Sparsity optimized high order finite element functions for H(curl) on tetrahedra

Sven Beuchler * Veronika Pillwein † Sabine Zaglmayr‡

March 5, 2012

Abstract

$H(\mathrm{curl})$ conforming finite element discretizations are a powerful tool for the numerical solution of the system of Maxwell's equations in electrodynamics. In this paper we construct a basis for conforming high-order finite element discretizations of the function space $H(\mathrm{curl})$ in 3 dimensions. We introduce a set of hierarchic basis functions on tetrahedra with the property that both the L^2-inner product and the $H(\mathrm{curl})$-inner product are sparse with respect to the polynomial degree. The construction relies on a tensor-product based structure with properly weighted Jacobi polynomials as well as an explicit splitting of the basis functions into gradient and non-gradient functions. The basis functions yield a sparse system matrix with $\mathcal{O}(1)$ nonzero entries per row.

The proof of the sparsity result on general tetrahedra defined in terms of their barycentric coordinates is carried out by an algorithm that we implemented in Mathematica. A rewriting procedure is used to explicitly evaluate the inner products. The precomputed matrix entries in this general form for the cell-based basis functions are available online.

1 Introduction

The main result of this paper is the construction of high order finite element basis functions for $H(\mathrm{curl})$ on tetrahedra yielding a sparse system matrix. These basis functions are defined via certain Jacobi-type polynomials. For the proof of the sparsity, integrals over the products of these basis functions and their partial derivatives need to be evaluated. Even though Jacobi polynomials have been studied extensively in the past, for this evaluation several identities are needed that are not yet folklore in the literature. For obtaining these necessary relations we invoke recently developed computer algebra algorithms. Furthermore the amount of data that needs to be handled forbids classical hand computations and we employ a program implemented in Mathematica to carry out this task.

The symbolic component in the construction of the basis functions and the proof of their properties is the main focus of the present work. The gain of invoking symbolic computation is twofold: on the one hand it is used as a practical tool to derive necessary identities and relations, on the other hand it is inevitable for dealing with the large number of integrals to be evaluated in a

*Institute for Numerical Simulation, University of Bonn, Wegelerstraße 6, D-53115 Bonn, Germany, beuchler@ins.uni-bonn.de

†Research Institute for Symbolic Computation, Altenberger Straße 69, A-4040 Linz, Austria, veronika.pillwein@risc.jku.at

‡Institute for Computational Mathematics, TU Graz, A-8010 Graz, Austria sabine.zaglmayr@tugraz.at

systematic manner. For the proof of the main result we use the packages "HolonomicFunctions" [28] and "SumCracker" [27] that are explained in more detail below. These are among several available tools for dealing with special functions in a symbolic way, such as, e.g., [45, 17, 16, 35, 36, 43]. The proof of the main result proceeds by a rewrite procedure of the given integrals that relies on identities discovered using these packages.

Finite element methods are nowadays the preferred tool for numerically solving partial differential equations (PDEs) on complicated domains, see e.g. [33, 13]. In the presence of smooth solutions the convergence rate of this approximation procedure can be accelerated significantly if basis functions of high polynomial degrees are used. This is called the p- and hp-version of the FEM, see e.g. [38, 19, 5]. The implementation of these methods however then becomes very involved and every simplification is most welcome [11, 31, 42, 25, 22, 37].
This note is the last in a series of papers dealing with the construction of sparsity optimized basis functions for different Sobolev spaces [8, 10, 6, 9, 7]. Except for the first one [9] that dealt with basis functions defined on triangles only, the computations were handed over to a computer algebra system. Still, the focus of these papers was on the numerical aspects of the construction.
$H(\mathrm{curl})$ conforming basis functions are chosen to be as piecewise polynomial functions on tetrahedrons with globally continuous tangential components along the interfaces of the tetrahedrons, see [11, 31, 42, 22]. The construction of the basis functions for the vector valued space $H(\mathrm{curl})$ follows the approach presented by Zaglmayr [37, 44]. They are built starting from (in principle) any set of H^1-conforming, i.e. globally continuous, basis functions and they are divided into curl-free basis functions and a set of non-curl-free basis functions that complete the basis. As we show below, they yield sparse system matrices and this is of advantage in the numerical computation concerning both computing time and memory requirement.

The outline of the paper is as follows. Section 2 gives an overview about the mathematical background from partial differential equations which is required to motivate the following sections. Namely, the Maxwell equations and FEM are described very briefly. Finally, the importance of the sparsity of the system matrix is motivated. The basis functions are defined in section 3. The main results are also formulated in this part of the paper. Section 4 summarizes the most important properties of Jacobi polynomials needed.
For the proof of the sparsity properties of the basis functions multi-integrals over certain Jacobi polynomials and weights over general tetrahedra have to be computed. We are evaluating these integrals symbolically using a rewrite procedure that we implemented in Mathematica and that is described in Section 5.

2 Maxwell's equations and the Finite Element Method

Variational formulation and the function space $H(\mathrm{curl})$ In this paper, we investigate the following problem in variational formulation: Find $u \in H(\mathrm{curl}, \Omega)$ such that

$$a(u,v) := \int_\Omega \mu^{-1} \operatorname{curl} u \cdot \operatorname{curl} v + \int_\Omega \kappa\, u \cdot v = \int_\Omega f \cdot v =: F(v) \quad \forall v \in H(\mathrm{curl}, \Omega) \tag{2.1}$$

holds. For ease of notation, we assume Neumann boundary conditions. The bilinear form $a(\cdot,\cdot)$ and the linear form $F(\cdot)$ are well-defined and bounded for $f \in [L_2(\Omega)]^3$ and $\mu, \kappa \in [L_\infty(\Omega)]^3$ with $\mu > 0$ and κ, μ are assumed to be piecewise constant. The variational formulation is well-defined for square-integrable vector-valued functions $u : \Omega \to \mathbb{R}^3$ with square-integrable rotation. We

denote the according function space by

$$H(\mathrm{curl},\Omega) := \{u \in [L_2(\Omega)]^3 \ : \ \mathrm{curl}\, u \in [L_2(\Omega)]^3\}. \tag{2.2}$$

The variational formulation (2.1) is obtained from the discretization of the Maxwell equations [29] by multiplication with a test function v, integration over the domain Ω and applying Green's formula to the curl-curl part. We refer the interested reader to [30, 24, 12] for more informations concerning this topic.
For complicated geometries Ω and real-life data it is not possible to solve the Maxwell equations (2.1) analytically. The finite element method (FEM) provides a general method for the numerical solution of partial differential equations. It is based on the variational formulation of the underlying PDE and provides a profound analysis.

Finite element discretization of $H(\mathrm{curl},\Omega)$ Galerkin methods such as, e.g., FEMs are among the most powerful methods for the solution of boundary value problems of the form (2.1). The Galerkin approximation relies on the orthogonal projection of the implicitly given solution u in (2.1) onto an N-dimensional subspace $\mathbb{V}_N \subset H(\mathrm{curl},\Omega)$ with respect to the bilinear form $a(\cdot,\cdot)$. Therefore, we construct a sequence of finite dimensional spaces $\mathbb{V}_N \subset H(\mathrm{curl},\Omega)$ and consider the solution of (2.1) on $\mathbb{V}_N$ (see e.g. [18, 39] or the textbooks [33, 13]), namely

$$\text{Find } u_N \in \mathbb{V}_N \text{ such that} \qquad a(u_N, v_N) = F(v_N) \quad \forall v_N \in \mathbb{V}_N. \tag{2.3}$$

The finite element method provides a special construction of these discrete spaces $\mathbb{V}_N$ by piecewise polynomial functions on an admissible subdivision (see [18]) $\mathcal{T}_h$ of Ω into simplices τ_s with $s = 1, ..., nel$, i.e.,

$$\mathbb{V}_N := \{u \in H(\mathrm{curl},\Omega) \ : \ u|_{\tau_s} \in \mathcal{P}^p(\tau_s)^3 \ \forall \tau_s \in \mathcal{T}_h\}, \tag{2.4}$$

where $\mathcal{P}^p$ is the space of all polynomials defined on τ_s of maximal total degree p. The elements τ_s are chosen such that κ and μ are constant on the elements. In *hp*-finite element methods the polynomial degree p can vary on each element τ_s which provides extraordinary fast convergence of the finite element method with respect to the number of degrees of freedom $N = \dim(\mathbb{V}_N)$, see e.g. [38]. This is crucial for the solution of real world problems of the form (2.1).
Since the space $\mathbb{V}_N$ is finite dimensional, the space is equipped with a row vector of basis functions $[\Psi] := [\psi_1, \ldots, \psi_N]$. The basis functions ψ_j are chosen such that they have local support (see e.g. [18]).
Then using the ansatz $u_N(x) = \sum_{i=1}^N u_i \psi_i(x)$ and setting $v = \psi_j$ for $j = 1, ..., N$ in (2.3) the problem becomes equivalent to solving the following system of N linear algebraic equations

$$\text{Find } \underline{u} \in \mathbb{R}^N\text{:} \qquad K\underline{u} = \underline{f} \tag{2.5}$$

for the coefficient vector $\underline{u} := [u_i]_{i=1}^N$, where the system matrix and right hand side vector are given by the relations $K = [a(\psi_j, \psi_i)]_{i,j=1}^N \in \mathbb{R}^{N\times N}$ and $\underline{f} = [F(\psi_j)]_{j=1}^N \in \mathbb{R}^N$, respectively. Note that the matrix K depends on the choice of the basis functions.

Efficient solution of algebraic system In practical problems, the dimension N usually becomes very large ($\succeq 10^6$). Iterative methods as the preconditioned GMRES method or the preconditioned conjugate gradient-method (pcg-method) for positive definite systems are preferred for the solution of (2.3). The two main important issues for the fast solution of the system $K\underline{u} = \underline{f}$ are

- the fast multiplication $K\underline{u}$,

- the choice of a good preconditioner C^{-1} for K such that the condition number $\kappa(C^{-1}K)$ becomes small,

in order to obtain a fast convergence of the iterative solver for the solution of (2.5). If K is a dense matrix, the operation $K\underline{u}$ requires N^2 flops. If K is a sparse matrix with a bounded number c of nonzero entries per row, the computational complexity of the matrix vector-multiplication is bounded by cN. Since K in (2.5) depends on the choice of the basis $[\Psi]$ it is essential to choose a basis with as many orthogonality relations as possible with respect to the bilinear form $a(\cdot,\cdot)$. The choice of the basis heavily influences the properties of the matrix K:

- the local support of finite element basis functions yields sparse system matrices K and hence a cheap matrix vector multiplication $K\underline{u}$
- the condition number of K and $C^{-1}K$, respectively, stability and less iterations in iterative solution methods.

In the lower order version of FEM, i.e., the h-version, multigrid solvers are the most powerful methods for discretizations of Maxwell's equations, see [4, 34, 23] and the references therein. For hp-FEM this strategy is combined with appropriate local smoothers and static condensation. We also refer to [20].

***hp*-FEM and choice of basis functions** In hp-FEM, the local polynomial degree p_s on the elements may be large. Despite of the local support of the basis functions $[\Psi]$, the local dimension n_s grows as $\mathcal{O}(p_s^3)$. Hence we are interested in a bounded number of nonzero entries in the system matrix independent of the polynomial degrees. Let $[\Phi_s] = [\phi_{i,s}]_{i=1}^{n_s}$ be the set of all basis functions ψ_j with $\operatorname{supp}\psi_j \cap \tau_s \neq \emptyset$, e.g. $[\Phi_s] = [\Psi]L_s$ with the (boolean) finite element connectivity matrices L_s.
In finite element methods, the global system matrix K is the result of assembling local matrices, see [18]. In our case, one obtains

$$K = \int_\Omega \mu^{-1} \operatorname{curl}[\Psi]^\top \cdot \operatorname{curl}[\Psi] + \kappa\, [\Psi]^\top \cdot [\Psi] = \sum_{s=1}^{nel} \int_{\tau_s} \mu^{-1} \operatorname{curl}[\Psi]^\top \cdot \operatorname{curl}[\Psi] + \kappa\, [\Psi]^\top \cdot [\Psi].$$

Together with $[\Phi_s] = [\Psi]L_s$ on τ_s, this implies

$$K = \sum_{s=1}^{nel} L_s^\top K_s L_s \tag{2.6}$$

with the local stiffness and mass matrices

$$\begin{aligned} K_s &= \int_{\tau_s} \left(\mu \operatorname{curl}[\Phi_s]^\top \cdot \operatorname{curl}[\Phi_s] + \kappa [\Phi_s]^\top \cdot [\Phi_s]\right) \qquad (2.7)\\ &= \mu_s \int_{\tau_s} \operatorname{curl}[\Phi_s]^\top \cdot \operatorname{curl}[\Phi_s] + \kappa_s \int_{\tau_s} [\Phi_s]^\top \cdot [\Phi_s] =: \mu_s A_s + \kappa_s M_s \end{aligned}$$

on the elements τ_s, respectively, where $\mu_s = \mu\,|_{\tau_s}$ and $\kappa_s = \kappa\,|_{\tau_s}$ are constants.
Therefore, the sparsity of the matrices K_s in (2.7) implies sparsity of the matrix K, cf. (2.6). Our aim is to develop a local polynomial basis $[\Phi_s]$ such that the matrices A_s and M_s in (2.7) have a bounded number of nonzero entries per row. The global basis is the obtained in the usual way, see e.g. [21].

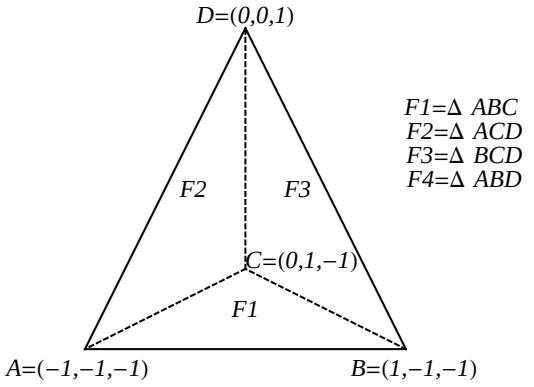

Figure 1: Notation of the vertices and edges/faces on the reference element $\hat{\triangle}$.

Model problem For ease of presentation, we are focusing on the following model problem: Let $\triangle$ denote an arbitrary non degenerated simplex $\triangle \subset \mathbb{R}^3$. Find a polynomial basis $[\Phi] = [\phi_i]_{i=1}^{n(p)}$ of degree p with $\phi_i : \triangle \mapsto \mathbb{R}^3$ such that the matrices

$$\begin{aligned} M &:= \left[\int_{\triangle} \phi_j \cdot \phi_i\right]_{i,j=1}^{n} = \int_{\triangle} [\Phi]^\top \cdot [\Phi] \\ A &:= \left[\int_{\triangle} \operatorname{curl} \phi_j \cdot \operatorname{curl} \phi_i\right]_{i,j=1}^{n} = \int_{\triangle} \operatorname{curl}[\Phi]^\top \cdot \operatorname{curl}[\Phi] \end{aligned} \tag{2.8}$$

have $\mathcal{O}(n)$ nonzero entries. This basis should be suited for $H(\mathrm{curl})$ conformity, [31]. This result is formulated as our main result of this paper, Theorem 3.2. Nevertheless, the suggested basis functions can be applied in any variational problem setting based on a $H(\mathrm{curl})$-conforming discretization.

3 Definition of the basis function

In this section we first define the basis functions, also referred to as shape functions, on a general 3-dimensional simplex τ_s, which are appropriate for tangential continuous, i.e. $H(\mathrm{curl})$-conforming, discretizations and imply sparse element stiffness and mass matrices. Note that enforcing tangential continuity prohibits to construct basis functions that are orthogonal with respect to the bilinear form $a(\cdot,\cdot)$ on the given subdivision, hence the goal is to find sparsity optimized bases. In order to allow for globally varying polynomial degree we use as common an edge-face-cell-based construction [2, 19] of the basis functions. For ease of notation we assume a uniform polynomial degree p.

Let $\triangle$ denote an arbitrary non-degenerated simplex $\triangle \subset \mathbb{R}^3$, its set of four vertices by $\mathcal{V} = \{V_1, V_2, V_3, V_4\}$, $V_i \in \mathbb{R}^3$ and $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \in \mathcal{P}^1(\triangle)$ its barycentric coordinates uniquely defined by $\lambda_i(V_j) = \delta_{ij}$, see Figure 1. The general construction concept follows [44]: The set of edge-based shape functions consists of the lowest-order Nédélec function shape functions and curl-free shape functions. The set of face-based shape functions and the set of interior based shape functions are split into a set of gradient and a set of non-gradient completion functions. Aiming at special orthogonality relations between the different basis functions we adapt the polynomial building blocks in the tensor-product construction. A detailed motivation based on the de-Rham complex is given in [8].

For this purpose we introduce properly weighted Jacobi-type polynomials as follows: For $n \geq 0, \alpha, \beta > -1$ and $x \in [-1,1]$ let

$$P_n^{(\alpha,\beta)}(x) = \frac{(-1)^n}{2^n n!(1-x)^\alpha(1+x)^\beta} \frac{\mathrm{d}^n}{\mathrm{d}x^n}\left((1-x)^{n+\alpha}(1+x)^{n+\beta}\right) \tag{3.1}$$

be the nth Jacobi polynomial with respect to the weight function $(1-x)^\alpha(1+x)^\beta$. The function $P_n^{(\alpha,\beta)}(x)$ is a polynomial of degree n, i.e. $P_n^{(\alpha,\beta)}(x) \in \mathbb{P}_n((-1,1))$, where $\mathbb{P}_n(I)$ is the space of all polynomials of degree n on the interval I. In the special case $\alpha = \beta = 0$, the functions $P_n^{(0,0)}(x)$ are called Legendre polynomials. Mainly, we will use Jacobi polynomials with $\beta = 0$. For sake of simple notation we therefore omit the second index in (3.1) and write $p_n^\alpha(x) := P_n^{(\alpha,0)}(x)$. Moreover for $n \geq 1$, let

$$\hat{p}_n^\alpha(x) = \int_{-1}^{x} p_{n-1}^\alpha(y)\,\mathrm{d}y, \quad \text{with} \quad \hat{p}_0^\alpha(x) = 1, \tag{3.2}$$

be the nth integrated Jacobi polynomial. Some properties of this type of Jacobi polynomials and relations between different Jacobi and integrated Jacobi polynomials are stated in section 4. For more details on Jacobi polynomials we refer the interested reader to the books of Abramowitz and Stegun [1], Szegö [40], and Tricomi [41].
Now, the basis functions can be defined as follows.

Edge-based shape functions For each edge $E_m = [e_1, e_2]$, $m = 1, \ldots, 6$, characterized by the vertices, we introduce the lowest-order Nédélec function [32] corresponding to the edge $[e_1, e_2]$

$$\varphi_{1,E_m} := \nabla\lambda_{e_1}\,\lambda_{e_2} - \lambda_{e_1}\nabla\lambda_{e_2} \tag{3.3}$$

and the high order functions gradient fields of scalar functions, [37],

$$\varphi_{i,E_m} := \nabla\left(\hat{p}_i^0\left(\frac{\lambda_{e_2}-\lambda_{e_1}}{\lambda_{e_2}+\lambda_{e_1}}\right)\left(\lambda_{e_2}+\lambda_{e_1}\right)^i\right), \quad 2 \leq i \leq p+1. \tag{3.4}$$

The vector of the edge based basis functions of one fixed edge e is denoted by

$$[\Phi_e] := [\varphi_{i,e}]_{i=1}^{p+1},$$

the vector of all edge based basis functions by

$$[\Phi_E] := \left[\begin{array}{ccccccc} [\Phi_{E_1}] & [\Phi_{E_2}] & [\Phi_{E_3}] & [\Phi_{E_4}] & [\Phi_{E_5}] & [\Phi_{E_6}] \end{array}\right]. \tag{3.5}$$

Face-based shape functions For each face $f = [f_1, f_2, f_3]$, characterized by the vertices V_{f_1}, V_{f_2} and V_{f_3}, we choose face based basis functions as

$$\begin{aligned} \varphi_{1j}^{F,1} &:= (\nabla\lambda_{f_1}\lambda_{f_2} - \nabla\lambda_{f_2}\lambda_{f_1})v_{1j}^F, && 1 \leq j \leq p-1, \\ \varphi_{ij}^{F,1} &:= \nabla u_i^F v_{ij}^F - \nabla v_{ij}^F u_i^F, && 2 \leq i; 1 \leq j; i+j \leq p+1, \\ \varphi_{ij}^{F,2} &:= \nabla(u_i^F v_{ij}^F), && 2 \leq i; 1 \leq j; i+j \leq p+1 \end{aligned} \tag{3.6}$$

using the face-based Jacobi-type polynomials

$$u_i^F := \hat{p}_i^0\left(\frac{\lambda_{f_2}-\lambda_{f_1}}{\lambda_{f_2}+\lambda_{f_1}}\right)\left(\lambda_{f_2}+\lambda_{f_1}\right)^i, \qquad v_{ij}^F := \hat{p}_j^{2i-1}\left(\lambda_{f_3}-\lambda_{f_2}-\lambda_{f_1}\right). \tag{3.7}$$

Let $[\Phi^f] := \left[\left[\varphi_{1j}^{F,1}\right]_{j=1}^{p-1}, \left[\varphi_{ij}^{F,1}\right]_{i=2,j=1}^{i+j\le p+1}, \left[\varphi_{ij}^{F,2}\right]_{i=2,j=1}^{i+j\le p+1}, \right]$ denote the row vector of the face based basis functions of one fixed face f, and

$$[\Phi_F] = \left[\begin{array}{cccc} [\Phi^{F1}] & [\Phi^{F2}] & [\Phi^{F3}] & [\Phi^{F4}] \end{array} \right] \tag{3.8}$$

be the row vector of all face based basis functions.

Interior (cell-based) shape functions The interior (cell-based) basis functions are constructed in two types. First we define the curl-free shape functions by the gradients

$$\varphi_{ijk}^{(b)}(x,y,z) := \nabla\big(u_i(x,y,z)\, v_{ij}(x,y,z)\, w_{ijk}(x,y,z)\big), \quad i\ge 2; j,k\ge 1;\ i+j+k\le p+1 \tag{3.9}$$

and complete the basis with the non-curl free cell-based shape functions

$$\begin{aligned}
\tilde\varphi_{1jk}^{(a)}(x,y,z) &:= \varphi_{1,E_1}(x,y,z) v_{1j}(x,y,z) w_{1jk}(x,y,z), && j,k\ge 1;\ j+k\le p-1,\\
\tilde\varphi_{ijk}^{(b)}(x,y,z) &:= \nabla u_i(x,y,z) v_{ij}(x,y,z) w_{ijk}(x,y,z), && i\ge 2; j,k\ge 1;\ i+j+k\le p+1,\\
\tilde\varphi_{ijk}^{(c)}(x,y,z) &:= u_i(x,y,z) v_{ij}(x,y,z) \nabla w_{ijk}(x,y,z), && i\ge 2; j,k\ge 1;\ i+j+k\le p+1,
\end{aligned} \tag{3.10}$$

where φ_{1,E_1} is the Nédélec function (3.3). Furthermore, we introduce the auxiliary functions u_i, v_{ij} and w_{ijk} by the mixed-weighted Jacobi-type polynomials

$$\begin{aligned}
u_i(x,y,z) &:= \hat p_i^0\left(\frac{\lambda_2-\lambda_1}{\lambda_2+\lambda_1}\right)(\lambda_2+\lambda_1)^i,\\
v_{ij}(x,y,z) &:= \hat p_j^{2i-1}\left(\frac{2\lambda_3-(1-\lambda_4)}{1-\lambda_4}\right)(1-\lambda_4)^j,\\
\text{and}\quad w_{ijk}(x,y,z) &:= \hat p_k^{2i+2j-2}(2\lambda_4-1),
\end{aligned} \tag{3.11}$$

respectively. Finally, we denote the row vectors of the corresponding basis functions as $[\Phi_b] = \left[\varphi_{ijk}^{(b)}(x,y,z)\right]_{i\ge 2,j,k,\ge 1}^{i+j+k\le p+1}$, and $[\widetilde\Phi_a] = \left[\tilde\varphi_{1jk}^{(a)}(z)\right]_{j,k=1}^{j+k\le p-1}$, $[\widetilde\Phi_b] = \left[\tilde\varphi_{ijk}^{(b)}(x,y,z)\right]_{i\ge 2,j,k,\ge 1}^{i+j+k\le p+1}$ and $[\widetilde\Phi_c] = \left[\tilde\varphi_{ijk}^{(c)}(x,y,z)\right]_{i\ge 2,j,k,\ge 1}^{i+j+k\le p+1}$. The vector of the interior shape functions is denoted by

$$[\Phi_I] := \left[\begin{array}{cc} [\Phi_b] & [\Phi_2] \end{array} \right] \quad \text{with} \quad [\Phi_2] := \left[\begin{array}{ccc} [\widetilde\Phi_a] & [\widetilde\Phi_b] & [\widetilde\Phi_c] \end{array} \right] \tag{3.12}$$

and the vector of all basis functions is given by

$$[\Phi] := \left[\begin{array}{ccc} [\Phi_E] & [\Phi_F] & [\Phi_I] \end{array} \right]. \tag{3.13}$$

Lemma 3.1. *Let $p\ge 1$. Then, the shape functions $[\Phi]$ are linearly independent spanning $(\mathcal{P}^p(T))^3$. Moreover, the rotations $[\nabla\times\Phi_2]$ span $(\mathcal{P}^{p-1}(T))^3|_{\mathbb{R}}$.*

Proof. The proofs of [44] also hold for the auxiliary functions (3.11), which implies the result. □

With these basis functions, see (3.13), the inner block of the system matrix is sparse and the number of nonzero entries is proportional to its dimension, where the blocks corresponding to the interior bubbles $[\Phi_I]$ (3.12) are defined by the relations

$$M^{II} := \int_\triangle [\Phi_I]^\top \cdot [\Phi_I] \quad \text{and} \quad A^{II} := \int_\triangle \operatorname{curl}[\Phi_I]^\top \cdot \operatorname{curl}[\Phi_I]. \tag{3.14}$$

This result is summarized in Theorem 3.2. Note that the statements below describe an upper bound only.

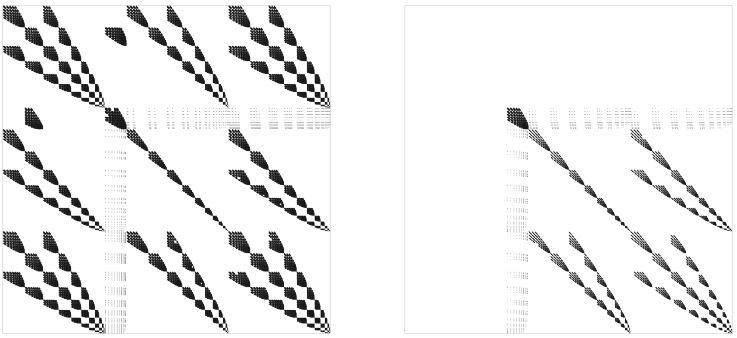

Figure 2: Inner block of mass (left) and stiffness matrix (right) for $p = 14$

Theorem 3.2. *Let M^{II} and K^{II} be inner blocks of the mass matrix and stiffness matrix, respectively, see* (3.14).
The inner block of the mass matrix M^{II} has in total $\mathcal{O}(p^3)$ nonzero matrix entries. More precisely, $M^{II}_{ijk;lmn} = 0$ if $|i-l| > 2$ or $|i-l+j-m| > 6$ or $|i-l+j-m+k-n| > 6$.
The inner block of the stiffness matrix A^{II} has in total $\mathcal{O}(p^3)$ nonzero matrix entries. More precisely, $A^{II}_{ijk;lmn} = 0$ if $|i-l| > 2$ or $|i-l+j-m| > 4$ or $|i-l+j-m+k-n| > 4$.

Proof. Explicit computation of the matrix entries using the algorithm described in Section 5, see also [8]. □

Remark 3.3. *The result considers only the inner blocks for the element mass matrix M and element stiffness matrix A on $\triangle$, see* (2.8). *Because of* (3.13), *also edge and face based shape functions are involved in the definition of the local basis functions $[\Phi]$. However, the asymptotically biggest portion of basis functions belong to $[\Phi_I]$, since the dimensions of $[\Phi_E]$ and $[\Phi_F]$ grow only as $\mathcal{O}(p)$ and $\mathcal{O}(p^2)$, respectively, cf. (3.5,3.4) and (3.8,3.6). Besides that, with the definition of the edge and face based functions in terms of Jacobi-type polynomials similar orthogonality effects can be observed for the coupling blocks between $[\Phi_E]$, $[\Phi_F]$ and $[\Phi_I]$ as well as for the face and edge blocks.*

If this sparsity pattern is taken into account, only $\mathcal{O}(p^3)$ matrix entries need to be considered for computation in the case of a polygonal domain and piecewise constant coefficients.
The explicit values for the mass and stiffness matrix entries on a general element defined by its barycentric coordinates are available online at

`http://www.risc.jku.at/people/vpillwei/hcurl/` .

4 Algorithms for special functions

Sparsity optimization of high-order basis functions on simplices relies on using particularly chosen Jacobi-type polynomials and some of their basic properties that are introduced in this section. First recall that we use the notation $p_n^\alpha(x)$ to denote Jacobi polynomials (3.1) with parameter $\beta = 0$. These polynomials are orthogonal with respect to the weight $(1-x)^\alpha$, i.e., there holds

$$\int_{-1}^{1} (1-x)^\alpha p_j^\alpha(x) p_l^\alpha(x)\,\mathrm{d}x = \rho_j^\alpha \delta_{jl}, \quad \text{where} \quad \rho_j^\alpha = \frac{2^{\alpha+1}}{2j+\alpha+1}. \tag{4.1}$$

This orthogonality relation is the basic ingredient for computing the entries of the different blocks of the mass and stiffness matrices. Recall also the definition of the integrated Jacobi polynomials (3.2) for $n \geq 1$ and $\alpha > -1$,

$$\hat{p}_n^\alpha(x) = \int_{-1}^{x} p_{n-1}^\alpha(y)\, \mathrm{d}y, \quad \text{with} \quad \hat{p}_0^\alpha(x) = 1.$$

Obviously, $\hat{p}_n^\alpha(-1) = 0$ for $n \geq 1$, and, by the orthogonality relation (4.1), integrated Legendre polynomials vanish at both endpoints of the interval $(-1, 1)$. Summarizing, one obtains

$$\hat{p}_n^\alpha(-1) = 0, \quad \hat{p}_n^0(1) = 0 \quad \text{for } n \geq 2, \alpha > -1. \tag{4.2}$$

Factoring out these roots, integrated Jacobi polynomials (3.2) can be expressed in terms of Jacobi polynomials (3.1) with modified weights, i.e.,

$$\hat{p}_n^\alpha(x) = \frac{1+x}{n} P_{n-1}^{(\alpha-1,1)}(x), \qquad n \geq 1, \tag{4.3}$$

$$\hat{p}_n^0(x) = \frac{1-x^2}{2n-2} P_{n-2}^{(1,1)}(x), \qquad n \geq 2. \tag{4.4}$$

The identity (4.4) is widely known and can be found in many introductory textbooks on high order finite element methods [26]. The other identity is less well known, hence a short computer algebra proof for (4.3) is given below. Note that this gives yet another motivation to utilize computer algebra as a tool in everyday's work. Often a particular identity that would be needed is either hard to dig out in the literature or not known at all, but with a simple application of symbolic computation algorithms it can be delivered in no time automatically.
The proof of the sparsity pattern detailed below proceeds by exact evaluation of the integrals following a rewrite procedure that turns the given integrand into a linear combination of integrals of the form (4.1), i.e., integrals where the orthogonality relation can be applied. For this process several identities relating different types of Jacobi polynomials and/or integrated Jacobi polynomials are needed. The key relations that enter the computations are summarized in Lemma 4.1 below. In [9, 6, 10] classical proofs for these identities were provided. In this note we want to stress that this type of identities need not be proven by hand, but should rather be *discovered* (and thus proven) automatically using symbolic computation. To illustrate how this can be carried out, we introduce briefly two particular packages that can handle these tasks, "SumCracker" [27] and "HolonomicFunctions" [28], both implemented in the computer algebra system Mathematica and both available for download at

`http://www.risc.jku.at/research/combinat/software/` .

There also more detailed descriptions of the algorithms as well as examples on how to apply the packages can be found.
SumCracker is a package developed by Kauers [27] containing algorithmic procedures for treating sequences that are described via certain systems of difference equations (recurrence relations). These systems can be possibly nonlinear, but with shifts in a single variable only. It can be used for proving known or conjectured identities as well as for discovering new identities. One advantage of this package as well as the package HolonomicFunctions is that they allow to enter Jacobi polynomials as symbolic expressions in standard Mathematica syntax.
In short the algorithm implemented in SumCracker proceeds by translating the given expressions that may be defined via systems of recurrence relations into polynomial form. Then using Gröbner bases computations [14] algebraic relations can be found on the polynomial level and translated back into the original setting. The scope of the package includes finding of linear recurrence relations (as a particular instance of an algebraic dependency) or closed form relations.

We illustrate these different approaches by using one easy example, the treatment of the sum $f(n) = \sum_{k=0}^{n}(2k+1)p_k^0(x)$. Firstly we want to evaluate the sum in a simple closed form. The standard Mathematica notation for Legendre polynomials $p_k^0(x)$ is LegendreP$[k, x]$ and the standard notation for a sum would be "Sum". In order to distinguish from the Mathematica built-in command in SumCracker sums are denoted by "SUM":

In[1]:= **Crack[SUM[(2k + 1)LegendreP[k, x], {k, 0, n}]]**

Out[1]= $\frac{n+1}{x-1}$(LegendreP$[n, x]$−LegendreP$[1+n, x]$) $\left(\text{in standard notation:}\quad f(n) = \frac{n+1}{x-1}\left(p_n^0(x) - p_{n+1}^0(x)\right)\right)$

By default SumCracker uses the expressions that are given in the input to express the solution. It is also possible to express the solution in different terms using the "Into" option (if such a closed form exists):

In[2]:= **Crack[SUM[(2k + 1)LegendreP[k, x], {k, 0, n}], Into → {n, JacobiP[n, 1, 0, x]}]**

Out[2]= $(1+n)$JacobiP$[n, 1, 0, x]$ $\left(\text{in standard notation:}\quad f(n) = (n+1)p_n^1(x)\right)$

Also "JacobiP" is the standard notation in Mathematica for Jacobi polynomials. As final example, we show how to obtain a recurrence relation for $f(n)$:

In[3]:= **GetLinearRecurrence[SUM[(2k + 1)LegendreP[k, x], {k, 0, n}], In → n, Head → f]**

Out[3]= $f[n+2] = \frac{1+15x+16nx+4n^2x}{(2+n)(3+2n)}f[n+1] - \frac{5+2n}{3+2n}f[n]$

In this case the result is a second order homogeneous recurrence for $f(n)$. Note again that this result is not approximate in any way but has been derived by a rigorous proving procedure. For further options and more details we refer to [27].

Jacobi polynomials (including integrated Jacobi polynomials) satisfy linear differential equations with respect to x and also linear difference equations (i.e., recurrence relations) with respect to n, α and β with polynomial coefficients. In other words, they are holonomic functions and many algorithms are known and implemented that deal with this type of objects [45, 17]. Holonomic functions are closed under several operations such as addition, multiplication (both Hadamard and Cauchy), partial summation or integration of the given sequences or functions. As a data structure for holonomic functions we use the defining difference/differential relations usual written in operator notation, where D_z denotes the standard derivative w.r.t. the unknown z and S_z denotes the forward shift in the variable z, i.e., $S_z f(z) = f(z+1)$. Note that the operators D_x, S_n commute with each other, D_x and n as well as S_n and x commute, but D_x does not commute with x neither S_n with n (i.e., they do not commute with the variables they are acting on).

Given a sequence/function that needs to be summed/integrated in the first step a set of annihilating linear operators in terms of shifts and/or derivatives with polynomial coefficients is computed. This set builds a basis for an annihilating ideal of the given input. Starting from this representation, i.e., a (system of) linear difference/differential equation(s) the defining difference/differential equation(s) for the desired sum/integral can be discovered entirely algorithmically. This procedure extends to multivariate input and covers also mixed difference-differential relations.

Koutschan implemented algorithms capable of carrying out these tasks in his Mathematica package "HolonomicFunctions" [28]. Also this package allows for a simple input structure following Mathematica notation. Annihilating operators for the given expression can be obtained, e.g., using the "Annihilator" command and the operators for shift and derivation are denoted by $S[\cdot]$ and Der$[\cdot]$, respectively. We illustrate the use of this package again with some simple examples. First we

compute the basis for an annihilating ideal for the sum over Legendre polynomials with respect to shifts in n and derivation with respect to x:

```
In[4]:= ann = Annihilator[Sum[LegendreP[k, x], {k, 0, n}], {S[n], Der[x]}]

Out[4]= {2(1 - x)^2(1 + x)Dx^2 - 2(1 + n)^2Sn - (1 - x)(3 + 2n + 7x + 2nx)Dx + (n + 2)(2n + x + 1),
        - (1 - x)SnDx - (1 + n)Sn - (1 - x)Dx + (2 + n),
        (2 + n)Sn^2 - (3 + 2n)(1 + x)Sn - 2(1 - x^2)Dx + (2 + n)(1 + 2x)}
```

With $f_n(x) = \sum_{k=0}^{n} p_k^0(x)$ the application of the output of "Annihilator" to $f_n(x)$ reads in standard notation as:

$$\begin{aligned} 2(1-x^2)(1+x)f_n''(x) - 2(1+n)^2 f_{n+1}(x) - (1-x)(3+2n+7x+2nx)f_n'(x) \\ +(n+2)(2n+x+1)f_n(x) = 0, \\ -(1-x)f_{n+1}'(x) - (1+n)f_{n+1}(x) - (1-x)f_n'(x) + (2+n)f_n(x) = 0, \\ (2+n)f_{n+2}(x) - (3+2n)(1+x)f_{n+1}(x) - 2(1-x^2)f_n'(x) + (2+n)(1+2x)f_n(x) = 0. \end{aligned}$$

The annihilator command gives a basis for all possible relations in the operators S_n, D_x. Using the "FindRelation" command specific identities can be constructed. Say we are interested in a recurrence relation for the given sum $f_n(x)$ then the command would be as follows:

```
In[5]:= FindRelation[ann, Support → {1, S[n], S[n]^2, S[n]^3}]

Out[5]= {(n + 3)Sn^3 + (-2nx - n - 5x - 3)Sn^2 + (2nx + n + 5x + 2)Sn + (-n - 2)}
```

If the support is chosen with fewer shifts then the output will be the empty set, i.e., there is no recurrence of order two in the computed annihilating ideal. Spelled out in traditional notation the above reads as

$$(n+3)f_{n+3}(x) - ((n+3) + x(2n+5))\, f_{n+2}(x) + ((n+2) + x(2n+5))\, f_{n+1}(x) - (n+2)f_n(x) = 0.$$

Keep in mind that this recurrence is the output of a proving procedure. The underlying computations are also using Gröbner bases, this time in a non-commutative setting. For further informations we refer to [28]. In particular for dealing with sums or integrals the method of creative telescoping is applied [45]. For proving (4.3) we use the corresponding HolonomicFunctions-command "CreativeTelescoping" explicitly. The identity $\hat{p}_n^{\alpha}(x) = \frac{1+x}{n} P_n^{(\alpha-1,1)}(x)$ can be easily guessed. In order to verify it, we compute recurrence relations for both sides of the identity. For the right hand side we use as input for the Jacobi polynomials the well known sum representation [3]

$$P_n^{(\alpha,\beta)}(x) = \frac{(\alpha+1)_n}{n!} \sum_{k=0}^{n} \frac{(-n)_k (n+\alpha+\beta+1)_k}{(\alpha+1)_k k!} \left(\frac{1-x}{2}\right)^k,$$

where $(a)_k = a(a+1)\cdots(a+k-1)$ denotes the Pochhammer symbol (or rising factorial). Below we replace the Mathematica built-in command "Pochhammer" by the corresponding symbol for better readability. With this we obtain for the left hand side:

```
In[6]:= annLHS = CreativeTelescoping[Annihilator[Integrate[
        (α + 1)_(n-1)/(n - 1)! (-n + 1)_k (n + α)_k/(α + 1)_k k! ((1 - y)/2)^k, {y, -1, x}], {S[k], S[n]}], S[k] - 1, S[n]][[1]]

Out[6]= {2(n + 2)(n + α + 1)(2n + α)Sn^2 - (2n + α + 1)(4n^2x + 4nxα + 4nx + xα^2 + 2xα + α^2 - 2α)Sn
        + 2n(n + α - 1)(2n + α + 2)}
```

For the right hand side it suffices to use the Annihilator command once more

```
In[7]:= annRHS = Annihilator[(1 + x)/nJacobiP[n - 1, α - 1, 1, x], {S[n]}]

Out[7]= {2(n + 2)(n + α + 1)(2n + α)S_n^2 - (2n + α + 1) (4n^2 x + 4nxα + 4nx + xα^2 + 2xα + α^2 - 2α) S_n
        + 2n(n + α - 1)(2n + α + 2)}
```

Hence both sides satisfy the same recurrence relation and checking initial values completes the proof. There is an overlap of features covered by HolonomicFunctions and the Maple implementations "Mgfun" [17, 15, 16] or "Gfun" [35].
Next we state the main relations that are needed in the evaluation of the bilinear forms.

Lemma 4.1. *Let $p_n^\alpha(x) = P_n^{(\alpha,0)}(x)$ denote the nth Jacobi polynomial and $\hat{p}_n^\alpha(x)$ the nth integrated Jacobi polynomial as defined in* (3.1)*. Then we have for $n \geq 1$*

$$p_n^{\alpha-1} = \frac{1}{2n+\alpha}[(n+\alpha)p_n^\alpha(x) - n\,p_{n-1}^\alpha(x)], \qquad \alpha > -1, \tag{4.5}$$

$$\begin{aligned} p_{n+1}^\alpha(x) = {} & \frac{2n+\alpha+1}{2(n+1)(n+\alpha+1)(2n+\alpha)}[(2n+\alpha+2)(2n+\alpha)x + \alpha^2]p_n^\alpha(x) \\ & - \frac{n(n+\alpha)(2n+\alpha+2)}{(n+1)(n+\alpha+1)(2n+\alpha)}p_{n-1}^\alpha(x), \qquad \alpha \geq -1, \end{aligned} \tag{4.6}$$

$$\begin{aligned} \hat{p}_n^\alpha(x) = {} & \frac{2(n+\alpha)}{(2n+\alpha-1)(2n+\alpha)}p_n^\alpha(x) + \frac{2\alpha}{(2n+\alpha-2)(2n+\alpha)}p_{n-1}^\alpha(x) \\ & - \frac{2(n-1)}{(2n+\alpha-1)(2n+\alpha-2)}p_{n-2}^\alpha(x), \qquad \alpha \geq -1, \end{aligned} \tag{4.7}$$

$$\hat{p}_n^\alpha(x) = \frac{2}{2n+\alpha-1}[p_n^{\alpha-1}(x) + p_{n-1}^{\alpha-1}(x)], \qquad \alpha > -1, \tag{4.8}$$

$$(\alpha-1)\hat{p}_n^\alpha(x) = (1-x)\,p_{n-1}^\alpha(x) + 2\,p_n^{\alpha-2}(x), \qquad \alpha > 1. \tag{4.9}$$

Proof. For the proofs below we utilize the packages described earlier in this section. Note that both SumCracker and HolonomicFunctions are using *forward* shifts only. Hence the formulas given by these algorithms are shifted versions of the normalized results stated in the lemma.
Proof of (4.5):

```
In[8]:= Crack[JacobiP[n + 1, α, 0, x], Into → {n, JacobiP[n, α + 1, 0, x]}]

Out[8]= -(n + 1)/(2n + α + 3) JacobiP[n, α + 1, 0, x] + (n + α + 2)/(2n + α + 3) JacobiP[n + 1, α + 1, 0, x]
```

For the proof of (4.9) we use the rewriting (4.3) of integrated Jacobi polynomials in terms of standard Jacobi polynomials in the input for SumCracker:

```
In[9]:= Crack[(1 + x)/(n + 1) JacobiP[n, α, 1, x], Into → {JacobiP[n, α + 1, 0, x], JacobiP[n, α - 1, 0, x]}]

Out[9]= (1 - x)/α JacobiP[n, α + 1, 0, x] + 2/α JacobiP[n + 1, α - 1, 0, x]
```

Note that (4.6) is merely the standard three term recurrence for Jacobi polynomials for $\beta = 0$. For the remaining identities we use HolonomicFunctions. In the first step we compute a basis for the annihilating ideal for integrated Jacobi polynomials with respect to the operators S_n, S_α, D_x and in the next step we search the ideal for relations of a certain type:

```
In[10]:= annIJ = Annihilator[(1 + x)/n JacobiP[n - 1, α - 1, 1, x], {S[n], S[α], Der[x]}]
```

Out[10]= $\{(\alpha+n)S_\alpha-(1+x)D_x-(n+\alpha-1), 2(n+1)(\alpha+n)S_n+(1-x^2)(2n+\alpha)D_x$
$-(\alpha+n-1)(\alpha+x(2n+\alpha)), (x^2-1)D_x^2+\alpha(x+1)D_x-n(\alpha+n-1)\}$

Identity (4.7) is a relation with a certain support and coefficients not depending on x:

In[11]:= **FindRelation[annIJ, Support → {S[n], S[n]^2 Der[x], S[n]Der[x], Der[x]}, Eliminate → {x}]**

Out[11]= $\{2(\alpha+n+1)(\alpha+2n)S_n^2D_x+2\alpha(\alpha+2n+1)S_nD_x$
$-(\alpha+2n)(\alpha+2n+1)(\alpha+2n+2)S_n-2n(\alpha+2n+2)D_x\}$

The missing identity (4.8) is proved completely analogously:

In[12]:= **FindRelation[annIJ, Support → {S[α], Der[x], Der[x]S[n]}]**

Out[12]= $\{2S_nD_x+(-\alpha-2n)S_\alpha+2D_x\}$

□

5 Proof of Theorem 3.2

In this section we turn to describing the algorithm that evaluates the integrals which yield the entries of the inner blocks of the local system matrices. The input for the program is a linear combination of products of u_i, v_{ij}, w_{ijk}, or partial derivatives thereof, and the output is the integral of such a function over a general tetrahedron specified by its barycentric coordinates as a linear combination of products of Kronecker deltas with rational coefficients in i, j and k. The basic idea for evaluating the integrals is to rewrite the given integrands until the orthogonality relation (4.1) for Jacobi polynomials

$$\int_{-1}^{1}\left(\frac{1-x}{2}\right)^{\alpha}p_i^{\alpha}(x)p_j^{\alpha}(x)\,\mathrm{d}x=\frac{2\delta_{ij}}{2i+\alpha+1}$$

can be used. For this rewriting procedure the identities summarized in Lemma 4.1 relating Jacobi and integrated Jacobi polynomials are needed. In theory these steps can also be carried out by hand and this has been done for the corresponding continuous basis functions for H^1 on triangles [9]. Already in the two dimensional case it is obviously a tedious task to carry out the necessary transformations. The situation becomes worse in the tetrahedral case for H^1 and hopeless if we pass to $H(\mathrm{curl})$ (or $H(\mathrm{div})$).
Whereas in the H^1 case for the mass matrix only products of $\phi_{ijk}=u_iv_{ij}w_{klm}$ need to be evaluated, which in fact can be done using paper and pencil, for $H(\mathrm{curl})$ already for the mass matrix entries partial derivatives of the basic polynomials enter, see (3.9) and (3.10). Taking the partial derivatives blows up the input significantly. Even though derivatives of Jacobi polynomials are again Jacobi polynomials with shifted parameters, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}x}P_n^{(\alpha,\beta)}(x)=\frac{n+\alpha+\beta+1}{2}P_{n-1}^{(\alpha,\beta)}(x),\tag{5.1}$$

the number of terms in the input increases beyond being suitable for hand computations. The situation becomes even worse when computing the stiffness matrix entries. The computerized method described in this paper follows the human approach.
The first step in the computations consists of decoupling integrals over a tetrahedral element using a well known substitution, the Duffy transformation which maps the tetrahedron to the unit cube. Let us illustrate this step using the reference tetrahedron depicted in Figure 1. The barycentric

coordinates for this tetrahedron are given by

$$\lambda_1(x,y,z) = \frac{-4x-2y-z+1}{8}, \qquad \lambda_2(x,y,z) = \frac{4x-2y-z+1}{8},$$
$$\lambda_3(x,yz) = \frac{2y-z+1}{4}, \qquad \lambda_4(x,y,z) = \frac{1+z}{2}.$$

The basic template integral that needs to be considered is of the form

$$\begin{aligned}&\int_{\triangle} f_1\left(\frac{4x}{1-2y-z}\right) f_2\left(\frac{2y}{1-z}\right) f_3(z)\mathrm{d}(x,y,z)\\ &= \int_{-1}^{1}\int_{-\frac{1-z}{2}}^{\frac{1-z}{2}}\int_{-\frac{1-2y-z}{4}}^{\frac{1-2y-z}{4}} f_1\left(\frac{4x}{1-2y-z}\right) f_2\left(\frac{2y}{1-z}\right) f_3(z)\mathrm{d}(x,y,z),\end{aligned}$$

where f_i are products of certain Jacobi or integrated Jacobi polynomials and Jacobi weight functions. For the Jacobi weight functions we introduce the abbreviation

$$w_\gamma(x) = \left(\frac{1-x}{2}\right)^\gamma.$$

By means of the substitutions $x \leftarrow \frac{1-2y-z}{4}x$ and $y \leftarrow \frac{1-z}{2}y$ the integrands decouple and we obtain

$$\int_{-1}^{1} f_1(x)\,\mathrm{d}x \int_{-1}^{1}\left(\frac{1-y}{2}\right) f_2(y)\,\mathrm{d}y \int_{-1}^{1}\left(\frac{1-z}{2}\right)^2 f_3(z)\,\mathrm{d}z.$$

For the computation of the matrix entries we still have a dependency of the integrands via the discrete parameters that appear as the polynomial degrees of the basis functions as well as in the polynomial parameters $\alpha(i) = 2i-1$ and $\beta(i,j) = 2i+2j-2$. This gives a natural order to carry out integration first with respect to x, then y and finally z. Note that the additional factors introduced by the Duffy substitution above are just contributing to the Jacobi weight functions.

The algorithm Let us fix the integration variable $\xi \in \{x,y,z\}$. After performing the Duffy substitution, the input in each step is an integrand that is a finite linear combination of products of Jacobi and/or integrated Jacobi polynomials and certain weight functions. For instance, if we consider the first component of the integrand $\int_{\hat{\triangle}} \varphi^{(b)}_{ijk}(x,y,z)\varphi^{(b)}_{lmn}(x,y,z)\ \mathrm{d}(x,y,z)$ over the reference tetrahedron depicted in Figure 1, then after decoupling the integrals it reads as follows,

$$\begin{aligned}\mathcal{I}_1 \quad = \quad & \int_{-1}^{1} p^0_{i-1}(x)p^0_{l-1}(x)\,\mathrm{d}x \int_{-1}^{1} \hat{p}^{2i-1}_j(y)\hat{p}^{2l-1}_m(y)w_{i+l-1}(y)\,\mathrm{d}y\\ & \times \int_{-1}^{1} \hat{p}^{2i+2j-2}_k(z)\hat{p}^{2l+2m-2}_n(z)w_{i+j+l+m}(z)\,\mathrm{d}z.\end{aligned}$$

Integrating with respect to x just requires an application of the Jacobi orthogonality relation and yields

$$\mathcal{I}_1 = \frac{2\delta_{i,l}}{2i-1}\int_{-1}^{1} \hat{p}^{2i-1}_j(y)\hat{p}^{2i-1}_m(y)w_{2i-1}(y)\,\mathrm{d}y \int_{-1}^{1} \hat{p}^{2i+2j-2}_k(z)\hat{p}^{2i+2m-2}_n(z)w_{2i+l+m}(z)\,\mathrm{d}z.$$

In the next step the integrand needs to be rewritten in terms of Jacobi polynomials using identity (4.7) in order to invoke the orthogonality relation and the result is

$$\begin{aligned}\mathcal{I}_1 &= \int_{-1}^{1}\Bigg(-\frac{16(j-1)(2i+j-3)\delta_{i,l}\delta_{j-2,m}}{(2i-1)(2i+2j-2)^{\underline{5}}}\hat{p}_k^{2i+2j-2}(z)\hat{p}_n^{2i+2j-6}(z)w_{2i+2j-2}(z)\\ &+\frac{16(2i-3)\delta_{i,l}\delta_{j-1,m}}{(2i+2j-1)^{\underline{5}}}\hat{p}_k^{2i+2j-2}(z)\hat{p}_n^{2i+2j-4}(z)w_{2i+2j-1}(z)\\ &+\frac{32(j^2+2ij-2j+4i^2-8i+3)\delta_{i,l}\delta_{j,m}}{(2i-1)(2i+2j)^{\underline{5}}}\hat{p}_k^{2i+2j-2}(z)\hat{p}_n^{2i+2j-2}(z)w_{2i+2j}(z)\\ &+\frac{16(2i-3)\delta_{i,l}\delta_{j,m-1}}{(2i+2j)^{\underline{5}}}\hat{p}_k^{2i+2j-2}(z)\hat{p}_n^{2i+2j}(z)w_{2i+2j+1}(z)\\ &-\frac{16(j+1)(j+2i-1)\delta_{i,l}\delta_{j,m-2}}{(2i-1)(2i+2j+2)^{\underline{5}}}\hat{p}_k^{2i+2j-2}(z)\hat{p}_n^{2i+2j+2}(z)w_{2i+2j+2}(z)\Bigg)\ \mathrm{d}z.\end{aligned}$$

Here $a^{\underline{k}} = a(a-1)\cdots(a-k+1)$ denotes the falling factorial. If we use $P_{\rho,\iota}(\xi), Q_{\rho,\iota}(\xi)$ to denote the generic polynomials and $w_{\gamma(\rho,\iota)}(\xi)$ to denote the weight function, then the general form of the integrand is

$$\text{int} = \sum_{\rho,\iota} C_{\rho,\iota} P_{\rho,\iota}(\xi) Q_{\rho,\iota}(\xi) w_{\gamma(\rho,\iota)}(\xi),$$

where $C_{\rho,\iota}$ are rational functions in the degree parameters and Kronecker deltas, but independent of ξ. If $P_{\rho,\iota}, Q_{\rho,\iota}$ are Jacobi polynomials of the same kind and the $w_{\gamma(\rho,\iota)}$ the corresponding weight function, then an evaluation of

$$\int_{-1}^{1} P_{\rho,\iota}(\xi) Q_{\rho,\iota}(\xi) w_{\gamma(\rho,\iota)}(\xi)\ \mathrm{d}\xi$$

amounts to a mere application of the Jacobi orthogonality relation - as in the first step in the example above. In all other cases the polynomials need to be expressed as linear combinations of these base cases such that the orthogonality applies. This is achieved by applying various transformations of Lemma 4.1 in a systematic manner. This rewriting procedure is carried out automatically and the algorithm proceeds as follows:

1. Collect the list of integrands and their coefficients
2. For each entry in the list of integrands, rewrite the integrated Jacobi polynomials in terms of Jacobi polynomials using (4.7), (4.8), or (4.9)
3. Update the list of integrands and their coefficients
4. For each entry in the list of integrands, adjust Jacobi polynomials to the appearing weight function $w_{\gamma(\rho,\iota)}(\xi)$
5. Update the list of integrands and their coefficients
6. For each entry in the list of integrands, evaluate the integral using orthogonality relation (4.1)

If the rewriting procedure returns expressions that do not fit to the evaluation pattern in step 6 a warning is issued and these terms remain unevaluated. The two steps of the algorithm that need further explanations are steps 2 and 4. In step 2 it needs to be decided which of the relations (4.7)–(4.9) leads to a proper integrand. Hence for each integrand in the list, we consider the weight function $w_\gamma(\xi)$ and each of the polynomials P_ρ, Q_ι separately. If one of them is an integrated Jacobi polynomial $\hat{p}_n^\alpha(\xi)$ then

2. rewrite $w_\gamma(\xi)\hat{p}_n^\alpha(\xi)$ in terms of Jacobi polynomials by

   (a) $\gamma-\alpha \geq 0$: transforming integrated Jacobi polynomials to Jacobi polynomials with same parameter using (4.7).

   (b) $\gamma - \alpha = -1$: transforming integrated Jacobi polynomials to Jacobi polynomials with parameter $\alpha - 1$ using (4.8).

   (c) $\gamma - \alpha = -2$: using the mixed relation (4.9) to obtain

$$w_\gamma(\xi)\hat{p}_n^{\gamma+2}(\xi) = \frac{2}{\gamma+1}\left(w_\gamma(\xi)p_n^\gamma(\xi) + w_{\gamma+1}(\xi)p_{n-1}^{\gamma+2}(\xi)\right).$$

In step 4 the list of polynomials consists of classical Jacobi polynomials only. In order to make them fit to the orthogonality relation (4.1) the polynomial parameter α needs to be adjusted to the given weight $w_\gamma(\xi)$. This adjustment with fixed, finite support can only be done by increasing the polynomial parameter using (4.5). If the situation is such that $\gamma < \alpha$, then the term has to be returned unevaluated, since Jacobi polynomials $p_n^\alpha(\xi)$ cannot be expressed as a fixed, finite number of linear combinations of $p_n^\gamma(\xi)$. This step in the computations is potentially a very expensive one as it might increase the number of terms significantly. The transformation (in the case $\gamma - \alpha > 0$) can be stated explicitly as

$$p_n^\alpha(\xi) = \sum_{m=0}^{\gamma-\alpha}(-1)^k \begin{pmatrix} \gamma-\alpha \\ m \end{pmatrix} \frac{(n+\gamma-m)^{\underline{\gamma-\alpha-m}}\, n^{\underline{m}}}{(2n+\gamma-m+1)^{\underline{\gamma-\alpha+1}}}(2n-2m+\gamma+1)p_{n-m}^\gamma(\xi),$$

where $a^{\underline{k}} = a(a-1)\cdots(a-k+1)$ again denotes the falling factorial.
The final output after execution of the program is a linear combination of Kronecker deltas with rational coefficients. Below we give a concrete example. Table 1 summarizes the number of integrals that need to be evaluated in each step for the different products of basis functions. The last column is to be understood as the number of different triples $\delta_{i+i',l}\delta_{j+j',m}\delta_{k+k',n}$ in the result. Note that for some integrands the number of integrals blows up significantly in the integration with respect to z just to collapse to a smaller number again in the final result. This is due to a high number of integrands of the form $p_{k+k'}^\gamma(\xi)p_{n+k'}^\gamma(\xi)w_\gamma(\xi)$ that give rise to a single $\delta_{k,n}$ in the end.

An example To illustrate the execution of the algorithm we choose only a small example since the basis functions for $H(\mathrm{curl})$ written explicitly are too large to be displayed here. The integrand we consider (already after decoupling of the variables) is

$$\mathcal{I}_{ijk;lmn}(x,y,z) = \hat{p}_i^0(x)p_l^0(x)w_{i+l+1}(y)\hat{p}_j^{2i}(y)\hat{p}_m^{2l-1}(y)w_{i+l+j+m+2}(z)\hat{p}_k^{2i+2j-1}(z)\hat{p}_n^{2l+2m-1}(z).$$

Already this seemingly simple integrand gives rise to a linear combination of 176 different products of Kronecker deltas. The first integration is with respect to x. The weight function is obviously 1, i.e., $\gamma = 0$. For $p_l^0(x)$ no rewriting is necessary and for $\hat{p}_i^0(x)$ Case 2(a) applies and we rewrite in terms of Legendre polynomials $p_i^0(x)$ using (4.7). This way for the first integral we obtain

$$\int_{-1}^{1} \hat{p}_i^0(x)p_l^0(x)\,\mathrm{d}x = \frac{2\delta_{i,l}}{(2i-1)(2i+1)} - \frac{2\delta_{i,l+2}}{(2i-3)(2i-1)}.$$

The final output of the algorithm is of this form, i.e., rational functions in the polynomial degrees times a product of Kronecker deltas relating (i,l), (j,m) and (k,n), respectively. The evaluation of the integral with respect to x assigns values to l in the remaining polynomials. If we continue with the first term above, then the remaining integrand after replacing $l \leftarrow i$ reads as

$$w_{2i+1}(y)\hat{p}^{2i}(y)\hat{p}_m^{2i-1}(y)w_{2i+j+m+2}(z)\hat{p}_k^{2i+2j-1}(z)\hat{p}_n^{2i+2m-1}(z).$$

int	x	y	z	NNZ
$\varphi^{(b)}_{ijk}\varphi^{(b)}_{lmn}$	9	216	287	369
$\varphi^{(b)}_{ijk}\tilde{\varphi}^{(a)}_{1mn}$	6	82	470	198
$\varphi^{(b)}_{ijk}\tilde{\varphi}^{(b)}_{lmn}$	6	99	206	252
$\varphi^{(b)}_{ijk}\tilde{\varphi}^{(c)}_{lmn}$	6	151	237	369
$\tilde{\varphi}^{(a)}_{1jk}\tilde{\varphi}^{(a)}_{1mn}$	9	25	381	117
$\tilde{\varphi}^{(a)}_{1jk}\tilde{\varphi}^{(b)}_{lmn}$	4	27	333	154
$\tilde{\varphi}^{(a)}_{1jk}\tilde{\varphi}^{(c)}_{lmn}$	4	42	343	187
$\tilde{\varphi}^{(b)}_{ijk}\tilde{\varphi}^{(b)}_{lmn}$	4	25	115	135
$\tilde{\varphi}^{(b)}_{ijk}\tilde{\varphi}^{(c)}_{lmn}$	4	58	140	252
$\tilde{\varphi}^{(c)}_{ijk}\tilde{\varphi}^{(c)}_{lmn}$	4	67	165	243

int	x	y	z	NNZ
$\tilde{\varphi}^{(a)}_{1jk}\tilde{\varphi}^{(a)}_{1mn}$	–	107	287	81
$\tilde{\varphi}^{(a)}_{1jk}\tilde{\varphi}^{(b)}_{lmn}$	4	78	246	98
$\tilde{\varphi}^{(a)}_{1jk}\tilde{\varphi}^{(c)}_{lmn}$	6	78	165	98
$\tilde{\varphi}^{(b)}_{ijk}\tilde{\varphi}^{(b)}_{lmn}$	4	59	71	75
$\tilde{\varphi}^{(b)}_{ijk}\tilde{\varphi}^{(c)}_{lmn}$	6	62	55	100
$\tilde{\varphi}^{(c)}_{ijk}\tilde{\varphi}^{(c)}_{lmn}$	9	65	35	125

Table 1: Number of integrals to be evaluated for the mass (left) and stiffness matrix (right) w.r.t. the different variables and final number of nonzero entries

Proceeding with the integration with respect to y following the algorithm we first transform integrated Jacobi polynomials into Jacobi polynomials with same parameter using (4.7). Then we need to lift the polynomial parameter to adjust to $\gamma = 2i+1$ using (4.5) once, respectively twice. This results in an output of a linear combination of $\delta_{j,m-4}, \delta_{j,m-3}, \ldots, \delta_{j,m+3}$. Each of these evaluations defines a product of polynomials in z with m replaced by the corresponding expression in j. For instance for $m = j$ the remaining integrand is

$$w_{2i+2j+2}(z)\hat{p}_k^{2i+2j-1}(z)\hat{p}_n^{2i+2j-1}(z).$$

The series of transformations is as before and results in a linear combination of $\delta_{k,m-4}, \ldots, \delta_{k,m+4}$. For $m = j-4$ the integrand is

$$w_{2i+2j-2}(z)\hat{p}_k^{2i+2j-1}(z)\hat{p}_n^{2i+2j-9}(z).$$

Here we run into Case 2(b) in the rewriting of the first polynomial $\hat{p}_k^{2i+2j-1}(z)$ in order to be able to match the appearing weight function.
If we define the variable "integrand" as $\mathcal{I}_{ijk;lmn}$, then the evaluation using our program executes precisely the above mentioned steps:

In[13]:= **eval = ComputeMatrixEntries[integrand, x, y, z, Infolevel → True];**

1. Collecting integrands depending on x
→ finished collecting (0.004 sec.)
→ 1 integrands
2. Rewriting integrated Jacobi polynomials in terms of Jacobi polynomials
Case 2(a) for phat[i, 0, x]
→ finished rewriting (0.004001 sec.)
3. Collecting integrands depending on x
→ finished collecting (0.012 sec.)
→ 2 integrands
6. Evaluate integrals using Jacobi orthogonality relation
→ finished evaluating (0.004001 sec.)
1. Collecting integrands depending on y

→ finished collecting (0.004 sec.)
→ 2 integrands
2. Rewriting integrated Jacobi polynomials in terms of Jacobi polynomials
Case 2(b) for phat[j, 2 i, y]
Case 2(a) for phat[m, -5 + 2 i, y]
Case 2(a) for phat[j, 2 i, y]
Case 2(a) for phat[m, -1 + 2 i, y]
→ finished rewriting (0.192012 sec.)
3. Collecting integrands depending on y
→ finished collecting (0. sec.)
→ 15 integrands
4. Adjusting Jacobi polynomials to appearing weight functions
→ finished adjusting (0.148009 sec.)
6. Evaluate, 34 integrals using Jacobi orthogonality relation
→ finished evaluating (0.060004 sec.)
1. Collecting integrands depending on z
→ finished collecting (0.644041 sec.)
→ 8 integrands
2. Rewriting integrated Jacobi polynomials in terms of Jacobi polynomials
Case 2(a) for phat[k, -1 + 2 i + 2 j, z]
...
Case 2(b) for phat[n, 7 + 2 i + 2 j, z]
→ finished rewriting (2.09613 sec.)
3. Collecting integrands depending on z
→ finished collecting (0.348022 sec.)
→ 69 integrands
4. Adjusting Jacobi polynomials to appearing weight functions
→ finished adjusting (12.4128 sec.)
6. Evaluate, 244 integrals using Jacobi orthogonality relation
→ finished evaluating (1.22408 sec.)

These computations were carried out on an 8 cores (2.33GHz), 16 Gb shared memory machine. Concerning the implementation of the algorithm one crucial step is to normalize both the input and output. Most of the steps rely on pattern matching and thus the representation needs to be uniform. E.g., a polynomial $\hat{p}_k^{2(i+j)}(z)$ has to be recognized to be the same as $\hat{p}_k^{2i+2j}(z)$. This becomes particularly tricky when collecting the different weight functions that might appear as factors $(y-1)$ that need to included in the standard notation for weights $w_\gamma(y) = \left(\frac{1-y}{2}\right)^\gamma$. The transformation to standard form is done as first step before the evaluation starts and for large integrands that are assembled from different partial derivations this can become a very costly step. Another computational issue is related to simplification of the coefficients. Certainly no undetected zeros should be carried along, not only for memory reasons, but also since the evaluation with finite support may rely on cancellation to happen. The final outcome of the program is neither fully simplified nor factored. The matrix entries that have been put online however are in simplified form, which was done in a post processing step that added significantly to the overall computation times.

The basis polynomials $\tilde{\varphi}^{(a)}_{1jk}(x,y,z)$ are only linear in x. Thus for the direct product $\tilde{\varphi}^{(a)}_{1jk} \cdot \tilde{\varphi}^{(a)}_{1mn}$ (or corresponding derivatives), it is handy to carry out the integration with respect to x beforehand separately. For mixed products with other basis polynomials the constant coefficient with respect to x of $\tilde{\varphi}^{(a)}_{1jk}$ is replaced by $p_0^0(x) = 1$ times this coefficient. Analogously the linear term x is replaced by $p_1^0(x)$.

This concludes the proof.

Acknowledgement: This work has been supported by the FWF-projects P20121-N12, P20162-N18, P23484-N18, and the doctoral program "Computational Mathematics" (W1214).

References

[1] M. Abramowitz and I. Stegun, editors. *Handbook of mathematical functions.* Dover-Publications, 1965.

[2] M. Ainsworth and J. Coyle. Hierarchic finite element bases on unstructured tetrahedral meshes. *Int. J. Num. Meth. Eng.*, 58(14):2103–2130, 2003.

[3] G.E. Andrews, R. Askey, and R. Roy. *Special Functions.* Encyclopedia of Mathematics and its Applications 71. Cambridge University Press, 2000.

[4] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$. *Numer. Math.*, 85(2):197–217, 2000.

[5] I. Babuška and B. Q. Guo. The *h-p* version of the finite element method for domains with curved boundaries. *SIAM J. Numer. Anal.*, 25(4):837–861, 1988.

[6] S. Beuchler and V. Pillwein. Shape functions for tetrahedral *p*-fem using integrated Jacobi polynomials. *Computing*, 80:345–375, 2007.

[7] S. Beuchler, V. Pillwein, J. Schöberl, and S. Zaglmayr. Sparsity optimized high order finite element functions on simplices. In Ulrich Langer and Peter Paule, editors, *Numerical and Symbolic Scientific Computing: Progress and Prospects*, pages ?–? Springer, Wien, 2011.

[8] S. Beuchler, V. Pillwein, and S. Zaglmayr. Sparsity optimized high order finite element functions for H(div) on simplices. Technical Report 2010-04, DK Computational Mathematics, JKU Linz, 2010.

[9] S. Beuchler and J. Schöberl. New shape functions for triangular *p*-fem using integrated jacobi polynomials. *Numer. Math.*, 103:339–366, 2006.

[10] Sven Beuchler and Veronika Pillwein. Completions to sparse shape functions for triangular and tetrahedral p-fem. In U. Langer, M. Discacciati, D.E. Keyes, O.B. Widlund, and W. Zulehner, editors, *Domain Decomposition Methods in Science and Engineering XVII*, volume 60 of *Lecture Notes in Computational Science and Engineering*, pages 435–442, Heidelberg, 2008. Springer. Proceedings of the 17th International Conference on Domain Decomposition Methods held at St. Wolfgang / Strobl, Austria, July 3–7, 2006.

[11] A. Bondeson, T. Rylander, and P. Ingelström. *Computational electromagnetics*, volume 51 of *Texts in Applied Mathematics.* Springer, New York, 2005.

[12] A. Bossavit. *Computational Electromagnetism: Variational Formulations, Complementary, Edge Elements.* Academic Press, San Diego, Calif., 1998.

[13] D. Braess. The convergence rate of multigrid with Gauss-Seidel relaxation for the Poisson equation. In W. Hackbusch and U. Trottenberg, editors, *Multigrid methods, Proceedings of the Conference held at Köln-Porz, November 23-27, 1981*, number 960 in Lecture notes in mathematics, pages 368–386, Berlin-Heidelberg-New York, 1982. Springer Verlag.

[14] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal)*. PhD thesis, Mathematical Institute, University of Innsbruck, Austria, 1965. English translation in J. of Symbolic Computation, Special Issue on Logic, Mathematics, and Computer Science: Interactions. Vol. 41, Number 3-4, Pages 475–511, 2006.

[15] F. Chyzak. *Fonctions holonomes en calcul formel*. Thèse universitaire, École polytechnique, 1998. INRIA, TU 0531. 227 pages.

[16] F. Chyzak. Groebner bases, symbolic summation and symbolic integration. In B. Buchberger and F. Winkler, editors, *Groebner Bases and Applications (Proc. of the Conference 33 Years of Gröbner Bases)*, volume 251 of *London Mathematical Society Lecture Notes Series*, pages 32–60. Cambridge University Press, 1998. ISBN 0-521-63298-6.

[17] F. Chyzak. An extension of Zeilberger's fast algorithm to general holonomic functions. *Discrete Math.*, 217(1-3):115–134, 2000. Formal power series and algebraic combinatorics (Vienna, 1997).

[18] P. Ciarlet. *The Finite Element Method for Elliptic Problems*. North–Holland, Amsterdam, 1978.

[19] L. Demkowicz. *Computing with hp Finite Elements*. CRC Press, Taylor and Francis, 2006.

[20] L. Demkowicz and A. Buffa. H^1, $H(\mathrm{curl})$ and $H(\mathrm{div})$-conforming projection-based interpolation in three dimensions. Quasi-optimal p-interpolation estimates. *Comput. Methods Appl. Mech. Engrg.*, 194(2-5):267–296, 2005.

[21] Leszek Demkowicz, Jason Kurtz, David Pardo, Maciej Paszyński, Waldemar Rachowicz, and Adam Zdunek. *Computing with hp-adaptive finite elements. Vol. 2*. Chapman & Hall/CRC Applied Mathematics and Nonlinear Science Series. Chapman & Hall/CRC, Boca Raton, FL, 2008. Frontiers: three dimensional elliptic and Maxwell problems with applications.

[22] J. Gopalakrishnan, L.E. García-Castillo, and L.F. Demkowicz. Nédélec spaces in affine coordinates. *Comput. Math. Appl.*, 49(7-8):1285–1294, 2005.

[23] R. Hiptmair. Multigrid method for Maxwell's equations. *SIAM J. Numer. Anal.*, 36(1):204–225, 1999.

[24] R. Hiptmair. Finite elements in computational electromagnetism. *Acta Numer.*, 11:237–339, 2002.

[25] P. Ingelström. A New Set of H(curl)-Conforming Hierarchical Basis Functions for Tetrahedral Meshes. *IEEE Transactions on Microwave Theory and Techniques*, 54(1):106–114, 2006.

[26] G.M. Karniadakis and S.J.Sherwin. *Spectral/HP Element Methods for CFD*. Oxford University Press. Oxford, 1999.

[27] M. Kauers. SumCracker – A Package for Manipulating Symbolic Sums and Related Objects. *Journal of Symbolic Computation*, 41(9):1039–1057, 2006.

[28] C. Koutschan. HolonomicFunctions (User's Guide). Technical Report 10-01, RISC Report Series, University of Linz, Austria, January 2010.

[29] L. D. Landau and E. M. Lifschitz. *Lehrbuch der theoretischen Physik ("Landau-Lifschitz"). Band VIII.* Akademie-Verlag, Berlin, fifth edition, 1990. Elektrodynamik der Kontinua. [Electrodynamics of continua], Translated from the second Russian edition by Georg Dautcourt, Helmut Günther, Horst Heino v. Borzeskowski [Horst-Heino von Borzeszkowski] and Stefan-Ludwig Drechsler, Translation edited by Gerd Lehmann, With the collaboration of Lifschitz and L. P. Pitajewski [L. P. Pitaevskiĭ], With a foreword by P. Ziesche and Lehmann.

[30] Peter Monk. *Finite element methods for Maxwell's equations.* Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 2003.

[31] J.-C. Nédélec. A new family of mixed finite elements in $\mathbf{R}^3$. *Numer. Math.*, 50(1):57–81, 1986.

[32] J.C. Nédélec. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35(35):315–341, 1980.

[33] A. Quateroni and A. Valli. *Numerical Approximation of partial differential equations.* Number 23 in Springer Series in Computational Mathematics. Springer. Berlin-Heidelberg-New York, 1997.

[34] S. Reitzinger and J. Schöberl. An algebraic multigrid method for finite element discretizations with edge elements. *Numer. Linear Algebra Appl.*, 9(3):223–238, 2002.

[35] B. Salvy and P. Zimmermann. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2):163–177, 1994.

[36] C. Schneider. Symbolic Summation Assists Combinatorics. *Sem. Lothar. Combin.*, 56:1–36, 2007.

[37] J. Schöberl and S. Zaglmayr. High order Nédélec elements with local complete sequence properties. *COMPEL*, 24(2), 2005.

[38] C. Schwab. *p− and hp−finite element methods. Theory and applications in solid and fluid mechanics.* Clarendon Press. Oxford, 1998.

[39] B. Szabo and I. Babuška. *Finite Element Analysis.* Wiley, 1991.

[40] G. Szegö. *Orthogonal Polynomials.* AMS Colloquium Publications, Volume XXIII. American Mathematical Society, 3 edition, 1974.

[41] F.G. Tricomi. *Vorlesungen über Orthogonalreihen.* Springer. Berlin-Göttingen-Heidelberg, 1955.

[42] J.P. Webb. Hierarchal Vector Basis Functions of Arbitrary Order for Triangular and Tetrahedral Finite Elements. *IEEE Transactions on Antennas and Propagation*, 47(8):1244–1253, 1999.

[43] K. Wegschaider. Computer Generated Proofs of Binomial Multi-Sum Identities. Master's thesis, RISC, J. Kepler University, May 1997.

[44] S. Zaglmayr. *High Order Finite Elements for Electromagnetic Field Computation.* PhD thesis, Johannes Kepler University, Linz, Austria, 2006.

[45] D. Zeilberger. A holonomic systems approach to special functions identities. *J. Comput. Appl. Math.*, 32(3):321–368, 1990.