Václav Finěk; Martina Šimůnková

Parallel implementation of Wavelet-Galerkin method

In: Jan Chleboun and Karel Segeth and Jakub Šístek and Tomáš Vejchodský (eds.): Programs and Algorithms of Numerical Mathematics, Proceedings of Seminar. Dolní Maxov, June 3-8, 2012. Institute of Mathematics AS CR, Prague, 2013. pp. 69–74.

Persistent URL: http://dml.cz/dmlcz/702693

Programs and Algorithms of Numerical Matematics 16
J. Chleboun, K. Segeth, J. Šístek, T. Vejchodský (Eds.)
Institute of Mathematics AS CR, Prague 2013

PARALLEL IMPLEMENTATION OF WAVELET-GALERKIN METHOD

Václav Finěk, Martina Šimůnková

KAP and KMD FP TU Liberec
Studentská 1402/2, 461 17 Liberec 1, Czech Republic
vaclav.finek@tul.cz, martina.simunkova@tul.cz

Abstract

We present here some details of our implementation of Wavelet-Galerkin method for Poisson equation in C language parallelized by POSIX threads library and show its performance in dimensions $d \in \{3, 4, 5\}$.

1. Introduction

Due to storage requirements and computational complexity, the approximate solution of PDEs computed by standard numerical methods is usually limited to problems with up to three or fourth dimensions. However in mathematical modeling, there is a lot of problems which involve more than three or four dimensions. For example, the pricing of financial derivatives, problems in quantum mechanics and particle physics. Here, the dimension grows with the number of considered derivatives, electrons or nuclei. An important issue for numerical methods for higher-dimensional PDEs is that typical domains are usually hypercubes. And it is well-known, that the curse of dimensionality can be broken on tensor product domain $(0, 1)^d$ by using sparse grids [1] or by wavelets [5].

To use wavelets efficiently to solve PDEs, it is necessary to have very efficient matrix-vector multiplication for vectors and matrices in wavelet coordinates and to have at one's disposal suitable wavelet bases. Wavelets should have short supports and vanishing moments, be smooth and known in closed form, and a corresponding wavelet basis should be well-conditioned.

In [5], authors were able to solve Poisson equation up to 10 dimensions by applying an adaptive wavelet scheme with orthonormal continuous piecewise linear multiwavelets proposed in [6]. They exploited the fact that the corresponding stiffness matrices are in tensor product wavelet coordinates well-conditioned independently on the dimension. Their approximations converged in energy norm with the same rate as the best N-term approximations independent of d with the cost of producing these approximations proportional to their length up to a constant factor growing potentially with the dimension, but only linearly.

We try to improve results obtained in [5] by applying higher order wavelet basis. In recent years, several promising constructions of wavelets were proposed. We mention, for example, a construction of spline-wavelet bases on the interval proposed in [2]. Their bases are compactly supported and generate multiresolution analyses on the unit interval with the desired numbers of vanishing wavelet moments for primal and dual wavelets. Moreover, the condition number of interval spline-wavelet bases is close to the condition number of the spline wavelet bases on the real line for bases up to order 4. In our contribution, we use recently proposed wavelets based on quadratic splines [3] which have shorter supports and are better conditioned. It is a modification of basis proposed in [4] with an improved condition number. Some preliminary results were already presented in [7]. There, a sequential algorithm was used to solve Poisson equation for $d \in \{2, 3\}$.

2. Problem formulation

We solve Dirichlet problem

$$\begin{aligned} -\sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} &= f \qquad x \in \Omega = (0,1)^d \\ u &= 0 \qquad x \in \partial\Omega \end{aligned}$$

by Galerkin method. Basis functions are wavelets based on quadratic splines proposed in [3] extended to higher dimensions by tensor product. Stiffness matrices are computed exactly. Used quadratic splines have points of discontinuity at $\frac{1}{2^L}, \frac{2}{2^L}, \ldots, \frac{2^L-1}{2^L}$ where L denotes the number of decomposition levels. Right-hand side integrals are calculated by adaptive Simpson rule. We split integration to hypercubes of size $(2^{-L})^d$ which enables efficient parallelization. We solve the arising system of linear algebraic equations originated from discretization by the conjugate gradient method with standard wavelet preconditioning consisting in normalizing all basis functions with respect to a bilinear form corresponding to stiffness matrix. It practically means that the stiffness matrix is multiplied from both sides by a diagonal matrix which has at its diagonal square root of diagonal elements of the original stiffness matrix.

We aim at an efficient implementation of adaptive wavelet methods for higher dimensional problems. For this purpose it is necessary to implement an efficient storage of sparse vectors and sparse matrices in wavelet coordinates and their efficient multiplication. We have so far implemented an efficient algorithm for matrix-vector multiplication in the case $d = 1$ and because stiffness matrices for Poisson equation in higher dimensions are computed from the stiffness matrices for Poisson equation in one dimension and from matrices of scalar products of basis functions in one dimension, we apply it here also for $d \in \{3, 4, 5\}$. Here, we present a non-adaptive implementation. It means that we choose a number of levels L and a dimension d which leads to 2^{Ld} basis functions.

3. Implementation and parallelization

In next subsections, we shortly describe some implementation details – a computation of right-hand side integrals and a multiplication of vector by stiffness matrix.

3.1. Computation of right-hand side integrals

Right-hand side integrals are in the form

$$\int_{(0,1)^d} \psi_{i_1}(x_1) \dots \psi_{i_d}(x_d) f(x_1 \dots x_d) \, \mathrm{d}x_1 \dots \mathrm{d}x_d, \tag{1}$$

where functions ψ_{i_j} are piecewise quadratic. Therefore we can split hypercube $(0,1)^d$ to hypercubes of size $(2^{-L})^d$ and compute integrals

$$\int x_1^{i_1} \dots x_d^{i_d} f(x_1 \dots x_d) \, \mathrm{d}x_1 \dots \mathrm{d}x_d \tag{2}$$

at each small hypercube for $i_1, \dots, i_d \in \{0, 1, 2\}$. Consequently, we compute (1) as a linear combination of integrals (2). To calculate (2) we use Fubini's theorem and a recursion. Let us denote $x_i = \frac{i}{2^L}$. We designed an implementation of Simpson rule for a computation of iterated integrals $I = \int_{x_i}^{x_{i+1}} F(x) \, \mathrm{d}x$ described below in 1.-5. Main goal of our design is to omit evaluation of the same value of function F twice because it is again an integral and its evaluation is computationally expensive.

1. Compute recursively $F(x_i)$ and $F(x_{i+1})$ and evaluate
$$I_0 = \tfrac{x_{i+1}-x_i}{2} \left(F(x_i) + F(x_{i+1}) \right).$$
2. Set $j = 0$.
3. Compute recursively $F(\xi_{i,k})$ with
$$\xi_{i,k} = x_i + \tfrac{2k-1}{2^{L+j+1}} \text{ for } k = 1, 2, 3, \dots, 2^j$$
and evaluate
$$I' = \tfrac{x_{i+1}-x_i}{2^j} \sum_{k=1}^{2^j} F(\xi_{i,k}).$$
4. If $|I_j - I'| > \varepsilon$, set $j = j + 1$, compute $I_j = \frac{1}{2}(I_{j-1} + I')$ and go to step 3.
5. Compute $I \approx \frac{1}{3}(I_j + 2I')$.

As mentioned above, we compute right-hand side integrals separately on hypercubes $(2^{-L})^d$. These integrals can be computed independently which enables simple parallelization. Our implementation is in C language and for parallelization we use a POSIX threads library. Every thread takes an index of a hypercube from a global variable in a loop, then increases the index and computes integrals. Taking and increasing global variable is a critical section. Therefore we use mutex (mutual exclusion) to synchronize threads there.

3.2. Multiplication of vector by stiffness matrix

We have implemented a very efficient algorithm of matrix multiplication in case $d = 1$. It stores stiffness matrix with entries

$$d_{ij} = \int_0^1 \psi_i'(x)\psi_j'(x)\,\mathrm{d}x \tag{3}$$

in a constant space with respect to number of levels L and run in a linear time with respect to a matrix order. You can find a description of this algorithm in [8]. We use a tensor product of 1D bases as a multi-dimensional basis

$$\psi_{i_1,\ldots,i_d}(x_1,\ldots,x_d) = \psi_{i_1}(x_1)\cdots\psi_{i_d}(x_d)$$

and entries of the corresponding stiffness matrix

$$a_{i_1,\ldots,i_d,i'_1,\ldots,i'_d} = \int_{[0,1]^d} \nabla\psi_{i_1,\ldots,i_d}\nabla\psi_{i'_1,\ldots,i'_d}. \tag{4}$$

We derive how to express the matrix $\mathbf{a}$ through matrices $\mathbf{d}$ and $\mathbf{g}$

$$g_{ij} = \int_0^1 \psi_i(x)\psi_j(x)\,\mathrm{d}x.$$

Note that used spline-wavelet basis is not orthonormal and so $\mathbf{g}$ is not identity matrix. We put $d = 3$ for the sake of simplicity. Matrix (4) is then given by

$$\begin{aligned} a_{i,j,k,i',j',k'} \;=\; & \int_{[0,1]^3} \psi_i'(x_1)\psi_j(x_2)\psi_k(x_3)\psi_{i'}'(x_1)\psi_{j'}(x_2)\psi_{k'}(x_3) + \\ & +\psi_i(x_1)\psi_j'(x_2)\psi_k(x_3)\psi_{i'}(x_1)\psi_{j'}'(x_2)\psi_{k'}(x_3) + \\ & +\psi_i(x_1)\psi_j(x_2)\psi_k'(x_3)\psi_{i'}(x_1)\psi_{j'}(x_2)\psi_{k'}'(x_3) \end{aligned}$$

and can be expressed as

$$a_{i_1,i_2,i_3,i'_1,i'_2,i'_3} = d_{ii'}g_{jj'}g_{kk'} + g_{ii'}d_{jj'}g_{kk'} + g_{ii'}g_{jj'}d_{kk'}$$

and multiplication of right-hand side $\mathbf{r}$ with $\mathbf{a}$ as

$$\sum_{i',j',k'} \left(d_{ii'}g_{jj'}g_{kk'} + g_{ii'}d_{jj'}g_{kk'} + g_{ii'}g_{jj'}d_{kk'}\right) r_{i'j'k'}. \tag{5}$$

To compute (5) we use the following algorithm

1. Compute $r^0_{ij'k'} = \sum_{i'} g_{ii'}r_{i'j'k'}$ and $r^1_{ij'k'} = \sum_{i'} d_{ii'}r_{i'j'k'}$ as a one-dimensional multiplication for all j', k'.

2. $r^0_{ijk'} = \sum_{j'} g_{jj'}r^0_{ij'k'}$,
   $r^1_{ijk'} = \sum_{j'} g_{jj'}r^1_{ij'k'}$,
   $r^2_{ijk'} = \sum_{j'} d_{jj'}r^0_{ij'k'}$.

3. $r^1_{ijk} = \sum_{k'} g_{kk'} r^1_{ijk'}$,
   $r^2_{ijk} = \sum_{k'} g_{kk'} r^2_{ijk'}$,
   $r^3_{ijk} = \sum_{k'} d_{kk'} r^0_{ijk'}$.

4. $r_{ijk} = r^1_{ijk} + r^2_{ijk} + r^3_{ijk}$.

Then, we have 8 matrix-vector multiplication in steps 1.–3. In each step, all multiplications are independent and are computed in parallel. In the case $d = 4$, we have 13 multiplications in 4 groups and for $d = 5$, we have 19 multiplications in 5 groups.

4. Numerical experiments

We run our code for Poisson equation in dimensions $d \in \{3, 4, 5\}$ with the solution

$$u(x_1, x_2, \ldots, x_d) = (1 - x_1)(1 - x_2) \ldots (1 - x_d) \left(1 - e^{(-10 x_1 x_2 \ldots x_d)}\right).$$

In Table 1, d denotes dimension, L denotes the decomposition level of wavelet basis, N is the matrix size, $RHS16$ (m) and $RHS8$ (m), respectively denotes time of computation of right-hand side integrals in minutes in 16 and 8 threads, respectively, $\#CG$ denotes the number of iterations of the conjugate gradient method and $CG(m)$ denotes time of computation of the conjugate gradient method in minutes. We used for our computation a processor with frequency 2.3 GHz and with 16 cores.

d	L	N	$RHS16(m)$	$RHS8(m)$	#CG	CG (m)	L_2 norm of error
3	8	2^{24}	10	21	177	100	$1.6 \cdot 10^{-11}$
3	9	2^{27}	136	260	199	920	$1.1 \cdot 10^{-12}$
4	5	2^{20}	9	18	161	5	$5.9 \cdot 10^{-9}$
4	6	2^{24}	49	92	203	120	$4.5 \cdot 10^{-10}$
5	4	2^{20}	250	480	128	3	$1.5 \cdot 10^{-8}$
5	5	2^{25}	520	-	176	200	$1.5 \cdot 10^{-9}$

Table 1: Results of numerical experiments.

5. Conclusion

We have presented here some details of our implementation of Wavelet-Galerkin method for Poisson equation in dimension $d \in \{3, 4, 5\}$ in C language parallelized by POSIX threads library. Parallelization of evaluation of right-hand side integrals is efficient – enables concurrent evaluation by as many threads as the number of available computational cores. The ratio of total CPU time and real time is in the case of 16 threads around 15.8. This is not the case for the conjugate gradient method and our future goal is to improve it. Another goal is to design and implement appropriate data structures for adaptive methods.

Acknowledgements

This work has been supported by the project ESF No. CZ.1.07/2.3.00/09.0155 "Constitution and improvement of a team for demanding technical computations on parallel computers at TU Liberec"

References

[1] Bungartz, H. J. and Griebel, M.: Sparse grids. Acta Numer. **13** (2004), 147–269.

[2] Černá; D. and Finěk, V.: Construction of optimally conditioned cubic spline wavelets on the interval. Adv. Comput. Math. **34** (2011), 519–552, 2011.

[3] Černá; D. and Finěk, V.: The construction of well-conditioned wavelet basis based on quadratic B-splines. To appear In: Simos, T. E. (Ed.) *ICNAAM – Numerical Analysis and Applied Mathematics*, American Institute of Physics, New York, 2012.

[4] Černá, D., Finěk, V., and Šimůnková, M.: A quadratic spline-wavelet basis on the interval. In: Chleboun, J., Segeth, K., Šístek, J., Vejchodský, T. (Eds.), *Programs and Algorithms of Numerical Matematics 16*, pp. 29–34. Institute of Mathematics AS CR, Prague, 2013.

[5] Dijkema, T. J., Schwab, Ch., and Stevenson, R.: An adaptive wavelet method for solving high-dimensional elliptic PDEs. Constr. Approx. **30** (3) (2009), 423–455.

[6] Donovan, G. C., Geronimo, J. S., and Hardin, D. P.: Intertwining multiresolution analyses and the construction of piecewise-polynomial wavelets. SIAM J. Math. Anal. **27** (6) (1996), 1791–1815.

[7] Finěk, V. and Šimůnková, M.: Effective implementation of wavelet Galerkin method. To appear. In: Venkov, G., Kovacheva, R., Pasheva, V. (Eds.), *AMEE – Applications of Mathematics in Engineering and Economics*, American Institute of Physics, New York, 2012.

[8] Šimůnková, M.: *Multiplication by wavelet matrix – efficient implementation.* Submitted to ACC Journal.