Numerical solution of the Poisson equation over hypercubes using reduced Chebyshev polynomial bases

Christophe De Luigi, Sylvain Maire, Serge Dumont

Numerical Solution Of The Poisson Equation Over Hypercubes Using Reduced Chebyshev Polynomial Bases

C. De Luigi [a], S. Dumont [b], S. Maire [a,*]

[a] *Université du Sud Toulon-Var, I.S.I.T.V., Avenue G. Pompidou BP 56, F-83162 La Valette du Var CEDEX, FRANCE*

[b] *LAMFA, Faculté de Mathématiques et d'Informatique Université de Picardie Jules Verne, rue Saint Leu, F-80039 Amiens CEDEX 1, FRANCE*

Abstract

We describe how to use new reduced size polynomial approximations for the numerical solution of the Poisson equation over hypercubes. Our method is based on a non-standard Galerkin method which allows tests functions which do not verify the boundary conditions. Numerical examples are given in dimensions up to 8 on solutions with different smoothness using the same approximation basis for both situations. A special attention is paid on conditioning problems.

Key words: Reduced size polynomial approximation, Poisson equation on hypercubes, Hybrid variational formulation

1991 MSC: 65C05, 65D15, 65D32, 65N35

1 Introduction

The aim of this paper is to use a sparse Chebyshev polynomial basis [10] to obtain, at a reasonable computational cost, very accurate approximations of the solution of the Poisson equation over hypercubes. These approximations are computed by means of an hybrid variational formulation [4]. We pay a special attention to numerical comparisons between our sparse basis and the standard tensor product one.

* Corresponding author
Email addresses: deluigi@univ-tln.fr (C. De Luigi), Serge.Dumont@u-picardie.fr (S. Dumont), maire@univ-tln.fr (S. Maire).

Spectral methods [2,3] have been developed for a wide range of partial differential equations. They rely on tensor product approximations on several polynomial bases. They are especially efficient for smooth solutions on simple geometries. In the case of the Poisson equation in dimension Q over an hypercube $D = [-1,1]^Q$, the most commonly used bases are the Legendre and the Chebyshev polynomial ones. Even in such favourable situations, the complexity of the spectral methods increases quickly with the dimension Q as the number of unknowns using the collocation method with tensor product approximations of degree N becomes $(N+1)^Q$. Moreover for a general partial differential equation one has to solve a plain linear symmetric system with a complexity of $O((N+1)^{2Q})$ using for example an iterative method like the conjugate gradient method with or without preconditioning. In the case of the Poisson equation, it is possible to reduced this complexity to $O(Q(N+1)^{Q+1})$ for either Legendre [17,18] or Chebyshev polynomials. The idea is to write the approximation on a basis where the spectral matrix is sparse.

Anyhow it is worth considering approximations on different kinds of reduced size bases to solve these equations in order to attenuate the dimensional effect. We can for example mention the work from Von Petersdorf and Schwab [16] where sparse piecewise approximations are used to solve parabolic equations in high dimensions. Concerning the approximations of periodic smooth functions on $Q-$dimensional Fourier bases, Korobov spaces [6] have been introduced. They rely on a decay of the Fourier coefficients a_m as

$$|a_m| \leq \frac{C}{(\widetilde{m}_1 \widetilde{m}_2 \cdots \widetilde{m}_Q)^\alpha}$$

where $\widetilde{m} = \max(1, |m|)$. The constant C and the parameter $\alpha > 1$ are linked to the smoothness of the integrand. The natural choice is to keep only the coefficients belonging to the set

$$\left\{ m \in \mathbb{Z}^Q / \, (\widetilde{m}_1 \cdots \widetilde{m}_Q) \leq d \right\}$$

where d is the level of approximations. Unfortunately one has to transform the original integrand using the periodization method to achieve such a decay for non-periodic functions. This periodization has a very bad effect on the constant C which grows very quickly with α. Another difficulty is the numerical computation of the coefficients which is handled using lattice rules in the purpose of numerical integration [5,19]. In the case of polynomial approximations, Novak and Ritter [14] have developed a method to integrate polynomials such that their total degree is below a given value. Their method performs very well for dimensions $Q \geq 8$ and the integrand does not need to be periodic.

We have introduced in [8] polynomial spaces very similar to Korobov ones in a sense that the basis functions are chosen in a multidimensional Chebyshev polynomial basis according to a criterion based on the product of the degree

of the polynomials of each variable. This basis has the same good properties than the previous Fourier basis but avoids the periodization problems. Our criterion is also more selective compared to the one developed in [14] and is directly linked to the regularity of the function to approximate. To compute the coefficients of an approximation on this basis, we have first used a sequential Monte Carlo algorithm [8] which was modified and improved by using quasi-Monte Carlo sequences [9]. The use of Chebyshev polynomials combined with random drawings associated to the Chebyshev weight were crucial in these works. We have finally replaced the algorithm by the least square problem of fitting our Chebyshev polynomial approximation model to some random, quasi-random points or quantified points [10]. This has led for instance to high accurate quadrature formulae, but our concern here is numerical approximations. In section 2, we summarize this method and give some numerical examples on various functions up to dimension 10 which will be the solutions of the Poisson equations of sections 3.

We describe in this last section a numerical method to use our basis for the numerical solution of the Poisson equation over an hypercube. This method is a variational formulation introduced in [4] which allows test functions not verifying the boundary conditions. Numerical tests are given on solutions with different smoothness and dimensions. We also introduce a new criterion to select our basis functions in order to reduce the condition number of the spectral matrix.

2 The approximation method

2.1 Description of the approximation

The Chebyshev polynomials $T_n(x) = \cos(n \arccos(x))$ are the orthogonal polynomials with respect to the inner product $< P, Q >= \int_{-1}^{1} \frac{P(x)Q(x)}{\sqrt{1-x^2}} dx$. They verify the differential equation

$$\frac{d}{dx}\left(\sqrt{1-x^2}T_n'(x)\right) + n^2 \frac{T_n(x)}{\sqrt{1-x^2}} = 0.$$

Using this equation, one can show that if $f \in C^{2L}([-1,1])$ the coefficients b_n of its mean-square approximation on the Chebyshev polynomials verify $|b_n| \leq \frac{C}{n^{2L}}$, where C is a constant depending on f and L. The multidimensional interpolation polynomial $P_N(f)$ at the points $y_i = \cos\left(\frac{2i+1}{N+1}\frac{\pi}{2}\right)$ of

the Chebyshev grid writes

$$P_N(f) = \sum_{i_1=0}^{N} \sum_{i_2=0}^{N} \cdots \sum_{i_Q=0}^{N} \alpha_{i_1,i_2,\cdots,i_Q} T_{i_1}(x_1) T_{i_2}(x_2) \cdots T_{i_Q}(x_Q)$$

where the $\alpha_{i_1,\cdots,i_Q}$ are defined by

$$\alpha_{i_1,\cdots,i_Q} = \frac{\pi^Q}{\prod_{j=1}^{Q} \left\| T_{i_j} \right\|_2^2 (N+1)^Q} \sum_{j_1=0}^{N} \cdots \sum_{j_Q=0}^{N} f(y_{j_1}, \cdots, y_{j_Q}) T_{i_1}(y_{j_1}) \cdots T_{i_Q}(y_{j_Q}).$$

Furthermore, standard approximation results [2] show that

$$\left\| f - P_N(f) \right\|_2 \leq \frac{C}{N^{2L}}$$

meaning that this approximation is really accurate especially when f is very smooth. However, this kind of approximation is very sensitive to the dimensional effect as its complexity is a $O(N^Q)$. Hence when the dimension Q increases one needs to consider other types of polynomial approximation which can also take advantage of the smoothness of the function f but with a smaller complexity. Letting $\widehat{m} = \max(1, m)$, we have proved in [8] that the coefficients b_m of the mean-square approximation in dimension Q verify, for another constant C_1,

$$|b_m| \leq \frac{C_1}{(\widehat{m}_1 \widehat{m}_2 \cdots \widehat{m}_Q)^{2L}}$$

still using the differential equation satisfied by the T_n for the Q integration variables. We can then give the approximation

$$f(x_1, \cdots, x_Q) = \sum_{m \in W_{Q,d}} b_m T_{m_1}(x_1) T_{m_2}(x_2) \cdots T_{m_Q}(x_Q) + r(x_1, \cdots, x_Q)$$

where the set $W_{Q,d} = \left\{ m \in \mathbb{N}^Q / \, (\widehat{m}_1 \cdots \widehat{m}_Q) \leq d \right\}$ corresponds to a level d of approximation and $r(x_1, \cdots, x_Q)$ is the error term.
Some values of $L_{Q,d} = \text{card}(W_{Q,d})$ are given in Table 1.

Some theoretical results are described in [8] which can be summarized by the control on $\int_D r(t)^2 dt$. Under the previous assumptions, $\forall \varepsilon > 0$ there is a constant $C_{Q,\varepsilon}$ depending only on Q and ε such that

$$\left\| r \right\|_2 \leq \frac{C_{Q,\varepsilon}}{d^{2L-0.5-\varepsilon}}.$$

The approximation of the function f writes

$$f(x_1, x_2, \cdots, x_Q) \simeq \sum_{m \in W_{Q,d}} \widetilde{b}_m T_{m_1}(x_1) T_{m_2}(x_2) \cdots T_{m_Q}(x_Q).$$

d	$L_{2,d}$	$l_{3,d}$	$L_{4,d}$	$L_{5,d}$	$L_{6,d}$
1	4	8	16	32	64
2	8	20	48	112	256
3	12	32	80	192	448
5	21	62	168	432	1072
7	31	98	280	752	1936
10	48	165	504	1432	3872
15	76	276	880	2592	7232

Table 1
Complexity of the approximation *w.r.t.* d and Q

We compute the $L_{Q,d}$ coefficients b_k belonging to $W_{Q,d}$ using a program which tests if $m_1 m_2 ... m_Q \leq d$ and stores the values of these parameters in Q lists $l_1(k), l_2(k) \cdots l_Q(k)$. We also define $c_1(k) = 1_{l_1(k) \geq 1}, \cdots, c_Q(k) = 1_{l_Q(k) \geq 1}$ which occur in the following normalizations. As we create this lists, we also store in another vector l the unique value of k corresponding to $l_1(k), l_2(k) \cdots l_Q(k)$. We now write

$$f(x_1, x_2, \cdots, x_Q) \simeq \sum_{k=1}^{L_{Q,d}} \widetilde{b}_k T_{l_1(k)}(x_1) T_{l_2(k)}(x_2) \cdots T_{l_Q(k)}(x_Q).$$

2.2 The least-square problem

We describe quickly the least-square method developed in [10] to compute the coefficients b_k. For the sake of normalization, we use a least-square problem weighted by the norms of the basis functions

$$J_1 = \frac{1}{M} \sum_{i=1}^{M} \left(\sum_{k=1}^{L_{Q,d}} \widetilde{s}_k \sqrt{2}^{\sum_{n=1}^{Q} c_n(k)} \prod_{n=1}^{Q} T_{l_n(k)}(X_n^{(i)}) - f(X_1^{(i)}, \cdots, X_Q^{(i)}) \right)^2$$

with

$$\widetilde{s}_k = \frac{\widetilde{b}_k}{\sqrt{2}^{\sum_{n=1}^{Q} c_n(k)}}.$$

The idea is to fit our approximation model to its observation at M data points of coordinates $(X_1^{(i)}, \cdots, X_Q^{(i)})$, $i = 1, \cdots, M$.
The minimization of J_1 is equivalent to the resolution of the system $B\widetilde{s} = q$ with

$$B_{k,j} = \frac{\sqrt{2}^{\sum_{n=1}^{Q} c_n(k) + c_n(j)}}{M} \sum_{i=1}^{M} \prod_{n=1}^{Q} T_{l_n(k)}(X_n^{(i)}) T_{l_n(j)}(X_n^{(i)})$$

and

$$q_k = \frac{\sqrt{2}^{\sum_{n=1}^{Q} c_n(k)}}{M} \sum_{i=1}^{M} \prod_{n=1}^{Q} T_{l_n(k)}(X_n^{(i)}) f(X_1^{(i)}, \cdots, X_Q^{(i)}).$$

The accuracy of the approximation depends on the condition number of the least square matrix B using the inequality $\|s - \widetilde{s}\| \leq \|B\| \, \|B^{-1}\| \, \frac{\|r\|}{\|q\|} \, \|s\|$. The relative error in quadratic norm is

$$\frac{\|s - \widetilde{s}\|_2}{\|s\|_2} \leq \|B\|_2 \left\|B^{-1}\right\|_2 \frac{\sqrt{C_{Q,\varepsilon}}}{\|q\|_2 \, d^{2L-0.5-\varepsilon}}.$$

If the data points $X^{(i)}$ are random variables with density

$$w(x) = \prod_{i=1}^{Q} \frac{1}{\pi\sqrt{1 - x_i^2}} 1_{[-1,1]}(x_i)$$

then the coefficients $B_{k,j}$ go to δ_{kj} with M because these coefficients are integrals computed by means of a Monte Carlo method. The speed of convergence toward the identity matrix is bounded by $\frac{C}{\sqrt{N}}$. The use of Chebyshev polynomials enables an uniform bound of this speed independent of d and Q that is $C \leq 1$. This crucial property is a straightforward consequence of the fact that these polynomials are uniformly bounded by 1. It has been observed for example in [8] that the constant C increases very quickly with d and Q when using Legendre polynomial basis. As mentioned in the introduction, it can be efficient to replace this Monte Carlo approximation by an approximation using Quasi-Monte Carlo sequences [7,12,19] as their rate of convergence for numerical integration is a $O\left(\frac{\ln(N)^{Q-1}}{N}\right)$. Another similar point of view is to find the best way to represent with M points the density $w(x)$ according to some criterion. Using the competitive learning vector quantization algorithm [15], we have computed the M points in D minimizing the functional

$$J(M) = \min \left(\int_D \inf_{1 \leq i \leq M} |x - x_i|^2 \, w(x) dx : \{x_1, x_2 \cdots x_M \in D\} \right).$$

We have made some tests on pseudo-random linear generator, Halton sequences, Sobol sequences and points built from optimal quadratic quantization. On some basic examples in dimension 3, the quantization points appeared to be the most efficient just before the Halton sequences. However, these points are difficult to build in practice and one can also choose an hybrid strategy to build the quadrature points: make some steps of the competitive learning vector quantization algorithm initialized by Halton sequences.

The first possiblity to compute the coefficients $\widetilde{b}_k$ of the approximation

$$f(x_1, x_2, \cdots, x_Q) \simeq \sum_{k=1}^{L_{Q,d}} \widetilde{b}_k T_{l_1(k)}(x_1) T_{l_2(k)}(x_2) \cdots T_{l_Q(k)}(x_Q)$$

is to solve the linear system $B\widetilde{s} = q$ which has to be done for each different function. A cheaper way to do it is to store the Cholesky factorization of the matrix B once and for all. An even more efficient way is to build quadrature formulae for the numerical approximation of all the coefficients $\widetilde{b}_k$ and as for the numerical approximation $\widetilde{I}(f)$ of

$$I(f) = \int\limits_{[-1,1]^Q} f(x)dx.$$

We first compute numerically the inverse matrix B^{-1} and we write $s = B^{-1}q$ to obtain

$$\widetilde{s}_k = \sum_{j=1}^{L_{Q,d}} B^{-1}_{kj} q_j = \sum_{j=1}^{L_{Q,d}} B^{-1}_{kj} \frac{\sqrt{2}^{\sum_{n=1}^{Q} c_n(j)}}{M} \sum_{i=1}^{M} \prod_{n=1}^{Q} T_{l_n(k)}(X_n^{(i)}) f(X_1^{(i)}, \cdots, X_Q^{(i)})$$

that is

$$\widetilde{s}_k = \sum_{i=1}^{M} \sum_{j=1}^{L_{Q,d}} B^{-1}_{jk} \frac{\sqrt{2}^{\sum_{n=1}^{Q} c_n(j)}}{M} \prod_{n=1}^{Q} T_{l_n(k)}(X_n^{(i)}) f(X_1^{(i)}, \cdots, X_Q^{(i)}).$$

Then, we have

$$\widetilde{b}_k = \sum_{i=1}^{M} \lambda_{i,k} f(X_1^{(i)}, \cdots, X_Q^{(i)})$$

with

$$\lambda_{i,k} = \sqrt{2}^{\sum_{n=1}^{Q} c_n(k)} \sum_{j=1}^{L_{Q,d}} B^{-1}_{jk} \frac{\sqrt{2}^{\sum_{n=1}^{Q} c_n(j)}}{M} \prod_{n=1}^{Q} T_{l_n(k)}(X_n^{(i)}).$$

If we are interested in numerical integration, we finally have

$$\widetilde{I}(f) = \sum_{k=1}^{L_{Q,d}} \widetilde{b}_k \prod_{n=1}^{Q} \int_{-1}^{1} T_{l_n(k)}(x)dx = \sum_{i=1}^{M} \alpha_i f(X_1^{(i)}, \cdots, X_Q^{(i)})$$

with

$$\alpha_i = \sum_{k=1}^{L_{Q,d}} \lambda_{i,k} \prod_{n=1}^{Q} \int_{-1}^{1} T_{l_n(k)}(x)dx.$$

We have observed no significant difference between the two numerical integration procedures on some numerical tests. Hence we use the quadrature

formulae for the computations of all the coefficients $\widetilde{b}_k$ of the approximation and for $\widetilde{I}(f)$. This means that the coefficients $\lambda_{i,k}$ and α_i are computed and stored once and for all.

2.4 Numerical results

In this section, we give some numerical examples of approximations of functions in dimension 3 to 5 which will be the analytical solutions of the Poisson equations studied in section 3. This will show the accuracy of our approximation method and will also allow to check the efficiency of our method of resolution of the partial differential equations. As a first example, we use the functions

$$f_1(x) = \exp\left(\frac{1}{Q}\sum_{i=1}^{Q} x_i\right)$$

which are obviously very smooth. As a second example, we build less regular functions f_2 having an exact degree 2 of smoothness. In dimension one, f_2 is the cubic spline approximation of the function $\cos(\frac{x}{2})$ on $[-1,1]$ at 7 equidistant points. In dimension Q, the functions f_2 are built using tensor products of this spline. We now give some numerical examples from dimension 3 to 5 letting respectively the absolute errors $e_h(\frac{1}{2})$, $e_h(0)$, $e_h(I)$ for a function h at the reference points $(\frac{1}{2},\cdots,\frac{1}{2}),(0,\cdots,0)$ and on the integral over the domain $[-1,1]^Q$. The $L_{Q,d}$ coefficients are computed using $[2.5 \times L_{Q,d}]$ points built from Halton sequences. This particular choice was shown to be both robust and cheap in [10].

The results for $Q = 3$ for the sparse basis and the tensor product basis are given respectively in Table 2 and Table 3. The approximations of both functions are really accurate. We achieve for example an accuracy of 10 digits

d	$L_{3,d}$	$e_{f_1}(\frac{1}{2})$	$e_{f_1}(0)$	$e_{f_1}(I)$	$e_{f_2}(\frac{1}{2})$	$e_{f_2}(0)$	$e_{f_2}(I)$
3	32	2×10^{-4}	2×10^{-3}	1×10^{-3}	9×10^{-4}	1×10^{-2}	7×10^{-4}
5	62	8×10^{-5}	1×10^{-5}	3×10^{-5}	8×10^{-4}	2×10^{-5}	8×10^{-5}
7	98	1×10^{-5}	3×10^{-5}	5×10^{-6}	1×10^{-4}	9×10^{-4}	1×10^{-5}
10	165	2×10^{-7}	4×10^{-7}	1×10^{-8}	1×10^{-4}	6×10^{-5}	2×10^{-6}
15	276	2×10^{-7}	6×10^{-8}	2×10^{-8}	3×10^{-5}	2×10^{-5}	4×10^{-6}
30	700	1×10^{-10}	3×10^{-10}	4×10^{-11}	4×10^{-6}	6×10^{-7}	7×10^{-8}
60	1702	6×10^{-14}	2×10^{-13}	4×10^{-14}	6×10^{-7}	3×10^{-8}	3×10^{-9}

Table 2
Numerical approximation sparse basis $Q = 3$

on the approximation of f_1 and 6 digits on the approximation of f_2 when $d = 30$. The corresponding number of basis functions is 700. As a comparison, we also give the same kind of results using the approximation on the tensor product Chebyshev interpolation polynomials of degree N with matrix size $S_N = (N+1)^3$.

N	S_N	$e_{f_1}(\frac{1}{2})$	$e_{f_1}(0)$	$e_{f_1}(I)$	$e_{f_2}(\frac{1}{2})$	$e_{f_2}(0)$	$e_{f_2}(I)$
3	64	1×10^{-4}	9×10^{-4}	5×10^{-4}	4×10^{-3}	3×10^{-2}	2×10^{-2}
5	216	3×10^{-7}	3×10^{-7}	2×10^{-7}	1×10^{-3}	2×10^{-3}	6×10^{-4}
7	512	1×10^{-10}	2×10^{-10}	1×10^{-10}	1×10^{-3}	1×10^{-3}	3×10^{-4}
10	1331	3×10^{-14}	5×10^{-14}	2×10^{-14}	5×10^{-4}	3×10^{-4}	7×10^{-5}

Table 3
Numerical approximation tensor basis $Q = 3$

We observe that the approximation results are slightly more accurate for the function f_1 but really less accurate for the function f_2. For example, when $N = 7$ which corresponds to 512 basis functions, we achieve an accuracy of 10 digits on the approximation of f_1 but only 3 digits on the approximation of f_2. This means that the reduced basis is less sensitive to the smoothness of the functions than the usual tensor product Chebyshev interpolation. This has been already observed in [10] and as our basis is also a lot less sensitive to the dimensional effect, we only keep it for higher dimensions.

We now turn to dimension 4, see Table 4. Once again, we obtain a good

d	$L_{4,d}$	$e_{f_1}(\frac{1}{2})$	$e_{f_1}(0)$	$e_{f_1}(I)$	$e_{f_2}(\frac{1}{2})$	$e_{f_2}(0)$	$e_{f_2}(I)$
3	80	4×10^{-4}	1×10^{-3}	2×10^{-4}	1×10^{-2}	2×10^{-2}	1×10^{-3}
5	168	8×10^{-6}	4×10^{-6}	5×10^{-6}	8×10^{-4}	8×10^{-4}	6×10^{-4}
7	280	2×10^{-6}	1×10^{-5}	5×10^{-6}	1×10^{-3}	8×10^{-4}	3×10^{-4}
10	504	1×10^{-7}	1×10^{-7}	2×10^{-7}	2×10^{-4}	6×10^{-5}	2×10^{-5}
15	880	3×10^{-8}	4×10^{-8}	1×10^{-9}	5×10^{-5}	6×10^{-5}	1×10^{-5}
30	2453	2×10^{-10}	3×10^{-10}	1×10^{-12}	6×10^{-6}	3×10^{-6}	4×10^{-7}

Table 4
Numerical approximation sparse basis $Q = 4$

accuracy on the approximations. The number of basis functions $L_{4,d}$ is about 3 times greater than $L_{3,d}$.

Finally in dimension 5, we give the results in Table 5. We still obtain a good accuracy on the approximations. The complexity of the approximation $L_{5,d}$ is about 3 times greater than $L_{4,d}$.

d	$L_{5,d}$	$e_{f_1}(\frac{1}{2})$	$e_{f_1}(0)$	$e_{f_1}(I)$	$e_{f_2}(\frac{1}{2})$	$e_{f_2}(0)$	$e_{f_2}(I)$
2	112	7×10^{-3}	4×10^{-3}	1×10^{-3}	2×10^{-2}	3×10^{-2}	4×10^{-2}
3	192	9×10^{-5}	4×10^{-4}	8×10^{-4}	5×10^{-3}	3×10^{-2}	1×10^{-2}
5	432	2×10^{-5}	9×10^{-6}	8×10^{-6}	5×10^{-3}	2×10^{-3}	2×10^{-3}
7	752	1×10^{-6}	5×10^{-6}	6×10^{-6}	2×10^{-3}	3×10^{-3}	7×10^{-4}
10	1432	6×10^{-8}	4×10^{-8}	5×10^{-9}	3×10^{-4}	4×10^{-5}	8×10^{-5}
15	2592	5×10^{-8}	1×10^{-8}	4×10^{-9}	7×10^{-5}	2×10^{-5}	2×10^{-5}

Table 5
Numerical approximation sparse basis $Q = 5$

3 The hybrid Galerkin Formulation

3.1 Introduction

The basis functions considered in our approximations do not verify automatically the boundary conditions. In order to use them as test functions anyway, we adopt here an hybrid variational formulation which was proposed in [4] following the ideas of the Nitsche method [1,13,20]. This hybrid variational formulation for the Poisson equation

$$-\triangle u = f$$

in a domain $D \subset \Re^d$ with boundary conditions $u = g$ on $\Gamma = \partial D$ writes

$$\int_D \nabla u.\nabla v dx - \int_\Gamma (\frac{\partial u}{\partial n} v + \frac{\partial v}{\partial n} u) ds = \int_D f v dx - \int_\Gamma g \frac{\partial v}{\partial n} ds$$

where v is a test function. In the original method, a penalization term of the form $r \int_\Gamma uv ds$ is added to the left handside of the variational formulation in order to enforce the coercivity of the symmetric bilinear form. It is shown in [4] that in the case of elliptic problems, it is not necessary to add this penalization term to obtain the uniqueness and the convergence of the solution of the discretized problem. However, the stiffness matrix is no longer positive but is still inversible. We can point out that for test functions vanishing on the boundary, this formulation is exactly the same as the classical one. This method has already been used with various test functions like finite elements which do not respect the shape of the boundary or with wavelets but yet only on problems written in $D \subset \Re^d$ with $d \leq 3$. Moreover, there is no lost of accuracy of using this method, unlike a technique using a penalized Galerkin formulation for example.

The test functions used here are product of Chebyshev polynomials of each variables. We solve the Poisson equation using different kinds of approximations based on these test functions. We first describe the variational formulations based on these approximations. Finally, we introduce additional approximation spaces to overcome part of the bad conditioning problems which appear when the size of the approximation space increases.

3.2 Description of the variational formulations

We first make in detail the description of this formulation using different polynomial basis in the case of Dirichlet homogeneous boundary conditions on cubes that is $g = 0$ and Γ on $D = [-1, 1]^Q$ for $Q = 1, 2, 3$. Then, we describe shortly how to extend this method to higher dimensions and to more general boundary conditions.

3.2.1 The one-dimensional case

In dimension one, the approximation of the solution writes

$$u_N(x) = \sum_{k=0}^{N} \alpha_k T_k(x)$$

where the coefficients α_k are solutions of the $N + 1$ equations

$$\sum_{k=0}^{N} \alpha_k (b_{k,j} + \gamma_{k,j}) = \beta_j$$

with

$$\beta_j = \int_{-1}^{1} f(x) T_j(x) dx,\ b_{k,j} = \int_{-1}^{1} T_k^{'}(x) T_j^{'}(x) dx$$

and

$$\gamma_{k,j} = T_k(-1)T_j^{'}(-1) + T_j(-1)T_k^{'}(-1) - T_k(1)T_j^{'}(1) - T_j(1)T_k^{'}(1).$$

As $T_j(1) = 1, T_j(-1) = (-1)^j, T_j^{'}(1) = j^2, T_j^{'}(-1) = (-1)^{j+1} j^2$, we have

$$\gamma_{k,j} = \left((-1)^{k+j+1} - 1\right)(j^2 + k^2).$$

Letting $a_{k,j} = b_{k,j} + \gamma_{k,j}$, we have to solve the linear system $A\alpha = \beta$. As the matrix is not positive and definite, a LU factorisation appears as a good choice for the resolution. The coefficients $b_{k,j}$ can be computed exactly and stored once and for all. The approximations of the coefficients β_j are computed using

quadrature formulae described in section 2. More precisely, we first obtain an approximation of the function of the form

$$f(x) \simeq \sum_{k=0}^{N} c_k T_k(x)$$

where the c_k are obtained via quadrature formulae. These quadratures can either be our quadratures or the usual quadratures based on Gauss points. Then we have

$$\beta_j \simeq \int_{-1}^{1} \sum_{k=0}^{N} c_k T_k(x) T_j(x) dx \simeq \sum_{k=0}^{N} c_k s_{k,j}$$

with

$$s_{k,j} = \int_{-1}^{1} T_k(y) T_j(y) dy.$$

The coefficients $b_{k,j}$ and $s_{k,j}$ can be computed exactly and stored once and for all. They are also useful in higher dimensions.

3.2.2 The bidimensional case

We describe how to use the variational formulation on the reduced basis. The approximation of the solution writes

$$u_d(x,y) = \sum_{k=1}^{L_{2,d}} \alpha_k T_{l_1(k)}(x) T_{l_2(k)}(y)$$

where the two lists $l_1(k)$ and $l_2(k)$ are used to locate to which basis functions the $L_{2,d}$ coefficients α_k belonging to $W_{2,d}$ correspond to. These coefficients α_k are solutions of the $L_{2,d}$ equations

$$\sum_{k=1}^{L_{2,d}} \alpha_k (\theta_{k,j} + \gamma_{k,j}) = \beta_j$$

with

$$\beta_j = \int_{-1}^{1} \int_{-1}^{1} f(x,y) T_{l_1(j)}(x) T_{l_2(j)}(y) dx dy.$$

and

$$\theta_{k,j} = b_{l_1(k),l_1(j)} s_{l_2(k),l_2(j)} + s_{l_1(k),l_1(j)} b_{l_2(k),l_2(j)}.$$

We now compute the last term

$$\gamma_{k,j} = \int_{\Gamma_D} \frac{\partial T_{l_1(k)}(x) T_{l_2(k)}(y)}{\partial n} T_{l_1(j)}(x) T_{l_2(j)}(y) + \frac{\partial T_{l_1(j)}(x) T_{l_2(j)}(y)}{\partial n} T_{l_1(k)}(x) T_{l_2(k)}(y) \; ds$$

which is more complicated. We write $\Gamma_D = \cup_{i=1,4}\Gamma_i$ where the Γ_i are the 4 parts of the boundaries starting with $\Gamma_1 = [-1,1] \times (-1)$ and so on. The integrals on the boundaries are respectively equal to

$$\gamma_{k,j}^{(1)} = -\left(T_{l_2(k)}(-1)T'_{l_2(j)}(-1) + T'_{l_2(k)}(-1)T_{l_2(j)}(-1)\right) \int_{-1}^{1} T_{l_1(j)}(x)T_{l_1(k)}(x)dx$$

that is

$$\gamma_{k,j}^{(1)} = -s_{l_1(k),l_1(j)}(-1)^{l_2(k)+l_2(j)+1}\left(l_2(j)^2 + l_2(k)^2\right),$$

then

$$\gamma_{k,j}^{(2)} = s_{l_1(k),l_1(j)}\left(l_2(j)^2 + l_2(k)^2\right), \ , \gamma_{k,j}^{(3)} = s_{l_2(k),l_2(j)}\left(l_1(j)^2 + l_1(k)^2\right),$$

$$\gamma_{k,j}^{(4)} = -s_{l_2(k),l_2(j)}(-1)^{l_1(k)+l_1(j)+1}\left(l_1(j)^2 + l_1(k)^2\right)$$

and finally

$$\gamma_{k,j} = -\gamma_{k,j}^{(1)} + \gamma_{k,j}^{(2)} + \gamma_{k,j}^{(3)} - \gamma_{k,j}^{(4)}.$$

The coefficients β_j are computed using the quadrature formulae of section 2. The approximation of f is

$$f_d(x,y) = \sum_{k=1}^{L_{2,d}} c_k T_{l_1(k)}(x)T_{l_2(k)}(y)$$

and hence

$$\beta_j \simeq \sum_{k=1}^{L_{2,d}} c_k s_{l_1(k),l_1(j)} s_{l_2(k),l_2(j)}$$

that is

$$\beta_j \simeq \sum_{k=1}^{L_{2,d}} \sum_{i=1}^{[2.5\times L_{2,d}]} \lambda_{i,k} f(X_1^{(i)}, X_2^{(i)}) s_{l_1(k),l_1(j)} s_{l_2(k),l_2(j)}$$

where the weights $\lambda_{i,k}$ and the points $(X_1^{(i)}, X_2^{(i)})$ have been defined in section 2. We can use the same methodology for the formulation based on tensor product approximations of degree N. The size of the approximation space is $(N+1)^2$ instead of $L_{2,d}$, two new lists are created to locate the basis functions corresponding to each coefficients of the approximation and the β_j are computed using Gauss-Chebyshev product rules.

3.2.3 The tridimensional case

We describe only the approximation of the solution on the sparse basis which writes

$$u_d(x,y) = \sum_{k=1}^{L_{3,d}} \alpha_k T_{l_1(k)}(x)T_{l_2(k)}(y)T_{l_3(k)}(z).$$

The coefficients α_k are solutions of the $L_{3,d}$ equations

$$\sum_{k=1}^{L_{3,d}} \alpha_k(\theta_{k,j} + \gamma_{k,j}) = \beta_j$$

with

$$\beta_j = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(x,y,z)T_{l_1(j)}(x)T_{l_2(j)}(y)T_{l_3(j)}(z)dxdydz$$

and

$$\begin{aligned}\theta_{k,j} = & b_{l_1(k),l_1(j)}s_{l_2(k),l_2(j)}s_{l_3(k),l_3(j)} + b_{l_2(k),l_2(j)}s_{l_1(k),l_1(j)}s_{l_3(k),l_3(j)} + \\ & b_{l_3(k),l_3(j)}s_{l_1(k),l_1(j)}s_{l_2(k),l_2(j)}.\end{aligned}$$

The coefficient $\gamma_{k,j}$ is now a sum of 6 terms corresponding to each side of the cube $[-1,1]^3$. We have

$$\begin{aligned}\gamma_{k,j}^{(1)} &= -s_{l_1(k),l_1(j)}s_{l_3(k),l_3(j)}(-1)^{l_2(k)+l_2(j)+1}\left(l_2(j)^2 + l_2(k)^2\right),\\ \gamma_{k,j}^{(2)} &= s_{l_1(k),l_1(j)}s_{l_3(k),l_3(j)}\left(l_2(j)^2 + l_2(k)^2\right),\\ \gamma_{k,j}^{(3)} &= s_{l_2(k),l_2(j)}s_{l_3(k),l_3(j)}\left(l_1(j)^2 + l_1(k)^2\right),\\ \gamma_{k,j}^{(4)} &= -s_{l_2(k),l_2(j)}s_{l_3(k),l_3(j)}(-1)^{l_1(k)+l_1(j)+1}\left(l_1(j)^2 + l_1(k)^2\right),\\ \gamma_{k,j}^{(5)} &= s_{l_1(k),l_1(j)}s_{l_2(k),l_2(j)}\left(l_3(j)^2 + l_3(k)^2\right),\\ \gamma_{k,j}^{(6)} &= -s_{l_1(k),l_1(j)}s_{l_2(k),l_2(j)}(-1)^{l_3(k)+l_3(j)+1}\left(l_3(j)^2 + l_3(k)^2\right)\end{aligned}$$

and finally

$$\gamma_{k,j} = -\gamma_{k,j}^{(1)} + \gamma_{k,j}^{(2)} + \gamma_{k,j}^{(3)} - \gamma_{k,j}^{(4)} + \gamma_{k,j}^{(5)} - \gamma_{k,j}^{(6)}.$$

The coefficients β_j are computed in the same way than in dimension 2.

3.2.4 Extension to general problems

The extension of the method to problems in dimension Q with Dirichlet homogeneous boundary conditions is easy. The coefficient $\theta_{k,j}$ is a sum of Q terms with first term equal to

$$b_{l_1(k),l_1(j)}\prod_{i=2}^{Q} s_{l_i(k),l_i(j)},$$

the coefficient $\gamma_{k,j}$ is a sum of $2Q$ terms with first 2 terms equal to

$$-\prod_{i\neq 2}^{Q} s_{l_i(k),l_i(j)}(-1)^{l_2(k)+l_2(j)+1}\left(l_2(j)^2 + l_2(k)^2\right)$$

and

$$\prod_{i\neq 2}^{Q} s_{l_i(k),l_i(j)} \left(l_2(j)^2 + l_2(k)^2 \right)$$

and the coefficients β_j are computed using quadrature formulae in dimension Q. In the case of more general boundary conditions, we also have to compute the terms

$$\int_{\Gamma_N} gvds - \int_{\Gamma_D} u_0 \frac{\partial v}{\partial n} ds.$$

We assume for the sake of simplicity that Γ_N and Γ_D are constituted of faces of the hypercube $[-1,1]^Q$. We use the same method than for the computation of the β_j. We compute approximations $\widetilde{g}$ and $\widetilde{u}_0$ of the functions g and u_0 on spaces of size $L_{Q-1,d}$ and then we integrate exactly the products $\widetilde{g}v$ or $\widetilde{u}_0 \frac{\partial v}{\partial n}$. This last computation leads to integrate terms of the form

$$\int_{-1}^{1} T_k(x) T_j'(x) dx$$

which are computed and stored once and for all.

3.3 Numerical results

We first study equations in dimension 3 with either a very smooth solution f_1 or a less smooth solution f_2. In Table 6, the solutions are computed at the two reference points and we denote by $\kappa(A)$ the condition number of the matrix A. Until $d = 15$, the condition number $\kappa(A)$ is relatively small and the

d	$L_{3,d}$	$e_{f_1}(\frac{1}{2})$	$e_{f_1}(0)$	$e_{f_2}(\frac{1}{2})$	$e_{f_2}(0)$	$\kappa(A)$
3	32	2×10^{-4}	1×10^{-3}	7×10^{-3}	3×10^{-2}	16
5	62	8×10^{-5}	1×10^{-5}	2×10^{-3}	2×10^{-4}	58
7	98	5×10^{-6}	2×10^{-5}	6×10^{-4}	1×10^{-3}	147
10	165	7×10^{-8}	1×10^{-7}	4×10^{-5}	3×10^{-5}	790
15	276	2×10^{-8}	3×10^{-8}	2×10^{-5}	7×10^{-5}	1111
30	700	4×10^{-10}	5×10^{-9}	1×10^{-4}	6×10^{-5}	2.8×10^{7}
60	1702	7×10^{-5}	2×10^{-4}	2×10^{7}	2×10^{7}	7.1×10^{32}

Table 6
Numerical solution sparse basis $Q = 3$

approximate solution is as accurate as the expansion of the exact solution on the same approximation basis. When $d = 30$ and even more when $d = 60$, $\kappa(A)$

becomes too large which perturbates the solutions. It is well-known [2] that approximations on polynomials of high degree have a bad impact on $\kappa(A)$. In order to diminish $\kappa(A)$, we change the approximation spaces by using an additional test which keeps only basis functions of maximal degree of each monomial equal to 10. We denote by $L'_{3,d}$ the size of this space and we build quadrature formulae and approximations relative to this space using the least-square method of section 2.

As a comparison, we study in Table 7 the previous example for $d = 15, 30, 60$. The condition number has really decreased and is now only equal to 40015

d	$L'_{3,d}$	$e_{f_1}(\frac{1}{2})$	$e_{f_1}(0)$	$e_{f_2}(\frac{1}{2})$	$e_{f_2}(0)$	$\kappa(A)$
15	216	2×10^{-8}	3×10^{-8}	3×10^{-4}	7×10^{-4}	409
30	400	4×10^{-10}	5×10^{-9}	2×10^{-4}	1×10^{-4}	3430
60	643	3×10^{-12}	1×10^{-11}	3×10^{-5}	2×10^{-5}	40015

Table 7
Numerical solution sparse basis $Q = 3$: New criterion

when $d = 60$. This new criterion allows to take larger values of d without having bad conditioning problems especially for smooth solutions. Furthermore, the number of unknowns has also decreased significantly. In Table 8, we look at tensor product approximations. No bad conditioning problems occur here

N	$(N+1)^3$	$e_{f_1}(\frac{1}{2})$	$e_{f_1}(0)$	$e_{f_2}(\frac{1}{2})$	$e_{f_2}(0)$	$\kappa(A)$
3	64	1×10^{-4}	9×10^{-4}	4×10^{-3}	4×10^{-2}	29
5	216	3×10^{-7}	3×10^{-7}	3×10^{-3}	2×10^{-3}	80
7	512	1×10^{-10}	2×10^{-10}	1×10^{-3}	1×10^{-3}	71
10	1331	3×10^{-14}	5×10^{-14}	6×10^{-4}	3×10^{-5}	145

Table 8
Numerical solution tensor basis $Q = 3$

as we have only taken values of $d \leq 10$. As noticed in section 2, the accuracy on the smooth solutions is good but the accuracy on less smooth solutions is really worse than with the sparse basis.
Moreover, the complexity of this method increases quickly with Q so we no longer use it in the last two examples in dimension 4 and 5 whose results are described in Table 9.
The approximate solutions are as accurate as the expansion of the exact solutions on the same approximation basis as we have taken here only small values of d. Nevertheless these small values are sufficient to obtain accuracies of 8 or 9 digits on the smooth solutions and 4 or 5 digits on the less smooth solutions.

d	$L_{4,d}$	$e_{f_1}(\frac{1}{2})$	$e_{f_1}(0)$	$e_{f_2}(\frac{1}{2})$	$e_{f_2}(0)$	$\kappa(A)$
3	80	1×10^{-4}	7×10^{-4}	7×10^{-3}	5×10^{-2}	44
5	168	3×10^{-5}	1×10^{-5}	2×10^{-3}	1×10^{-3}	260
7	280	2×10^{-6}	1×10^{-5}	9×10^{-4}	2×10^{-3}	666
10	504	5×10^{-9}	5×10^{-8}	3×10^{-5}	4×10^{-5}	6007
d	$L_{5,d}$	$e_{f_1}(\frac{1}{2})$	$e_{f_1}(0)$	$e_{f_2}(\frac{1}{2})$	$e_{f_2}(0)$	$\kappa(A)$
3	192	5×10^{-5}	5×10^{-4}	9×10^{-3}	6×10^{-2}	126
5	432	1×10^{-5}	4×10^{-6}	4×10^{-3}	3×10^{-3}	1267
7	752	8×10^{-7}	5×10^{-6}	1×10^{-3}	5×10^{-3}	3273
10	1432	2×10^{-9}	3×10^{-8}	2×10^{-4}	2×10^{-4}	3.53×10^5

Table 9
Numerical solution sparse basis $Q = 4, 5$

3.4 Higher dimensions and complexity of the method

In the case of the smooth solutions or on even smoother solutions, we can still use this method in higher dimensions. Indeed small values of d are suffcient to obtain a good approximation and the size of the spectral matrix is still not too large. We performed all our computations using Matlab© on a Transtec 1001L (2 Intel Xeon Dual Core 5150 2.667GhZ) with 16Gb (DDR2 FB667Mhz) of memory.
Our main problem was to built the quadratures used to compute the coefficients β_j. For $d = 5$, because of memory problems we were not able to go further than $Q = 8$ due to the size of the least-square matrix. In dimension $Q = 8$, we obtain the following errors on the function f_1 on the new function f_3 defined by $f_3(x) = f_1(x/2)$ which is obviously smoother. We obtain a very

d	$L_{8,d}$	$e_{f_1}(\frac{1}{2})$	$e_{f_1}(0)$	$e_{f_3}(\frac{1}{2})$	$e_{f_3}(0)$	$\kappa(A)$
2	1280	2×10^{-3}	2×10^{-4}	2×10^{-4}	1×10^{-5}	5975
3	2304	1×10^{-4}	2×10^{-4}	5×10^{-5}	1×10^{-5}	4260
4	5120	4×10^{-7}	1×10^{-6}	2×10^{-8}	2×10^{-8}	1.3×10^5
5	6144	2×10^{-6}	1×10^{-6}	2×10^{-8}	8×10^{-8}	3.3×10^6

Table 10
Numerical solution sparse basis $Q = 8$

good accuracy for these very small values of d especially for the function f_3. Note that in the case of a tensor product approximation, with $d = 5$, the size of the matrix would have been $6^8 = 1679616$ instead of 6144 with our method. The complexity of the method depends mainly on the computation

of the coefficients β_j and on the resolution of the linear system which both are a $O(L_{Q,d}^3)$ as we use a direct method to solve this linear system. The CPU time of resolution for $d = 10$ in dimension 4 was 0.52 seconds, 4.2 seconds for $d = 10$ in dimension 5 and 185 seconds for $d = 5$ in dimension 8. Most of the CPU time is spent building the spectral matrix. We do not count in this CPU times the time of construction of the quadratures which we consider as preprocessing. They can still be reduced by some more preprocessing for the computation of the matrix coefficients or by using another method of resolution of the linear system. We could for example compute the coefficients β_j writing

$$\beta_j \simeq \sum_{i=1}^{[2.5\times L_{Q,d}]} \mu_{i,j} f(X_1^{(i)}, \cdots, X_Q^{(i)})$$

with

$$\mu_{i,j} \simeq \sum_{k=1}^{L_{Q,d}} \lambda_{i,k} s_{l_1(k),l_1(j)} \cdots s_{l_Q(k),l_Q(j)}$$

and by storing the coefficients $\mu_{i,j}$.

4 Conclusion

We have described how to combine an hybrid variational formulation and a sparse polynomial basis for the numerical approximations of the Poisson equation over hypercubes. The same formulation is used for solutions with different smoothness. This method has provided a very accurate approximation of the solution whenever it is smooth on problems in dimensions up to 8. This method can certainly be still efficient in higher dimensions if the solution is very smooth. Part of the bad conditioning problems which happen with this method when the size of the sparse basis increases have been handled by adding another criterion on the choice of the basis functions. An other idea to reduced the condition number of the matrix is to use the stochastic spectral formulation introduced in [11]. One can certainly combine our method and the method developped in [17,18] in order to obtain a sparse and reduced size spectral matrix. Concerning less smooth solutions, one can also think of using piecewise sparse polynomial approximations in order to diminish the condition number of the linear system.

References

[1] I. BABUSKHA, U. BANERJEE, J.E. OSBORN, Survey of meshless methods and generalized finite elements methods: a unified approach, Acta Numerica, pp 1-125, 2003.

[2] C. BERNARDI, Y. MADAY, Approximations spectrales de problèmes aux limites elliptiques, Springer-Verlag, 1992.

[3] C. CANUTO, M. Y. HUSSAINI, A. QUARTERONI, T. A. ZANG, Spectral methods in fluid dynamics, Springer-Verlag, 1988.

[4] S. DUMONT, O. GOUBET, T. HA-DUONG, P. VILLON, Mesh free methods and boundary conditions, Int. J. Numer. Meth. Engng., Vol 67, pp 989-1011, 2006.

[5] S. JOE, I. SLOAN, Imbedded lattice rules for multidimensional integration, SIAM J. Numer. Anal. Vol.29, no. 4 pp. 1119-1135, 1992.

[6] A. R. KROMMER, C. W. UEBERHUBER. Computational integration. SIAM, 1998.

[7] C. LÉCOT, F. EL KHETTABI, Quasi-Monte Carlo simulation of diffusion. Dagstuhl Seminar on Algorithms and Complexity for continuous problems (1998), Journal of complexity 15, no.3, pp. 342-359, 1999.

[8] S. MAIRE, An iterative computation of approximations on Korobov-like spaces, Journal of Computational and Applied Mathematics, 157, pp. 261-281, 2003.

[9] S. MAIRE, Polynomial Approximations of multivariate smooth functions from quasi-random data, Statistics and Computing, 14, pp. 333-336, 2004.

[10] S. MAIRE, C. DE LUIGI, Quasi-Monte Carlo quadratures for multivariate smooth functions, Applied Numerical mathematics, 56, pp. 146-162, 2006.

[11] S. MAIRE, E. TANRE, Some new simulation schemes for the evaluation of Feynman-Kac representations, Monte Carlo Methods and Applications, 14-1, pp. 29-51, 2008.

[12] H. NIEDERREITER, Quasi-Monte Carlo methods and pseudorandom numbers, Bull. Amer. Math. Soc. 84, pp. 957-1041, 1978.

[13] J. A. NITSCHE, Convergence of non conforming methods, Proc. Sympos. Math. Res. Center, Univ. Wisconsin, Madison, pp 15-53, 1974.

[14] E. NOVAK, K. RITTER, High dimensional integration of smooth functions over cubes, Numerishe Mathematik, 75, pp.79-97, 1996.

[15] G. PAGES, A space vector quantization for numerical Integration, Journal of computational and applied mathematics, 89, pp. 1-38, 1997.

[16] T. VON PETERSDORFF, C. SCHWAB, Numerical solution of parabolic equations in high dimensions, M2AN Math. Model. Numer. Anal. 38, no.1, pp. 93-127, 2004.

[17] J. SHEN, Efficient Spectral-Galerkin Method I: Direct Solvers for the Second and Fourth Order Equations Using Legendre Polynomials, SIAM J. Sci. Comput. Vol. 15,No. 6, pp. 1489-1505, 1994.

[18] J. SHEN, Efficient Spectral-Galerkin Method II: Direct Solvers for the Second and Fourth Order Equations Using Chebyshev Polynomials, SIAM J. Sci. Comput. Vol. 16,No. 1, pp. 74-87, 1995.

[19] I . H. SLOAN, P. J. KACHOYAN, Lattice methods for multiple integration: Theory, error analysis and examples, SIAM J. Numer. Anal. 24, pp. 116-128, 1987.

[20] R. STENBERG, On some techniques for approximating boundary conditions in the finite element method, Journal of Computational and applied Mathematics, 63, pp 139-148, 1995.