2}$$

and where u is given on the boundary, i.e.

$$u\Big|_{\partial\Omega} = g. \tag{6.3}$$

We consider the classical case where $k = 2$ and Ω is bounded. We assume uniform ellipticity, i.e. there exists a constant $K > 0$ such that for all $x \in \Omega$

$$\sum_{ij=1}^{n} a_{ij}(x)\xi_i\xi_j \geq K|\xi|^2. \tag{6.4}$$

In the classical case Schauder boundary estimates are available. We cite them in the context of a standard existence result. for a scalar function h in Ω we introduce the norms

$$\|h\|_k^{bd} = \sum_{j=0}^{k} \sum_{|\delta|=j} \|D^\delta h\|_0 \tag{6.5}$$

where

$$\|h\|_0 := \sup_{x\in\Omega} |h(x)|, \tag{6.6}$$

and

$$\|h\|_{k+\alpha}^{bd} = \|h\|_k^{bd} + \sum_{j=0}^{k} \sum_{|\delta|=j} H_\alpha^{bd}\left(D^\delta h\right), \tag{6.7}$$

where $H_\alpha^{bd}(f)$ is the Hölder coefficient of a given function f in Ω. We assume that the coefficient functions $x \to a_{ij}(x)$ (diffusion terms), $x \to b_i(x)$ (drift terms), the potential term $(x \to c(x))$, and the right side $x \to f(x)$ are uniformly Hölder continuous (exponent α) such that

$$\|a_{ij}\|_\alpha^{bd} \leq C, \|b_i\|_\alpha^{bd} \leq C, \|c\|_\alpha^{bd} \leq C, \|f\|_\alpha^{bd} \leq C \tag{6.8}$$

for some generic constant C.

Theorem 6.1 *Assume that conditions* (6.4) *and* (6.8) *hold, and assume that* $c \leq 0$. *Furthermore, assume that* $\partial\Omega$ *belongs to* $C^{2+\alpha}$ *and that* g *belongs to* $C^b_{2\alpha}d$. *Then the inequalities*

$$\begin{aligned} \|u\|_{2+\alpha}^{bd} &\leq C\left(\|g\| + \|u\|_0 + \|f\|_\alpha^{bd}\right) \\ &\leq C\left(\|g\| + \sup_{\partial\Omega}|g| + C\sup_\Omega|f| + \|f\|_\alpha^{bd}\right) \end{aligned} \tag{6.9}$$

hold. Furthermore there exist a unique solution $u \in C^{bd}_{2+\alpha}$ *to the Dirichlet problem.*

The interpolation polynomial p_Θ described in the preceding section is by construction such that

$$L(u - p_\Theta) = \sum_{|\alpha|\leq k} a_\alpha(x)\frac{\partial(u - p_\theta)}{\partial x^\alpha} = \Delta f(x), \tag{6.10}$$

and

$$u - p_\Theta\big|_{\partial\Omega} = \Delta g. \tag{6.11}$$

It follows that

Theorem 6.2 *Assume the same conditions as in theorem 6.1.. Then*

$$\|u - p_\Theta\|_{2+\alpha}^{bd} \leq C\left(\|\Delta g\| + \sup_{\partial\Omega}|\Delta g| + C\sup_\Omega|\Delta f| + \|\Delta f\|_\alpha^{bd}\right) \tag{6.12}$$

Note that this implies an L^2-error even for the second derivatives of the global solution function, hence essentially an estimate in $H^2(\Omega)$. Even stronger results can be obtained if additional equations for the derivatives of u are considered (cf. [7]).

6. 2. Convergence for a hyperbolic linear partial differential equations equation

We consider again the hyperbolic equation mentioned above of the form

$$Lu = f \text{ on } O \subset \Omega, \tag{6.13}$$

where

$$Lu \equiv \sum_{ij} h_{ij} \frac{\partial u}{\partial x_i i \partial x_j} + \sum_i \frac{\partial}{\partial x_j} + c(x)u \tag{6.14}$$

and (h_{ij}) is a symmetric matrix of signature $(n,1)$, if $\dim\Omega = n+1$. We assume that some $O \subset \Omega$ is bounded by two spacelike surfacesΣ_i and Σ_e and swept out by a family of spacelike surfaces $\Sigma_e(s)$. Recall that the initial conditions

$$u = g \text{ and } du = \omega. \tag{6.15}$$

Let p be the interpolation polynom described above such that

$$L(u-p) = \Delta f \text{ on } O \subset \Omega. \tag{6.16}$$

$$u - p = \Delta g \text{ and } du = \Delta\omega. \tag{6.17}$$

Then we use the following energy estimate

Proposition 6.3 *Let u solve the intial value problem* (6.13), (6.15). *Let*

$$O(s) = \overline{O} \cap \{t \leq s\} \tag{6.18}$$

(swept out by the spacelike surfaces $\Sigma_e(s)$). Then

$$\begin{aligned} &\textstyle\int_{O(s)} |u|^2 dV \leq \\ &\textstyle\int_{\Sigma_i^b(s)} |g|^2 dS + C(s-s_0) \int_{\Sigma_i} \left(|g|^2 + |\omega|^2\right) dS + C \int_{O(s)} |f|^2 dV \end{aligned} \tag{6.19}$$

for $s \in [s_0, s_1]$.

This implies

Theorem 6.4 *With the same assumptions as in propostion 6.2. we have*

$$\begin{aligned} &\textstyle\int_{O(s)} |u-p|^2 dV \leq \\ &\textstyle\int_{\Sigma_i^b(s)} |\Delta g|^2 dS + C(s-s_0) \int_{\Sigma_i} \left(|\Delta g|^2 + |\omega|^2\right) dS + C \int_{O(s)} |\Delta f|^2 dV \end{aligned} \tag{6.20}$$

for $s \in [s_0, s_1]$.

Hence the polynomial interpolation scheme described in Section 4 leads to L^2-convergence. One can improve this scheme assuming regularity of solutions and considering systems of equations including equations for derivatives of the solution u (cf. [7]).

7 Applications to parabolic equations (connection to WKB-expansions)

We summarize some results concerning WKB-expansions of parabolic equations (cf. [6] for details). Let us consider the parabolic diffusion operator

$$\frac{\partial u}{\partial t} - Lu \equiv \frac{\partial u}{\partial t} - \frac{1}{2}\sum_{i,j} a_{ij}\frac{\partial^2 u}{\partial x_i \partial x_j} - \sum_i b_i \frac{\partial u}{\partial x_i}, \tag{7.1}$$

where the diffusion coefficients a_{ij} and the first order coefficients b_i in (7.1) depend on the spatial variable x only. In the following let $\delta t = T - t$, and let

$$(x,y) \rightarrow d(x,y) \geq 0, \quad (x,y) \rightarrow c_k(x,y), \; k \geq 0 \tag{7.2}$$

denote some smooth functions on the domain $\mathbb{R}^n \times \mathbb{R}^n$. Then a set of (simplified) conditions sufficient for pointwise valid WKB-representations of the form

$$p(\delta t, x, y) = \frac{1}{\sqrt{2\pi\delta t}^n}\exp\left(-\frac{d^2(x,y)}{2\delta t} + \sum_{k=0}^{\infty} c_k(x,y)\delta t^k\right), \tag{7.3}$$

for the solution $(t,x) \rightarrow p(\delta t, x, y)$.

$$\begin{aligned} &\frac{\partial u}{\partial \delta t} - Lu = 0, \text{with final value} \\ &u(0,x,y) = \delta(x-y), \end{aligned} \tag{7.4}$$

is given by

(A) The operator L is uniformly elliptic in $\mathbb{R}^n$, i.e. the matrix norm of $(a_{ij}(x))$ is bounded below and above by $0 < \lambda < \Lambda < \infty$ uniformly in x,

(B) the smooth functions $x \rightarrow a_{ij}(x)$ and $x \rightarrow b_i(x)$ and all their derivatives are bounded.

For more subtle (and partially weaker conditions) we refer to [6]. We consider the case where there exists a global transformation to the Laplace operator. If we add the uniform boundedness condition

(C) there exists a constant c such that for each multiindex α and for all $1 \leq i,j,k \leq n$,

$$\left|\frac{\partial a_{jk}}{\partial x^\alpha}\right|, \; \left|\frac{\partial b_i}{\partial x^\alpha}\right| \leq c\exp\left(c|x|^2\right), \tag{7.5}$$

then the function $d^2 = (x-y)^2$ (in the transformed coordinates and c_k equals its Taylor expansion around $y \in \mathbb{R}^n$, i.e $c_k, k \geq 0$ have the power series representations

$$c_k(x,y) \quad = \sum_\alpha c_{k,\alpha}(y)\delta x^\alpha, k \geq 0. \tag{7.6}$$

Moreover $c_k, k \geq 0$ are determined by the recursive equations

$$-\frac{n}{2} + \frac{1}{2}Ld^2 + \frac{1}{2}\sum_i\left(\sum_j \left(a_{ij}(x) + a_{ji}(x)\right)\frac{d^2_{x_j}}{2}\right)\frac{\partial c_0}{\partial x_i}(x,y) = 0, \tag{7.7}$$

where the boundary condition

$$c_0(y,y) = -\frac{1}{2}\ln\sqrt{\det\left(a^{ij}(y)\right)} \tag{7.8}$$

determines c_0 uniquely for each $y \in \mathbb{R}^n$, and for $k+1 \geq 1$ we have

$$\begin{array}{ll} (k+1)c_{k+1}(x,y) + \frac{1}{2}\sum_{ij} a_{ij}(x)\left(\frac{d^2_{x_i}}{2}\frac{\partial c_{k+1}}{\partial x_j} + \frac{d^2_{x_j}}{2}\frac{\partial c_{k+1}}{\partial x_i}\right) \\ = \frac{1}{2}\sum_{ij} a_{ij}(x)\sum_{l=0}^{k}\frac{\partial c_l}{\partial x_i}\frac{\partial c_{k-l}}{\partial x_j} + \frac{1}{2}\sum_{ij} a_{ij}(x)\frac{\partial^2 c_k}{\partial x_i \partial x_j} + \sum_i b_i(x)\frac{\partial c_k}{\partial x_i}, \end{array} \tag{7.9}$$

with boundary conditions

$$c_{k+1}(x,y) = R_k(y,y) \text{ if } \quad x = y, \tag{7.10}$$

R_k being the right side of (7.9). In case $a_{ij} = \delta_{ij}$ we have the representations

$$d^2(x,y) = \sum_i (x_i - y_i)^2, \tag{7.11}$$

$$c_0(x,y) = \sum_i (y_i - x_i)\int_0^1 b_i(y + s(x-y))ds, \tag{7.12}$$

and

$$c_{k+1}(x,y) = \int_0^1 R_k(y + s(x-y), y)s^k ds, \tag{7.13}$$

R_k being again the right-hand-side of (7.9). The integrals can be taken out if the functions $x \to b_i(x)$ are given by multivariate power series and error estimates for the truncation error in space and time are obtained (cf. [6, 8]). However, even if the coefficient functions are analytic, i.e. equal locally a power series, it is not possible to approximate such a function globally by their Taylor polynomial. As an example consider the equation

$$\frac{\partial u}{\partial t} - \frac{1}{2}\Delta u - \sum_i^n \frac{1}{1+x_i}\frac{\partial u}{\partial x_i} = 0 \tag{7.14}$$

Here, the coefficient functions

$$x_i \to \frac{1}{1+x_i} = b_i(x) \tag{7.15}$$

are univariate locally analytic function with convergence radius 1. Such type of equations occur in praxis of finance (cf. [4, 8]). In order to obtain an approximation of the WKB-expansion say up to order 5, i.e. compute the coefficient functions

$$x \to c_k(x,y), k = 0, \cdots, 5, \tag{7.16}$$

we need a global approximation of the functions (7.15) and their derivatives up to order 10! This is due to the recursion equations for the $c_k, k \geq 1$ which involve second derivatives of c_{k-1}. If we have 20 interpolation points on the x-axis this implies that our regular interpolation algorithm computes a polynomial of order 231. We do the computation in a more modest example in order to keep the resulting polynomial representable on one page in the following section.

8 A numerical example

The following polynomial is a similtaneous approximation of the function

$$\begin{aligned} &f : [0,5,4] \subseteq \mathbb{R} \to \mathbb{R} \\ &f(x) = \tfrac{1}{1+x} \end{aligned} \tag{8.1}$$

and its first, second, and third derivative on the domain $[0,5,4]$ with 19 interpolation points. Hence the degree of this univariate polynomial is 74. Note that the convergence radius of f is 1.

$$p_{76}(x) = \sum_{m=0}^{75} a_m (x - x_{m\mathrm{div}4})^{m\mathrm{mod}4} \Pi_{l=0}^{m\mathrm{div}4-1} (x - x_l)^4 \tag{8.2}$$

Note that

$$\frac{d^n}{dx^n}\left(\frac{1}{1+x}\right)|_{x=0} = \frac{(-1)^n n!}{(1+x)^{n+1}}|_{x=0} = (-1)^n n! \tag{8.3}$$

This leads to the values $a_0 = 1$, $a_1 = -1$, $a_2 = 1$, and $a_3 = -1$ for the coefficients of our interpolation polynomial at $x_0 = 0$. Note that the coefficients a_i of the interpolation polynomial tend to become smaller for large indexes i as you would expect.

$$
\begin{aligned}
&a_0 = 1.0\\
&a_1 = -1.000000000000,\ a_2 = 1.000000000000,\ a_3 = -1.000000000000\\
&a_4 = 0.769230765432,\ a_5 = -0.591715921811,\ a_6 = 0.455165657066\\
&a_7 = -0.350124490177,\ a_8 = 0.218822618520,\ a_9 = -0.136753442090\\
&a_{10} = 0.085452696109,\ a_{11} = -0.053404312398,\ a_{12} = 0.028144617397\\
&a_{13} = -0.014953338935,\ a_{14} = 0.008262188243,\ a_{15} = -0.005370784216\\
&a_{16} = 0.003734873988,\ a_{17} = -0.003633027430,\ a_{18} = 0.004645502211\\
&a_{19} = -0.006813570816,\ a_{20} = 0.007086610952,\ a_{21} = -0.007144432312\\
&a_{22} = 0.006238342564,\ a_{23} = -0.002646146059,\ a_{24} = -0.002374282360\\
&a_{25} = 0.008387675067,\ a_{26} = -0.015766978592,\ a_{27} = 0.024857498610\\
&a_{28} = -0.025373687351,\ a_{29} = 0.025025735340,\ a_{30} = -0.023974098174\\
&a_{31} = 0.022321168853,\ a_{32} = -0.015945627926,\ a_{33} = 0.011155207224\\
&a_{34} = -0.007619506803,\ a_{35} = 0.005069120726,\ a_{36} = -0.002759684498\\
&a_{37} = 0.001479716734,\ a_{38} = -0.000790172686,\ a_{39} = 0.000430223475\\
&a_{40} = -0.000208511304,\ a_{41} = 0.000106279314,\ a_{42} = -0.000056281013\\
&a_{43} = 0.000028862889,\ a_{44} = -0.000011153733,\ a_{45} = 0.000002201139\\
&a_{46} = 0.000002233629,\ a_{47} = -0.000004202246,\ a_{48} = 0.000003699371\\
&a_{49} = -0.000002870941,\ a_{50} = 0.000002068390,\ a_{51} = -0.000001402599\\
&a_{52} = 0.000000753699,\ a_{53} = -0.000000375935,\ a_{54} = 0.000000159621\\
&a_{55} = -0.000000037499,\ a_{56} = -0.000000015690,\ a_{57} = 0.000000032004\\
&a_{58} = -0.000000031616,\ a_{59} = 0.000000023406,\ a_{60} = -0.000000010968\\
&a_{61} = 0.000000001564,\ a_{62} = 0.000000005590,\ a_{63} = -0.000000011521\\
&a_{64} = 0.000000012190,\ a_{65} = -0.000000012095,\ a_{66} = 0.000000011769\\
&a_{67} = -0.000000011467,\ a_{68} = 0.000000008698,\ a_{69} = -0.000000006652\\
&a_{70} = 0.000000005132,\ a_{71} = -0.000000003988,\ a_{72} = 0.000000002523\\
&a_{73} = -0.000000001600,\ a_{74} = 0.000000001015,\ a_{75} = -0.000000000643
\end{aligned}
\tag{8.4}
$$

9 Conclusion

We have designed regular polynomial interpolation algorithms and variations which produce families of multivariate polynomials which solve linear systems of partial differential equations on arbitrary sets of interpolation points. In our basic algorithm the members of the family of polynomials are defined recursively each being an extension of the preceding member in the sense that the preceding member agrees with a given member on the set of interpolation points on which the preceding member satisfies the linear system of partial differential equations. We have shown that the family of multivariate polynomials has the global solution as its natural limit if some a priori information on the system of partial differential equations is available. The information needed can variate from case to case. In any case a solution should exist. We have shown how to use a priori estimates of elliptic equations and of hyperbolic systems of equations in order to obtain error estimates adapted to the regularity of the solution. Similar is true for parabolic equations. All this makes our approach compatible with new techniques like sparse grids or weighted Monte-Carlo algorithms developed in order to treat systems of higher dimension. In case of parabolic equations we showed how regular polynomial interpolation of known functions can be used in order to compute higher order approximations of WKB-expansions of fundamental solutions. We also constructed extensions where the algorithm is parallelized on different set of interpolation points an showed how these partial polynomial approximations can be patched together to one multivariate polynom which fits the given system of linear partial differential equations on the union of sets of interpolation points.

References

[1] de Boor, C. , Ron, A., *On multivariate polynomial interpolation*, Constr. Approx. , 1990.

[2] Dinh-Düng, Calvi, J.P., Trung, N.T., *Polynomial Projectors preserving homogeous partial differential equations*, Journal of Approximation Theory, 2005.

[3] Frazer, R. A., Jones, W.P., Scan, S. W., *Approximations to functions and to solutions of differential equations*, Gr. Br. Aero. Res. Councel. Rep. Memo 1799 , 1937.

[4] Fries, C., Kampen, J., *Proxy Simulation Schemes for generic robust Monte Carlo sensitivities, process oriented importance sampling and high accuracy drift approximation (with applications to the LIBOR market model)*, Journal of Computational Finance, Vol. 10, Nr. 2, 2007.

[5] Gasca,M., Sauer, Th., *On the history of multivariate polynomial interpolation*, J. Comput. Appl. Math. 122 (2000), S. 23-35.

[6] Kampen, J., *The WKB-Expansion of the fundamental solution of linear parabolic equations and its applications* .book, submitted to Memoirs of the American Mathematical Society, (electronically published at SSRN ,2006).

[7] Kampen, J., *Regular polynomial interpolation and linear systems of partial differential equations II: refined error estimates, applications and numerical examples* (in preperation).

[8] Kampen, Jörg; Kolodko, Anastasia; Schoenmakers, John: Monte Carlo Greeks for exotic products via approximative Greenian Kernels, WIAS preprint 2007.

[9] Lodha, S., Goldman, R., *A Unified Approach to evaluation algorithms for Multivaluate polynomials*, Mathematics of computation. 66, 1997, 1521-1553.

[10] Sauer, Th., *Polynomial interpolation of minimal degree*, Numerische Math. 78, 1997, 59-85.

[11] Sauer, Th., *Multiindexing points for poised subproblems in mulivariate interpolation*, Monograficas del Seminario Matematico de Galdeano 31, 2004, 229-234.