Graph grammar based multi-frontal direct solver for isogeometric FEM simulations on GPU

***M. Paszyński[1], K. Kuźnik[1], V.M. Calo[2] and D. Pardo[3]**

[1]AGH University of Science and Technology, Krakow, Poland.
[2]King Abdullah University of Science and Technology, Thuwal, Saudi Arabia.
[3]The University of the Basque Country, UPV/EHU and Ikerbasque, Bilbao, Spain.

*Corresponding author: maciej.paszynski@AGH.EDU.PL

Abstract

We present a multi-frontal direct solver for two dimensional isogeometric finite element method simulations with NVIDIA CUDA and perform numerical experiments for linear, quadratic and cubic B-splines. We compare the computational cost $O(Np^2)$ for 2D parallel shared memory implementation with the corresponding estimate $O(N^{1.5}p^3)$ for a standard 2D sequential implementation. We conclude the presentation with observation that computational cost of the shared memory direct solver scales like p^2 when we increase the global continuity of the isogeometric solution, which is an adventage with respect to sequential isogeometric solver scalability of the order of p^3.

Keywords: Multi-frontal direct solver, isogeometric finite element method, computational cost, shared memory machine

Introduction

The isogeometric finite element method (Cottrel et al. 2009) is a higher order method providing global C^k continuity of the solution. It is based on the usage of B-spline basis functions delivering higher order global regularity of the solution. The classical higher order finite element method (Demkowicz 2006, Demkowicz et al. 2007) provides C^0 global continuity only. The isogeometric finite element method generates a sparse system of equations that can be solved by multi-frontal direct solver algorithm (Duff et al. 1984, Duff et al. 1983, Geng et al. 2006).
In this paper we present how isogeometric C^k finite element method multi-frontal solver differs from C^0 higher order finite element method solver by factor of p^3.We also show how this p^3 factor can be reduce down to p^2 factor by using shared memory implementation.

B-spline based isogeometric finite element method

We focus on the 2D model problem, namely the Laplace equation over a square domain

$$\Delta u(x_1,x_2)=0 \text{ for } (x_1,x_2)\in\Omega=[0,1]^2 \tag{1}$$

$$\text{tr } u(x_1,x_2)=0 \text{ for } x_1\in[0,1],\ x_2=0 \tag{2}$$

$$\text{tr } u(x_1,x_2)=1 \text{ for } x_1\in[0,1],\ x_2=1 \tag{3}$$

$$\frac{\partial u}{\partial x_1}(x_1,x_2)=0 \text{ for } x_1\in\{0,1\},\ x_2\in[0,1] \tag{4}$$

where

$$\Gamma_D=\{(x_1,x_2): x_1\in[0,1], x_2\in\{0,1\}\} \tag{5}$$

$$\Gamma_N=\{(x_1,x_2): x_1\in\{0,1\}, x_2\in(0,1)\} \tag{6}$$

This is a simple model problem, and the solution is $u(x_1,x_2)=x_2$. The problem is transformed into weak variational form. We seek $u\in\tilde{u}_0+H^1(\Omega)$ where $\tilde{u}_0+H^1(\Omega)=\{\tilde{u}_0+v: v\in H^1(\Omega)\}$ and $\tilde{u}_0(x_1,x_2)=x_2$ is the lift of the Dirichlet b.c. In other words $u(x_1,x_2)=w(x_1,x_2)+\tilde{u}_0(x_1,x_2)=w(x_1,x_2)+x_2$ and

$$\int_\Omega \sum_{i=1,2} \frac{\partial^2 u(x_1,x_2)}{\partial x_i} v(x_1,x_2)dx_1dx_2 = 0 \quad \text{for all } v\in \mathbf{V}=\{v\in H^1(\Omega): \mathrm{tr}\, v=0 \text{ on } \Gamma_D\} \tag{7}$$

we integrate by parts

$$-\int_\Omega \sum_{i=1,2} \frac{\partial u(x_1,x_2)}{\partial x_i}\frac{\partial v(x_1,x_2)}{\partial x_i}dx_1dx_2 + \int_{\Gamma_N} \frac{\partial u(x_1,x_2)}{\partial n} v(x_1,x_2)dS + \int_{\Gamma_D} \frac{\partial u(x_1,x_2)}{\partial n} v(x_1,x_2)dS = 0 \tag{8}$$

$\mathrm{tr}\, v=0$ on Γ_D which implies $\int_{\Gamma_D} \frac{\partial u(x_1,x_2)}{\partial n} v(x_1,x_2)dS=0$, $\frac{\partial u}{\partial x_1}(x_1,x_2)=0$ on Γ_N which implies $\int_{\Gamma_N} \frac{\partial u(x_1,x_2)}{\partial n} v(x_1,x_2)dS=0$ (since $\frac{\partial u}{\partial n}=\pm\frac{\partial u}{\partial x_1}$ on Γ_N) and we get

$$\int_\Omega \sum_{i=1,2} \frac{\partial u(x_1,x_2)}{\partial x_i}\frac{\partial v(x_1,x_2)}{\partial x_i}dx_1dx_2 = 0 \tag{9}$$

incorporating "shift" of the Dirichlet boundary condition $u(x_1,x_2)=w(x_1,x_2)+\tilde{u}_0(x_1,x_2)=w(x_1,x_2)+x_2$

$$\int_\Omega \sum_{i=1,2} \frac{\partial w(x_1,x_2)}{\partial x_i}\frac{\partial v(x_1,x_2)}{\partial x_i}dx_1dx_2 = -\int_\Omega \sum_{i=1,2} \frac{\partial \tilde{u}_0(x_1,x_2)}{\partial x_i}\frac{\partial v(x_1,x_2)}{\partial x_i}dx_1dx_2$$ and since $\frac{\partial \tilde{u}_0(x_1,x_2)}{\partial x_1}=0$, $\frac{\partial \tilde{u}_0(x_1,x_2)}{\partial x_2}=1$

we get: Find $w\in H^1(\Omega)$ such that:

$$\int_\Omega \sum_{i=1,2} \frac{\partial w(x_1,x_2)}{\partial x_i}\frac{\partial v(x_1,x_2)}{\partial x_i}dx_1dx_2 = -\int_\Omega \frac{\partial v(x_1,x_2)}{\partial x_2}dx_1dx_2 \quad \text{for all } v\in \mathbf{V}=\{v\in H^1(\Omega): \mathrm{tr}\, v=0 \text{ on } \Gamma_D\} \tag{10}$$

If we utilize B-splines for p=1

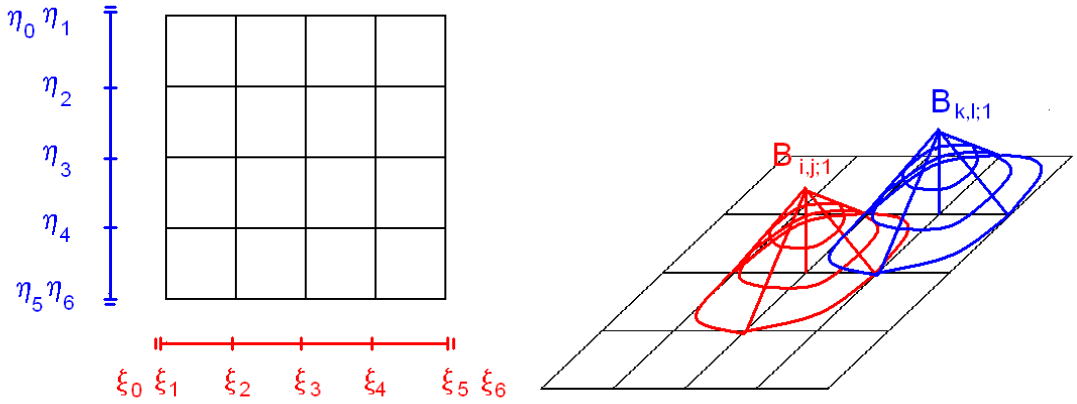

Figure 1. Linear B-splines over 2D patch

our week variational form

$$b(v,w)=l(v) \quad \forall v\in\mathbf{V}; \quad b(v,w)=\int_\Omega \sum_{i=1,2} \frac{\partial w(x_1,x_2)}{\partial x_i}\frac{\partial v(x_1,x_2)}{\partial x_i}dx_1dx_2; \quad l(v)=-\int_\Omega \frac{\partial v(x_1,x_2)}{\partial x_2}dx_1dx_2 \tag{11}$$

has the following tensor product structure

$$B_{i,j;1}(x_1,x_2)=N_{i;1}(x_1)N_{j;1}(x_2) \tag{12}$$

$$w(x_1,x_2) \approx \sum_{i,j} B_{i,j;1}(x_1,x_2) a_{i,j} = \sum_{i,j} N_{i;1}(x_1) N_{j;1}(x_2) a_{i,j} \tag{13}$$

$$v(x_1,x_2) \leftarrow B_{i,j;1}(x_1,x_2) = N_{i;1}(x_1) N_{j;1}(x_2) \tag{14}$$

$$B_{i,j;1}(x_1,x_2) = N_{i;1}(x_1) N_{j;1}(x_2) \tag{15}$$

$$b\left(B_{i,j;1}, B_{k,l;1}\right) = l\left(B_{k,l;1}\right) \tag{16}$$

where

$$\begin{aligned} b\left(B_{i,j;1}, B_{k,l;1}\right) &= \int_{\Omega} \sum_{m=1,2} \frac{\partial\left[N_{i;1}(x_1) N_{j;1}(x_2)\right]}{\partial x_m} \frac{\partial\left[N_{k;1}(x_1) N_{l;1}(x_2)\right]}{\partial x_m} dx_1 dx_2 \\ l\left(B_{k,l;1}\right) &= -\int_{\Omega} \frac{\partial\left[N_{k;1}(x_1) N_{l;1}(x_2)\right]}{\partial x_2} dx_1 dx_2 \end{aligned} \tag{17}$$

using Gaussian quadrature the integration over the domain can be substituted by a weighted summation over Gaussian points

$$\begin{aligned} b\left(B_{i,j;1}, B_{k,l;1}\right) &= \int_{\Omega} \sum_{m=1,2} \frac{\partial\left[N_{i;1}(x_1) N_{j;1}(x_2)\right]}{\partial x_m} \frac{\partial\left[N_{k;1}(x_1) N_{l;1}(x_2)\right]}{\partial x_m} dx_1 dx_2 = \\ &= \sum_n w_n \sum_{m=1,2} \frac{\partial\left[N_{i;1}\left(x_1^n\right) N_{j;1}\left(x_2^n\right)\right]}{\partial x_m} \frac{\partial\left[N_{k;1}\left(x_1^n\right) N_{l;1}\left(x_2^n\right)\right]}{\partial x_m} \\ l\left(B_{k,l;1}\right) &= -\int_{\Omega} \frac{\partial\left[N_{k;1}(x_1) N_{l;1}(x_2)\right]}{\partial x_2} dx_1 dx_2 = -\sum_n w_n \frac{\partial\left[N_{k;1}\left(x_1^n\right) N_{l;1}\left(x_2^n\right)\right]}{\partial x_2} \end{aligned} \tag{18}$$

Multi-frontal solver algorithm for linear B-splines

We partition the mesh into “elements”, compare Figure 2

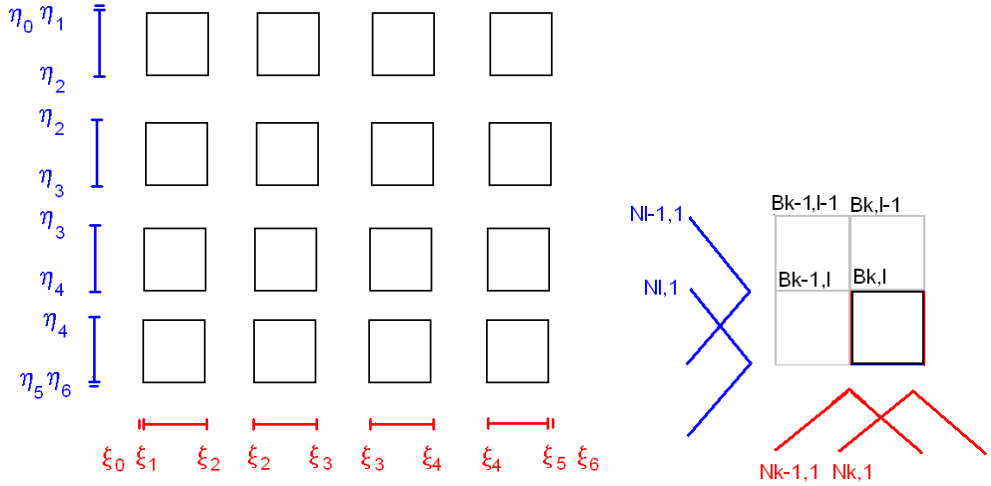

Figure 2. Left panel: Partitioning of the 2D patch into elements
Right panel: Linear B-splines over a single element.
The maxim values of B-splines ale located at mesh nodes,
thus we identify B-splines with mesh nodes

We have $(p+1)(p+1)$ functions of order p assigned to element $E_{k,l}=[\xi_K,\xi_{K+1}]\times[\eta_L,\eta_{L+1}]$

$$\left\{B_{m,n;1}(x_1,x_2)\right\}_{m=k-p,\ldots k;n=l-p,\ldots l}=\left\{N_{m;1}(x_1)N_{n;1}(x_2)\right\}_{m=k-p,\ldots k;n=l-p,\ldots l} \tag{19}$$

We need to perform the integration over a single element

$$\begin{aligned}
&b\left(B_{i,j;1},B_{k,l;1}\right)=\sum_n w_n\sum_{m=1,2}\frac{\partial\left[N_{i;1}\left(x_1^n\right)N_{j;1}\left(x_2^n\right)\right]}{\partial x_m}\frac{\partial\left[N_{k;1}\left(x_1^n\right)N_{l;1}\left(x_2^n\right)\right]}{\partial x_m}=\\
&\sum_n\left\{w_n\frac{\partial\left[N_{i;1}\left(x_1^n\right)N_{j;1}\left(x_2^n\right)\right]}{\partial x_1}\frac{\partial\left[N_{k;1}\left(x_1^n\right)N_{l;1}\left(x_2^n\right)\right]}{\partial x_1}+\frac{\partial\left[N_{i;1}\left(x_1^n\right)N_{j;1}\left(x_2^n\right)\right]}{\partial x_2}\frac{\partial\left[N_{k;1}\left(x_1^n\right)N_{l;1}\left(x_2^n\right)\right]}{\partial x_2}\right\}=\\
&\sum_n\left\{w_n N_{j;1}\left(x_2^n\right)\frac{\partial N_{i;1}\left(x_1^n\right)}{\partial x_1}N_{l;1}\left(x_2^n\right)\frac{\partial N_{k;1}\left(x_1^n\right)}{\partial x_1}+N_{i;1}\left(x_1^n\right)\frac{\partial N_{j;1}\left(x_2^n\right)}{\partial x_2}N_{k;1}\left(x_1^n\right)\frac{\partial N_{l;1}\left(x_2^n\right)}{\partial x_2}\right\}\\
&l\left(B_{k,l;1}\right)=-\sum_n w_n\frac{\partial\left[N_{k;1}\left(x_1^n\right)N_{l;1}\left(x_2^n\right)\right]}{\partial x_2}=-\sum_n w_n N_{k;1}\left(x_1^n\right)\frac{\partial N_{l;1}\left(x_2^n\right)}{\partial x_2}
\end{aligned} \tag{20}$$

For $p=1$ there are 2x2=4 two dimensional B-splains (compare Figure 2) so we need to compute 4x4 matrix with all four functions talking to each other (see Table 1) so we need to compute 2x2=4 two dimensional B-splines (see Table 2) so we need to compute 2+2=4 one dimensional B-splines (see Table 3).

$b\left(B_{k-1,l-1;1},B_{k-1,l-1;1}\right)$	$b\left(B_{k,l-1;1},B_{k-1,l-1;1}\right)$	$b\left(B_{k-1,l;1},B_{k-1,l-1;1}\right)$	$b\left(B_{k,l;1},B_{k-1,l-1;1}\right)$
$b\left(B_{k-1,l-1;1},B_{k,l-1;1}\right)$	$b\left(B_{k,l-1;1},B_{k,l-1;1}\right)$	$b\left(B_{k-1,l;1},B_{k,l-1;1}\right)$	$b\left(B_{k,l;1},B_{k,l-1;1}\right)$
$b\left(B_{k-1,l-1;1},B_{k-1,l;1}\right)$	$b\left(B_{k,l-1;1},B_{k-1,l;1}\right)$	$b\left(B_{k-1,l;1},B_{k-1,l;1}\right)$	$b\left(B_{k,l;1},B_{k-1,l;1}\right)$
$b\left(B_{k-1,l-1;1},B_{k,l;1}\right)$	$b\left(B_{k,l-1;1},B_{k,l;1}\right)$	$b\left(B_{k-1,l;1},B_{k,l;1}\right)$	$b\left(B_{k,l;1},B_{k,l;1}\right)$

Table 1. Element matrix for linear B-splines

$B_{k,l;1}(x_1,x_2)=N_{k;1}(x_1)N_{l;1}(x_2)$	$B_{k,l-1;1}(x_1,x_2)=N_{k;1-1}(x_1)N_{l;1}(x_2)$
$B_{k-1,l;1}(x_1,x_2)=N_{k-1;1}(x_1)N_{l;1}(x_2)$	$B_{k-1,l-1;1}(x_1,x_2)=N_{k-1;1-1}(x_1)N_{l;1}(x_2)$

Table 2. Contribution to a single entry of linear B-splines based element matrix

$N_{k;1}(x_1)$	$N_{k-1;1}(x_1)$
$N_{l;1}(x_2)$	$N_{l-1;1}(x_2)$

Table 3. One dimensional B-splines contributing to linear B-splines based element matrix

In the second step we merge 2x2 elements (actually we merge four 4x4 matrices into a single 9x9 matrix) as it is presented on left panel in Figure 3. We order the matrices in such a way so the fully assembled B-spline (denoted by dark green color on left panel) is the first row in the matrix so we can eliminate the first row (we can eliminate the single fully assembled B-spline).
In the third step we merge four patches of 2x2 element with internal B-splines already eliminated (actually we merge four 8x8 matrices into a single 21x21 matrix), compare middle panel in Figure 3. We order the matrix in such a way so the 5 fully assembled B-splines (denoted by dark green color on the middle panel) are the first rows in the matrix so we can eliminate the first rows (we can eliminate the internal fully assembled B-splines).
In the last step we eliminate the boundary nodes, see right panel in Figure 3 (we have a single 16x16 matrix). We just perform fully forward elimination over the matrix. This is the most expensive step and its computational complexity is $O(N^{3/2})$ since the boundary has $O(N^{1/2})$ B-splines.

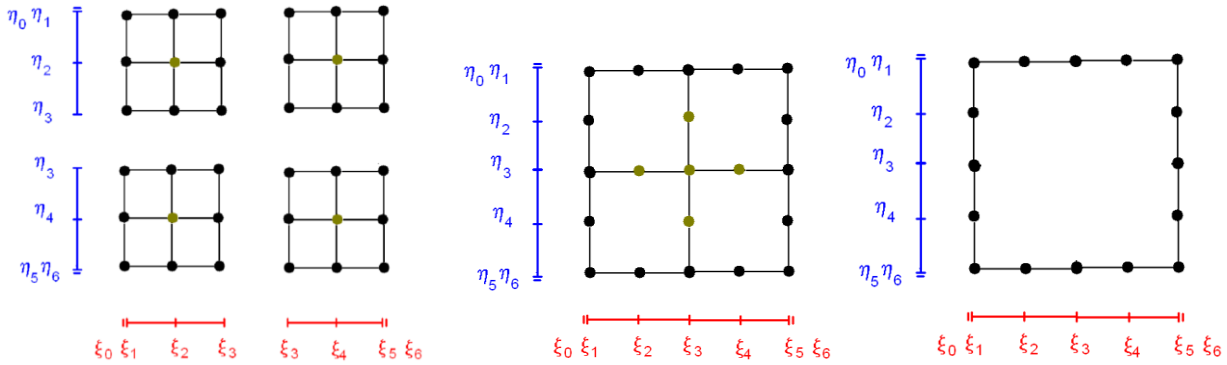

Figure 3. Left panel: Second step of the linear B-splines based solver algorithm Middle panel: Third step of the linear B-splines based solver algorithm (dark yellow dots denote B-splines to be eliminated) Right panel: We use linear B-splines so the top problem has one layer of B-splines identified with mesh nodes

Multi-frontal solver algorithm for quadratic B-splines

In the first step we generate "elements", with quadratic B-splines as it is presented in in Figure 4. For p=2 there are 3x3=9 two dimensional B-splines so we need to compute 9x9 matrix with all nine functions talking to each other (order in the matrix corresponds to the row by row location of 2D B-splines) (compare Table 4) so we need to compute 3x3=9 two dimensional B-splines (compare Table 5) so we need to compute 3+3=6 one dimensional B-splines (compare Table 6). We already know how to compute them effectively. Thus, the parallel shared memory implementation of the integration algorithm requires only p+1 steps.

$b(B_{k-2,l-2;1}, B_{k-2,l-2;1})$	…	$b(B_{k,l;1}, B_{k-2,l-2;1})$
…	…	…
$b(B_{k-2,l-2;1}, B_{k,l;1})$	…	$b(B_{k,l;1}, B_{k,l;1})$

Table 4. Element matrix for quadratic B-splines

$B_{k,l;1}(x_1,x_2) = N_{k;1}(x_1)N_{l;1}(x_2)$	$B_{k,l-1;1}(x_1,x_2) = N_{k;1-1}(x_1)N_{l;1}(x_2)$	$B_{k,l-2;1}(x_1,x_2) = N_{k;1}(x_1)N_{l-2;1}(x_2)$
$B_{k-1,l;1}(x_1,x_2) = N_{k-1;1}(x_1)N_{l;1}(x_2)$	$B_{k-1,l-1;1}(x_1,x_2) = N_{k-1;1-1}(x_1)N_{l;1}(x_2)$	$B_{k-1,l-2;1}(x_1,x_2) = N_{k-1;1-1}(x_1)N_{l-2;1}(x_2)$
$B_{k-2,l;1}(x_1,x_2) = N_{k-2;1}(x_1)N_{l;1}(x_2)$	$B_{k-2,l-1;1}(x_1,x_2) = N_{k-2;1-1}(x_1)N_{l;1}(x_2)$	$B_{k-2,l-2;1}(x_1,x_2) = N_{k-2;1-1}(x_1)N_{l-2;1}(x_2)$

Table 5. Contribution to a single entry of quadratic B-splines based element matrix

$N_{k;1}(x_1)$	$N_{k-1;1}(x_1)$	$N_{k-2;1}(x_1)$
$N_{l;1}(x_2)$	$N_{l-1;1}(x_2)$	$N_{l-2;1}(x_2)$

Table 6. One dimensional B-splines contributing to quadratic B-splines based element matrix

In the second step we merge 3x3 elements (actually we merge nine 4x4 matrices into a single 16x16 matrix), compare left panel in Figure 5. We order the matrices in such a way so the one fully assembled B-spline (denoted by yellow color on figure) is the first row in the matrix so we can eliminate the first row (we can eliminate the single fully assembled B-spline).
In the third step presented on right panel in Figure 5 we merge four patches of 3x3 elements each with one B-spline already eliminated. We get a matrix with 8x8-4 = 60 rows / columns. We can eliminate 12 fully assembled B-splines.

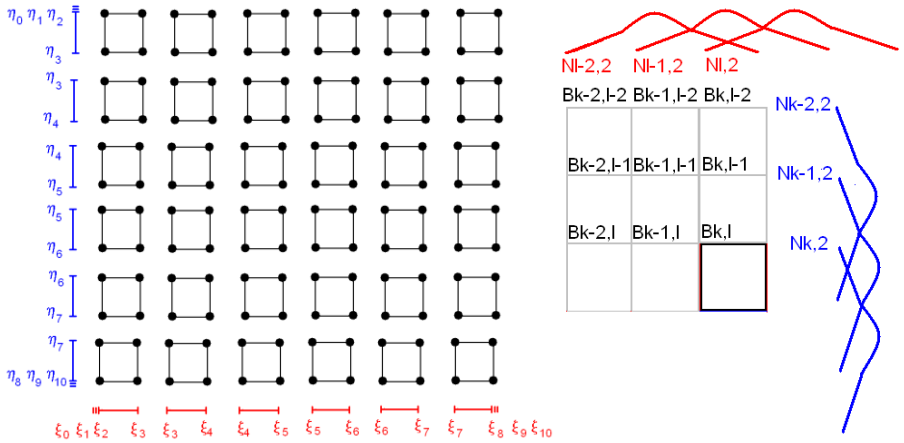

Figure 4. Left panel: Partitioning of the 2D patch into elements
Right panel: Quadratic B-splines over a single element. The maximum values of B-splines are located at element centers, thus we identify element centers with B-splines. Additionally, on the boundary elements we have one B-spline with maximum located outside the domain.

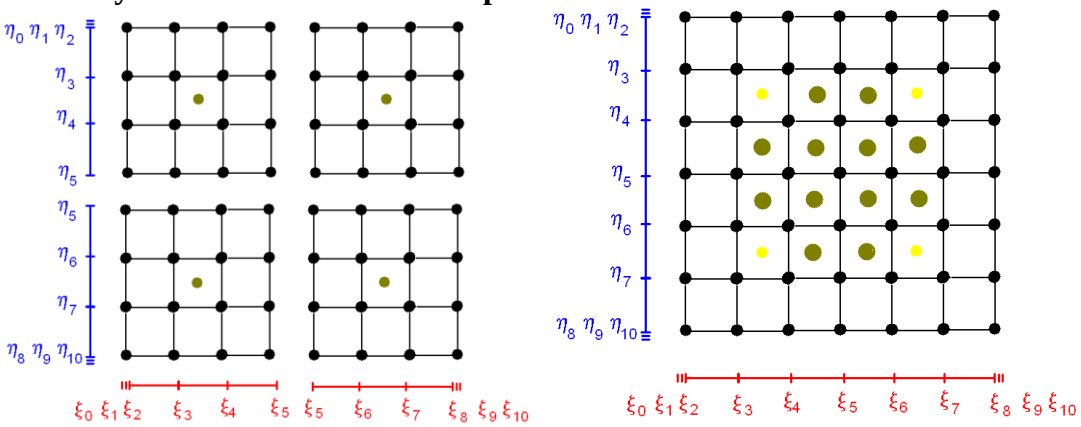

Figure 5. Left panel: First step of the elimination for quadratic B-splines
Right panel: Second step of the elimination for quadratic B-splines
(dark dots denote B-splines to be eliminated, light dots denote already eliminated B-splines)

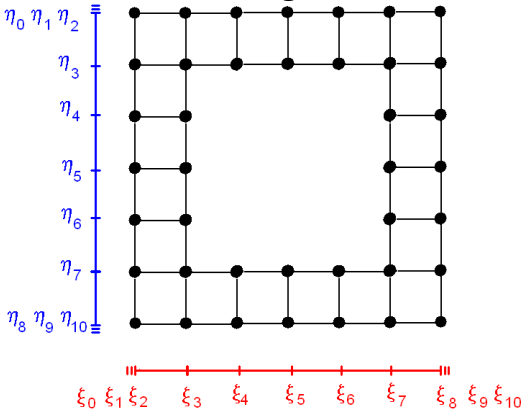

Figure 6. The top problem for quadratic B-splines finite elements has two layers, since we have one B-spline at each element center, as well as we have boundary B-splines outside the domain, one additional layer of B-splines

In the last step we end up with the boundary problem, presented in Figure 6. We have a single dense 48x48 matrix here, resulting from one layer of B-splines located at element interiors plus one additional layer of elements located outside the domain.

Observation 1. *The top problem for the B-splines based direct solver for C^k isogeometric finite element method has size $O(N^{1/2} p)$. The resulting computational cost of the top problem solution is $O(N^{3/2}p^3)$. This is the computationally most expensive part of the solution.*
Observation 2. *The top problem for the direct solver for classical C^0 finite element method has size $O(N^{1/2})$. The computational cost of the top problem solution is $O(N^{3/2})$.*

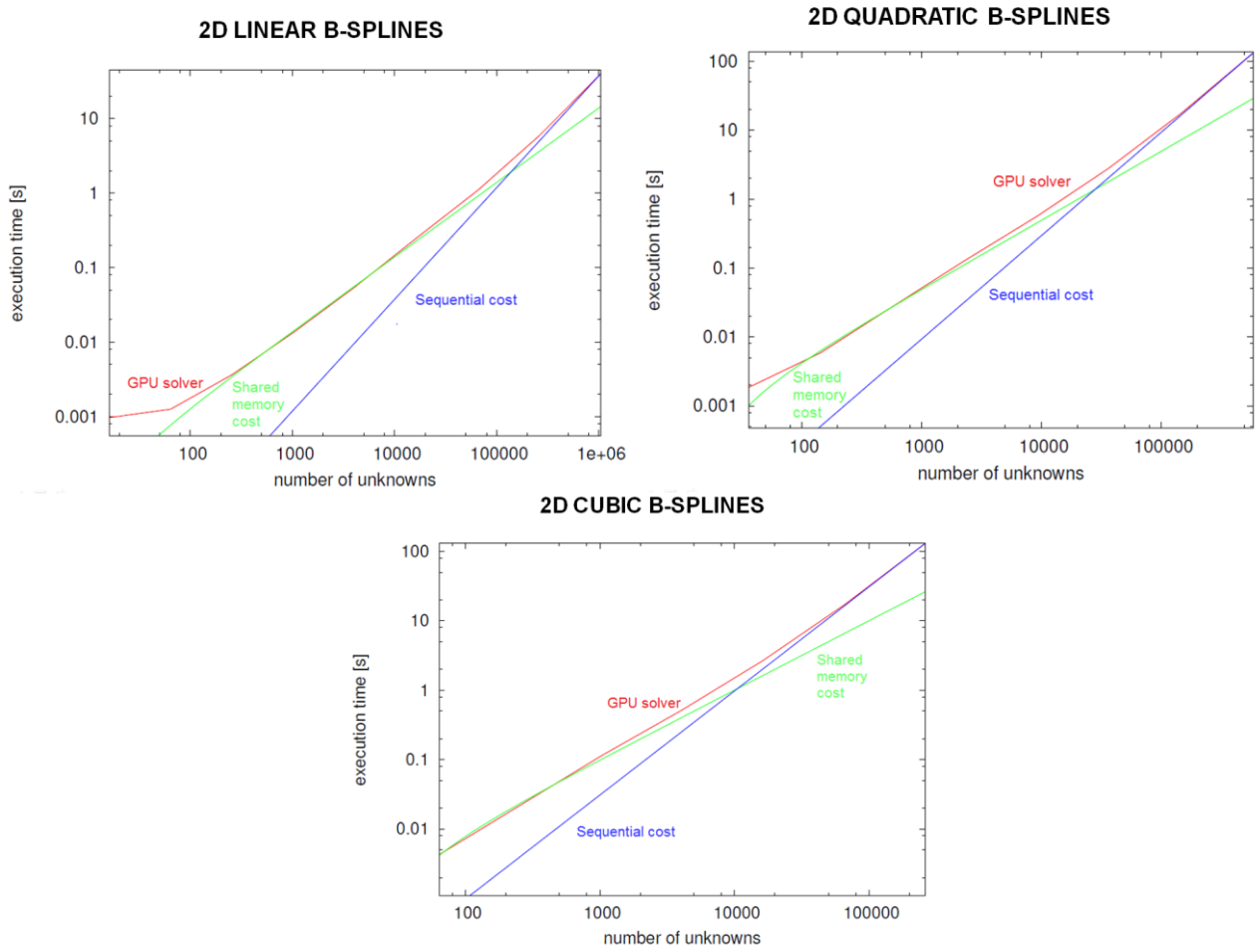

Figure 7. Comparison between theoretical and experimental computational costs

Graph grammar based multi-frontal solver

The shared memory implementation of the isogeometric direct solver algorithm has been already described in (Kuźnik et. al. 2012). The solver algorithm presented there is based on graph grammar concept and implemented in NVIDIA CUDA environment.
The theoretical estimates presented in (Kuźnik et al. 2013, Collier et al. 2012) imply the following computational complexities of the isogeometric as well as classical C^0 higher order FEM (Table 7).

Observation 3. *In the parallel shared memory implementation of the full Gaussian elimination for the top dense problem it is possible to perform row subtractions at the same time. There are $O(N^{1/2}p)$ rows to be subtracted at the same time, the size of each row is $O(N^{1/2}p)$, and these row subtractions must be performed $O(N^{1/2}p)$ times. This implies $O(Np^2)$ computational complexity of the isogeometric C^k finite element method shared memory direct solver.*

The experiments were performed on NVidia Tesla C2070 device, which has 14 multiprocessors with 32 CUDA cores per multiprocessor, which gives us 448 CUDA cores. The total amount of global memory is 5375 megabytes. We used CUDA 4.0 version.

	1D	2D	3D
parallel shared memory IGA	$O(p^2 \log(N/p))$	$O(Np^2)$	$O(N^{1.33} p^2)$
sequential single core IGA	$O(p^3(N/p))$	$O(N^{1.5}p^3)$	$O(N^2p^3)$
parallel shared memory hp-FEM	$O(\log(N/p))$	$O(N)$	$O(N^{1.33})$
sequential single core hp-FEM	$O(N/p)$	$O(N^{1.5})$	$O(N^2)$

Figure 7. Comparison between sequential and shared memory computational costs

The comparison of the theoretical and experimental computational costs are presented in Figure 7. We can observe here how the computational cost of the shared memory solver varies from ideal theoretical cost to sequential cost, when the problem size grows. There are the following reasons for such the behavior. For 2D solver, frontal matrices grow up the elimination tree.

- For the frontal matrices close to the root of the tree, even single row of a matrix cannot fit a shared memory of a single multiprocessor (only one core per multiprocessor is running)
- For the frontal matrices below the root, only a few rows fit into a shared memory of a single multiprocessor (several cores may be idle)
- For the frontal matrices close to leaves several rows fit into a shared memory of a single multiprocessor (all cores over all multiprocessors are running)
- For frontal matrices at the leaves entire frontal matrix fit into a shared memory of a single multiprocessor (all cores over all multiprocessors are running)

Conclusions

In this paper we presented how the isogeometric finite element method increases the computational cost of the multi-frontal solver by factor p^3. We also showed how shared memory version of the multi-frontal solver can reduce this factor down to p^2. The numerical experiments performed on NVIDIA CUDA GPU confirmed the theoretical estimates.

Acknowledgements

This work was supported by Polish National Science Center grant no. UMO-2012/07/B/ST6/01229.

References

Collier N.O., Pardo D., Paszynski M., Dalcin L., Calo V.M. (2012), The cost of continuity: a study of the performance of isogeometric finite elements using direct solvers, *Computer Methods in Applied Mechanics and Engineering*, 213-216, pp.353-361

Cottrel, J. A., Hughes, T. J. R., Bazilevs, Y. (2009) *Isogeometric Analysis. Towards Integration of CAD and FEA*, Wiley

Demkowicz, L. (2006) *Computing with hp-Adaptive Finite Element Method. Vol. I. One and Two Dimensional Elliptic and Maxwell Problems*. Chapmann & Hall / CRC Applied Mathematics and Nonlinear Science

Demkowicz L., Kurtz J., Pardo D., Paszynski M., Zdunek A. (2006), *Computing with hp-Adaptive Finite Element Method. Vol. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems*. Chapmann & Hall / CRC Applied Mathematics and Nonlinear Science

Duff I. S., Reid J. K. (1984), The multifrontal solution of unsymmetric sets of linear systems, *SIAM Journal of Scientific and Statistical Computing*, vol. 5, pp.633-641.

Duff I. S., Reid J. K. (1983) The multifrontal solution of indefinite sparse symmetric linear equations, *ACM Transations on Mathematical Software*, vol. 9, pp. 302-325

Geng P., Oden T. J., van de Geijn R. A. (2006) A Parallel Multifrontal Algorithm and Its Implementation, *Computer Methods in Applied Mechanics and Engineering*, vol. 149, pp.289-301.

Kuznik K., Paszynski M., Calo V. (2012) Graph Grammar-Based Multi-Frontal Parallel Direct Solver for Two-Dimensional Isogeometric Analysis. *Procedia Computer Science* 9, pp.1454-1463.

Kuznik K., Paszynski M., Calo V., Pardo D. (2013) Multi-Frontal Solvers for IGA Discretization in GPU, *Computers and Mathematics with Applications*, submitted.