SEECCM III
3rd South-East European Conference on Computational Mechanics-
an ECCOMAS and IACM Special Interest Conference
M. Papadrakakis, M. Kojic, I. Tuncer (eds.)
Kos Island, Greece, 12–14 June 2013

COMPUTATION OF THE ISOGEOMETRICANALYSIS STIFFNESS MATRIX ON GPU

Alexander Karatarakis[1], **Panagiotis Karakitsios**[2] **and Manolis Papadrakakis**[2]

[1] Institute of Structural Analysis and Antiseismic Research National Technical University of Athens
Zografou Campus, Athens 15780
e-mail: alex@karatarakis.com

[2] Institute of Structural Analysis and Antiseismic Research National Technical University of Athens
Zografou Campus, Athens 15780
pkarak@hotmail.com, mpapadra@central.ntua.gr

Keywords: Isogeometric analysis, Gauss quadrature, NURBS, Stiffness matrix assembly, Parallel computing, GPU acceleration

Abstract. *Due to high regularity across mesh elements of isogeometric analysis, this new method achieves higher accuracy per degree of freedom and improved spectrum properties, among others, compared to finite element analysis. However, this inherent feature of isogeometric analysis reduces the sparsity pattern of stiffness matrix and requires more elaborate numerical integration schemes for its computation. For these reasons, the assembly of the stiffness matrix in isogeometric analysis is a computationally demanding task, which needs special attention in order to be affordable in real-world implementations. In this paper we address the computational efficiency of assembling the stiffness matrix using the standard element-wise Gaussian quadrature. A novel approach is proposed for the formulation of the stiffness matrix which exhibits several computational merits, among them its amenability to parallelization and the efficient utilization of the graphic possessing units to drastically accelerate computations.*

1 INTRODUCTION

Isogeometric analysis (IGA) was recently introduced by Hughes et al. [1] and since then it has attracted a lot of attention for solving boundary value problems as a result of using the same basis functions adopted from CAD community for describing the domain geometry and for building the numerical approximation of the solution.

Despite IGA's promising methodology and superior features [1]–[4] compared with finite element analysis (FEA) the computation of mass, stiffness and advection matrices is more laborious, which increase the cost of IGA in real-world applications. For a certain number of degrees of freedom, IGA, due to its higher inter-element continuity, produces quite more elements than FEA. This leads to an increase of the number of Gauss points and consequently of the computational cost for assembling the characteristic matrices. This drawback increases dramatically the computational cost in the multivariate domains, especially for 3D cases.

It was shown [2], [3] that standard element-wise Gauss rules are inefficient, because they do not take precise account of the preserved smoothness at the element boundaries in the case of higher-order NURBS and polynomial B-SPLines, and that the higher the inter-element regularity is the fewer the required number of Gauss points per element. However, recently proposed integration rules, although they are optimal or nearly optimal in terms of the number of function evaluations, they are either cumbersome to implement [2] or need special consideration to be given to the boundary elements [3]. In an effort to deal with the overhead in the computation of IGA characteristic matrices, collocation methods have been introduced for obtaining higher order accurate methods with a minimum number of quadrature points [5].

Applications of graphics processing units (GPUs) to scientific computations are attracting a lot of attention due to their low cost in conjunction with their inherently remarkable performance features. Driven by the demands of the gaming industry, graphics hardware has substantially evolved over the years with remarkable floating point arithmetic performance. Unlike CPUs, GPUs have an inherent parallel throughput architecture that focuses on executing many concurrent threads slowly, rather than executing a single thread very fast.

A number of studies in engineering applications have been recently reported on a variety of GPU platforms using implicit computational algorithms [6]–[16]. Linear algebra applications have also been a topic of scientific interest for GPU implementations [17]–[20]. A hybrid CPU-GPU implementation of domain decomposition methods is presented in [21] where speedups of the order of 40x have been achieved with just one GPU.

The present work achieves a drastic reduction of the computational effort required for assembling the stiffness matrix of IGA by implementing a novel control point pair-wise procedure recently proposed for the computation of the stiffness matrix in element-free Galerkin formulations [22]. This approach is amenable to parallel computations since it does not have race conditions or need synchronization and it is particularly suitable for massively parallel systems with GPUs. The numerical results indicate that the proposed methodology succeeds in overcoming the drawback of the quadrature cost associated with IGA by performing the assembly of the stiffness matrix in orders of magnitude less computation time than that of the standard element-wise Gauss quadrature scheme.

2 BASIC INGREDIENTS OF THE ISOGEOMETRIC ANALYSIS METHOD

2.1 Non-Uniform Rational B-SPLines (NURBS)

In IGA there is no approximate mesh since, even in the case of very coarse meshes, the exact geometry is always represented. For the implementation of IGA three spaces should be defined: the physical space, the parameter space and the index space. For NURBS shape functions, the parameter space is very important as all calculations refer there, while the index space plays an auxiliary role. The input data is drawn from the physical space, which contains the Cartesian coordinates of the control points and their corresponding weights. The number of basis functions is equal to the number of degrees of freedom. The unknowns of the resulting algebraic equations correspond to the displacements of the control points, while the knots are the boundaries of the corresponding isogeometric elements. In the case of uniform knot vector, knot spans have the same size in the parameter space while in the physical space they can have any size depending on the corresponding control points and shape functions. The discretized NURBS-model is subdivided into patches which are subdomains with the same material and geometry type and consist of a full tensor product grid of elements. In this respect, they are analogous to elements in FEA as the basis functions are interpolatory at its boundaries.

A knot vector is a non-decreasing set of coordinates in the parameter space, written as $\Xi=\{\xi_1,\xi_2,\ldots,\xi_{n+p+1}\}$, where $\xi_i\in\mathbb{R}$ is the ith knot, i is the knot index, $i=1,2,\ldots,n+p+1$, p is the polynomial order and n is the number of basis functions used to construct the B-SPLine curve. The knots partition the parameter space into elements. Element boundaries in the physical space are simply the images of knot lines under the B-SPLine mapping. Figure 1 illustrates the quadratic C^1 continuous B-SPLine basis functions, which are produced by the open uniform knot vector $\Xi=\{0,0,0,1,2,3,4,5,6,7,8,9,9,9\}$. Control points are shown as circles, while knots as rectangles. The interval $[0,9]$ is a single patch and consists of 9 elements and 11 control points, which correspond to 11 B-SPLine basis functions.

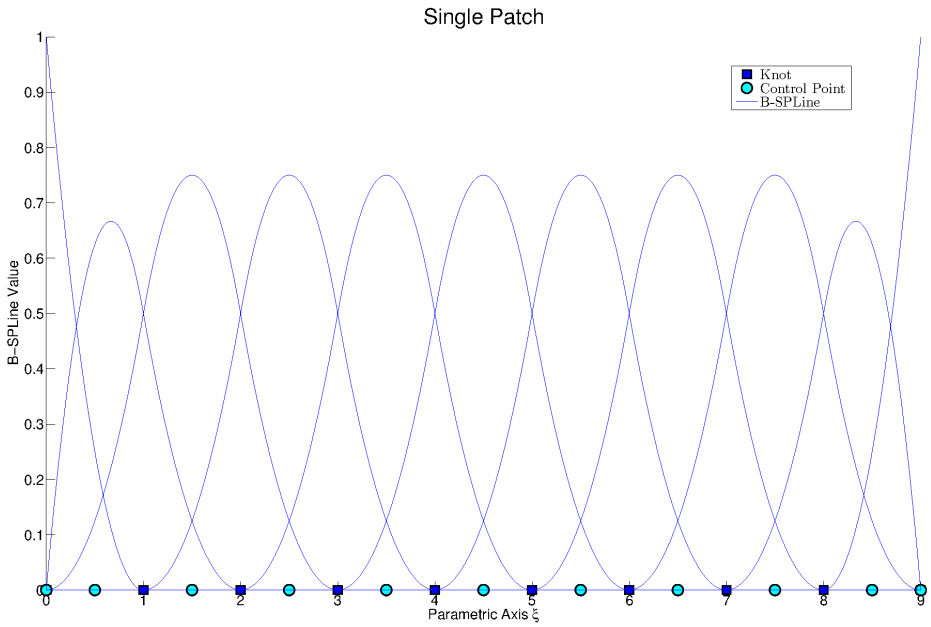

Fig. 1. C^1 continuous quadratic basis derived from open uniform knot vector

$$\Xi=\{0,0,0,1,2,3,4,5,6,7,8,9,9,9\}$$

Given an open uniform knot vector $\Xi=\{\xi_1,\xi_2,\ldots,\xi_{n+p+1}\}$, the B-SPLine basis functions $N_i^p(\xi)$ are defined by the Cox-de Boor recursion formula:

$$N_i^0(\xi)=\begin{cases}1, & if\ \xi_i\le\xi<\xi_{i+1}\\ 0, & otherwise\end{cases} \tag{1}$$

$$N_i^p(\xi)=\frac{\xi-\xi_i}{\xi_{i+p}-\xi_i}N_i^{p-1}(\xi)+\frac{\xi_{i+p+1}-\xi_i}{\xi_{i+p+1}-\xi_i}N_{i+1}^{p-1}(\xi) \tag{2}$$

Due to their higher regularity between inter-element boundaries, they exhibit greater overlapping in comparison with the shape functions of FEA. Their basic feature is their tensor product nature. In the case of polynomial B-SPLines, basis functions are used as shape functions, while in the case of NUBRS, shape functions are produced from the following formula in 1D case:

$$R_i^p(\xi)=\frac{N_i^p W_i}{\sum_{i=1}^{n}\left[N_i^p(\xi)W_i\right]} \tag{3}$$

in the 2D case:

$$R_{i,j}^{p,q}(\xi,\eta)=\frac{N_i^p(\xi)M_j^q(\eta)W_{i,j}}{\sum_{i=1}^{n}\sum_{j=1}^{m}N_i^p(\xi)M_j^q(\xi)W_{i,j}} \tag{4}$$

in the 3D case:

$$R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)=\frac{N_i^p(\xi)M_j^q(\eta)L_k^r(\zeta)W_{i,j,k}}{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l}N_i^p(\xi)M_j^q(\xi)L_k^r(\zeta)W_{i,j,k}} \tag{5}$$

where W_i are weight factors with a full tensor product nature:

$$W_{i,j}=W_iW_j \tag{6}$$

$$W_{i,j,k}=W_iW_jW_k \tag{7}$$

The approximation of 1D displacement field in terms of control point variables can be written as

$$u(\xi)=\sum_{i=1}^{n}\left\{R_i^p(\xi)u_{CPi}\right\} \tag{8}$$

where $R_i^p(\xi)$ are the shape functions, n is the number of basis functions or control points, p is the polynomial order and u_{CPi} is the displacement of control point i. The exact geometry is described by

$$X(\xi)=\sum_{i=1}^{n}\left\{R_i^p(\xi)X_{CPi}\right\} \tag{9}$$

where X_{CPi} are the Cartesian coordinate of the control point i.

There is a connection between polynomial basis order p, knot multiplicity m and continuity/ regularity k, given by

$$k=p-m,\quad 1\le m\le p+1 \tag{10}$$

Regularity -1 means discontinuity and it appears for the extreme knots of a single patch. In this case, basis functions are interpolatory at these extreme knots. Regularity 0 resembles to the case of finite elements and is the minimum continuity for interior knots with basis functions interpolatory at that knots. The case of maximum continuity is $p-1$ and occurs when every interior knot is repeated only once.

Assuming an one-dimensional case, polynomial order p, multiplicity m and number of elements n^{el}, the corresponding number of control points n, which are directly linked to the number of degrees of freedom, is equal to

$$n=(p+1)n^{el}-(k-1)(n^{el}-1) \tag{11}$$

The corresponding knot vector has $n+p+1$ knot values. The external knots are repeated $p+1$ times and the interior m times.

2.2 Stiffness matrix formulation

A given domain is represented with several NURBS-based isogeometric models, depending on its geometry features. Every NURBS-based models decomposed into subdomains, the so-called patches, according to the variance of its geometry and material. The more abrupt the geometry is, the more subdomains are considered. They can be assumed as macro-elements consisting of a tensor product mesh of elements and they are assembled in the same way as in finite elements. The arrays for the patches are constructed and assembled in element-by-element fashion by numerically integrating contributions over each element. In the parameter space, elements are rectangular.

The equilibrium equations applied to control points of the whole domain are expressed as

$$\mathbf{K}\,\mathbf{u}=\mathbf{f} \tag{12}$$

In order to formulate the total stiffness matrix, the stiffness matrix of every patch $i=1,\ldots,N_p$ has to be calculated:

$$\mathbf{K}^i=\int_V \left(\mathbf{B}^i\right)^T \mathbf{E}^i\,\mathbf{B}^i\,dV=\iiint_{\xi,\eta,\zeta} \left(\mathbf{B}^i\right)^T \mathbf{E}^i\,\mathbf{B}^i\,det\,\mathbf{J}^i\,d\xi d\eta d\zeta \tag{13}$$

where $\mathbf{E}^i$, $\mathbf{B}^i$ are the elasticity and deformation matrix of the patch i respectively.

We will present below the stiffness matrix formulation in 2D elasticity cases. For the 3D case, the formulation is analogous. Assuming n, m control points per parametric axis ξ, η respectively, the 2D control points are $N=nm$ (full tensor product) and the deformation matrix $\mathbf{B}$ is given by:

$$\underset{(3\times N)}{\mathbf{B}}=\underset{(3\times 4)}{\mathbf{B}_1}\;\underset{(4\times N)}{\mathbf{B}_2} \tag{14}$$

with

$$\underset{(3\times 4)}{\mathbf{B}_1}=\frac{1}{det\,\mathbf{J}(\xi)}\begin{bmatrix} J_{22} & -J_{12} & 0 & 0 \\ 0 & 0 & -J_{21} & J_{11} \\ -J_{21} & J_{11} & J_{22} & -J_{12} \end{bmatrix} \tag{15}$$

and

$$\underset{(4\times N)}{\mathbf{B}_2}=\begin{bmatrix} R_{1,\xi} & 0 & R_{2,\xi} & 0 & \ldots & R_{N,\xi} & 0 \\ R_{1,\eta} & 0 & R_{2,\eta} & 0 & \ldots & R_{N,\eta} & 0 \\ 0 & R_{1,\xi} & 0 & R_{2,\xi} & 0 & \ldots & R_{N,\xi} \\ 0 & R_{1,\eta} & 0 & R_{2,\eta} & 0 & \ldots & R_{N,\eta} \end{bmatrix} \tag{16}$$

The Jacobian matrix is local to patches rather than to elements and is given by

$$\underset{(2\times 2)}{\mathbf{J}(\xi,\eta)}=\underbrace{\begin{bmatrix} R_{1,\xi}(\xi,\eta) & R_{2,\xi}(\xi,\eta) & \ldots & R_{N,\xi}(\xi,\eta) \\ R_{1,\eta}(\xi,\eta) & R_{2,\eta}(\xi,\eta) & \ldots & R_{N,\eta}(\xi,\eta) \end{bmatrix}}_{(2\times N)}\underbrace{\begin{bmatrix} X_{CP1} & Y_{CP1} \\ X_{CP2} & Y_{CP2} \\ \vdots & \vdots \\ X_{CPN} & Y_{CPN} \end{bmatrix}}_{(N\times 2)}=\underset{(2\times 2)}{\begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}} \tag{17}$$

where $R_l(\xi,\eta)$ is the shape function that corresponds to the control point l, with Cartesian coordinates X_{CPl}, Y_{CPl}, and

$$R_{l,\xi}(\xi,\eta)=\frac{dR_l(\xi,\eta)}{d\xi},\quad R_{l,\eta}(\xi,\eta)=\frac{dR_l(\xi,\eta)}{d\eta} \tag{18}$$

The derivatives in eq. (18) are obtained by applying the quotient rule to eq. (4).

2.3 Quadrature rule

The Gauss quadrature rule is applied to the non-zero knot spans as in FEA. However, the standard element-wise Gauss rule requires extensive function evaluations due to increased support of the shape functions. According to [3], for the case of an one-dimensional function of order p the optimal (minimum exact) number of Gauss points per element is equal to $(p+1)/2$ or $(p+2)/2$, for odd and even p respectively. For the computation of the stiffness matrix in 1D elasticity case, the integrand's order is equal to $q=2p-2$ and the optimal number of Gauss points per element is equal to $(q+2)/2=p$. For 2D and 3D elasticity cases, the integrand's order is equal to $q=2p$ and the optimal number of Gauss

points per element is equal to $(q+2)/2=p+1$. The above rules are optimal for the case of minimum continuity. For higher continuity, new macro-element rules have been proposed [2], [3], which are more efficient, but also more involved and difficult to implement.

3 ELEMENT-WISE FORMULATION OF THE STIFFNESS MATRIX

In order to build the stiffness matrix of a domain (or patch), the contributions of all Gauss points need to be added. The contributions are expressed by the products $\mathbf{B}_G^T\mathbf{E}\mathbf{B}_G$

$$\mathbf{K}=\sum_G \mathbf{B}_G^T\mathbf{E}\,\mathbf{B}_G=\sum_G \mathbf{Q}_G \tag{19}$$

where the deformation matrix $\mathbf{B}_G$ is computed at the corresponding Gauss point.

However, instead of adding each Gauss contribution to the global stiffness matrix, the standard procedure is to first build the stiffness matrix of each element by adding the contributions of all Gauss points G of the element to its local stiffness matrix:

$$\mathbf{K}_E=\sum \mathbf{B}_G^T\mathbf{E}\,\mathbf{B}_G=\sum_{E_G} \mathbf{Q}_G \tag{20}$$

S ent modify obal
stiff ditions loca tive
cont is benefici is
subs ffness matrix

$$=\sum_E \mathbf{K}_E \tag{21}$$

T ch Gauss poi nain
of i amount of k is
sign of influence the
corr eas influenci 1D
case nparison bet ntrol
poin l in Fig. 2, fo ould
be r between co ents
prov ng" so the co ore.
Thro lress the issu two
metl ol points or e

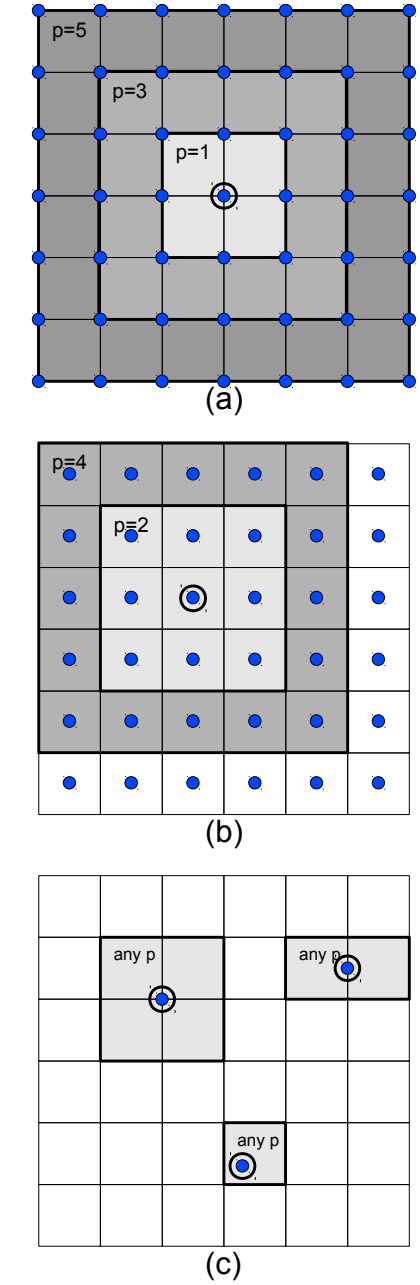

Fig. 2. Areas influencing a control point in (a) IGA (p odd); (b) IGA (p even); (c) FEA. The influencing entities are the Gauss points in the shaded areas.

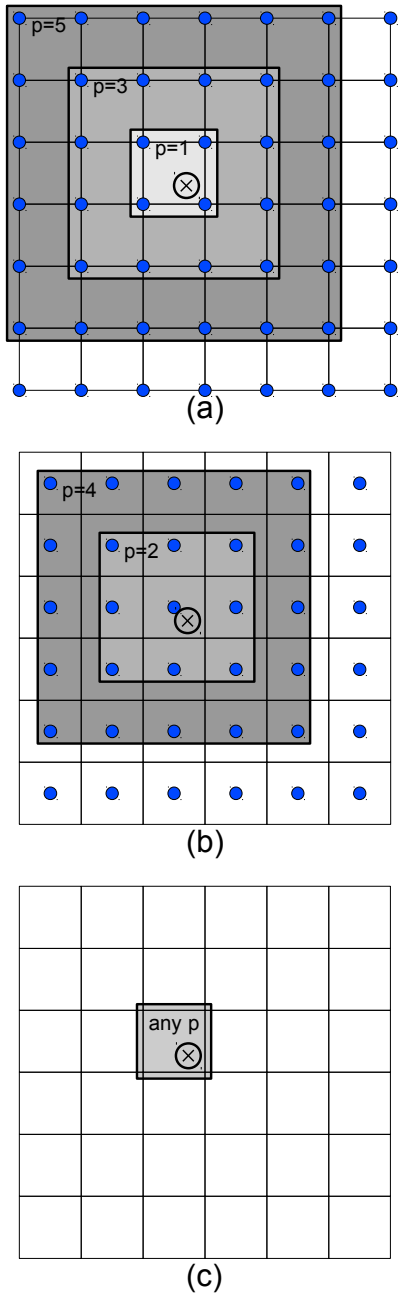

Fig. 3. Control points/nodes influenced by Gauss point in (a) IGA (p odd); (b) IGA (p even); (c) FEA.

In FEA, each Gauss point is involved in computations with nodes within its own element. The shape functions and their derivatives are predefined for each element type and need to be evaluated on all combinations of nodes and Gauss points within the element. In IGA, however, each Gauss point is involved in computations with control points of surrounding areas as well (Fig. 3), while the shape functions are not predefined and span across larger domains with a significantly higher amount of Gauss-control point interactions.

The domain stiffness matrix is large and therefore needs to be stored with appropriate efficient storage methods. Depending on the method used for the solution of the resulting equations and constraints imposed by the hardware (e.g. memory limit), it may be in sparse format, skyline format, multi-diagonal format (since the grid is structured), etc. Throughout this work a sparse matrix format specifically tailored for the assembly phase is used.

For equivalent meshes, the bandwidth is the same between the two methods but IGA has a larger amount of interactions and, consequently, denser stiffness matrices. Furthermore, the computation of each non-zero coefficient is more laborious because the control point pairs have a lot more shared elements (on average) and consequently significantly more Gauss point contributions.

3.1 Performance of the element-wise approach

The examples that will be presented through this work are detailed in Table 1, while the performance of the element-wise (EW) approach in the CPU is shown in Table 2. The proposed EW approach is compared with the "conventional" one without several improvements and code optimizations. The examples are run on a Core i7-980X which has 6 physical cores (12 logical cores) at 3.33GHz and 12MB cache. The examples have no trivial knot spans in order to maximize the number of calculations.

	Example	p	n	Control points	dof	Elements	Gauss points
2D	P2-1	2	225	50,625	101,250	49,729	447,561
	P2-2	2	500	250,000	500,000	248,004	2,232,036
	P2-3	2	633	400,689	801,378	398,161	3,583,449
	P3-1	3	225	50,625	101,250	49,284	788,544
	P3-2	3	320	102,400	204,800	100,489	1,607,824
	P3-3	3	388	150,544	301,088	148,225	2,371,600
	P4-1	4	160	25,600	51,200	24,336	608,400
	P4-2	4	225	50,625	101,250	48,841	1,221,025
	P4-3	4	275	75,625	151,250	73,441	1,836,025
3D	P2-1	2	19	6,859	20,577	4,913	132,651
	P2-2	2	26	17,576	52,728	13,824	373,248
	P2-3	2	33	35,937	107,811	29,791	804,357
	P3-1	3	19	6,859	20,577	4,096	262,144
	P3-2	3	21	9,261	27,783	5,832	373,248
	P3-3	3	26	17,576	52,728	12,167	778,688
	P4-1	4	15	3,375	10,125	1,331	166,375
	P4-2	4	17	4,913	14,739	2,197	274,625
	P4-3	4	19	6,859	20,577	3,375	421,875

Table 1. Example details of 2D square ($n \times n$) and 3D cubic ($n \times n \times n$) domains.

As shown in Table 2, by optimizing the element-wise method, we were able to achieve about 3X and 5X speedup in 2D and 3D cases, respectively.

Example		dof	CPU Time (seconds)		Speedup Ratio
			Conventional element wise	Proposed element wise	
2D	P2-1	101,250	14	5	2.7
	P2-2	500,000	60	20	2.9
	P2-3	801,378	96	32	3.0
	P3-1	101,250	41	14	2.9
	P3-2	204,800	83	27	3.1
	P3-3	301,088	124	39	3.1
	P4-1	51,200	56	19	3.0
	P4-2	101,250	113	36	3.1
	P4-3	151,250	171	57	3.0
3D	P2-1	20,577	37	8	4.8
	P2-2	52,728	98	21	4.8
	P2-3	107,811	212	43	5.0
	P3-1	20,577	305	59	5.2
	P3-2	27,783	430	83	5.2
	P3-3	52,728	900	168	5.3
	P4-1	10,125	635	131	4.9
	P4-2	14,739	1,055	218	4.9
	P4-3	20,577	1,599	333	4.8

Table 2. Computing time for the formulation of the stiffness matrix in the CPU implementations of the element-wise approach.

4 CONTROL POINT PAIR-WISE FORMULATION OF THE STIFFNESS MATRIX

An alternative way to perform the computation of the global stiffness matrix is the proposed control point pair-wise approach. The computation of the global stiffness coefficient $\mathbf{K}_{ij}$ is performed for all interacting $i-j$ control points and is formed from contributions by the shared Gauss points of their domains of influence. Two control points are interacting if there is at least one Gauss point that influences both control points. In IGA, it is more convenient to define two control points as being interacting if there is at least one element shared by both control points, but care must be taken in cases where there are trivial knot spans which have no Gauss points.

4.1 Interacting control point pairs and their shared elements

The interacting control point pairs approach initially identifies a) the interacting control point pairs and b) the shared elements of the interacting control point pairs. The interacting control points associated with a specific control point are those located within a fixed range dictated by the order p of each axis.

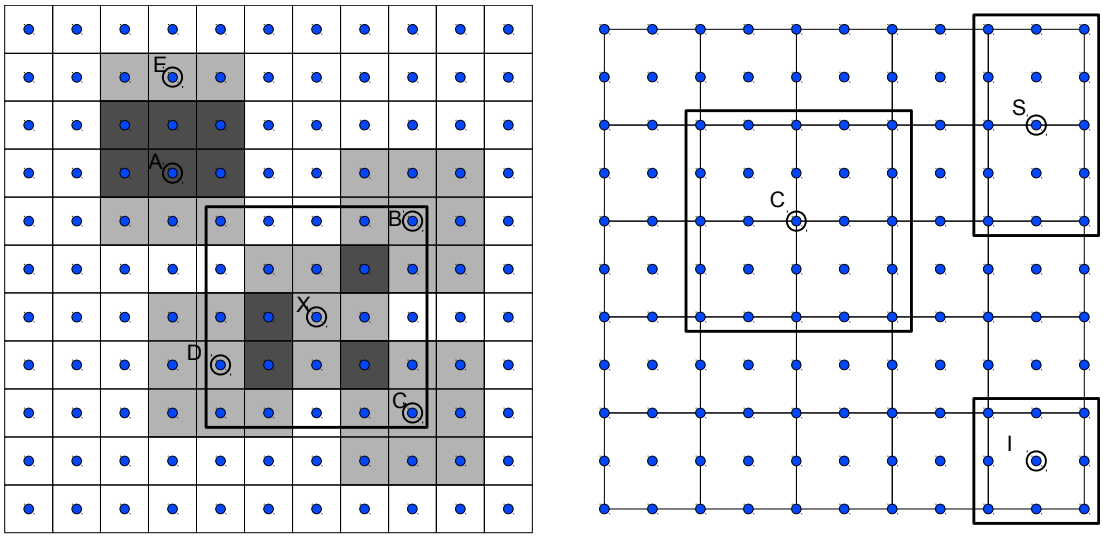

Fig. 4. Interacting control points/nodes for p = 2: (a) IGA; (b) FEA.

In FEA the nodes interact through neighboring elements only and thus the interacting node pairs can be easily defined from the element-node connectivity (Fig. 4b). In IGA, however, a control point pair contributes non-zero entries to the stiffness matrix, and therefore is active, if there is at least one (non-empty) element shared between the two control points (Fig. 4a). Thus, control point X interacts with B,C,D, but not with A or E. If the basis order is p, then the interacting control points extend up to p elements in all directions. This can be observed for $p=2$ in Fig. 4a. The gray shaded regions are the influence domains of each control point. The thick-lined rectangles in Fig. 4 include all control points/nodes that are interacting with the corresponding control point/node.

4.2 Computation of global stiffness coefficients for each interacting control point pair

The computation of the stiffness elements for each interacting control point pair is split in two phases. In the first phase, the shape function derivatives for each influenced control point of every Gauss point are calculated as in the element-wise method. In the second phase, instead of continuing with the calculation of the stiffness matrix coefficients corresponding to a particular element, the stiffness matrix coefficients of each interacting control point pair is computed. Both phases are amenable to parallelization.

4.3 Performance of the control point pair-wise approach

Table 3 shows the time required for the two phases with the control point pair-wise (CPPW) approach as well as the total computing time for the formulation of the stiffness matrix. Comparing the serial CPU times with the corresponding values of the element-wise (EW) approach (Table 2), it can be observed that the CPPW approach performs better than the EW approach. Furthermore, contrary to the EW approach, the CPPW approach is amenable to parallelization, even in massively parallel processors, as will be subsequently discussed.

Example		dof	CPU Time (seconds)		
			Shape Functions	Assembly	Total
2D	P2-1	101,250	2	2	4
	P2-2	500,000	10	6	17
	P2-3	801,378	17	10	27
	P3-1	101,250	7	5	12
	P3-2	204,800	13	10	23
	P3-3	301,088	18	16	34
	P4-1	51,200	8	8	16
	P4-2	101,250	15	17	31
	P4-3	151,250	21	26	47
3D	P2-1	20,577	2	4	7
	P2-2	52,728	6	10	17
	P2-3	107,811	12	23	36
	P3-1	20,577	9	33	42
	P3-2	27,783	12	46	58
	P3-3	52,728	24	95	119
	P4-1	10,125	11	73	84
	P4-2	14,739	18	123	141
	P4-3	20,577	27	184	211

Table 3. Computing time for the formulation of the stiffness matrix in the serial CPU implementation of the control point pair-wise (CPPW) approach.

4.4 Parallelization features of the interacting control point pair-wise approach

The CPPW approach has certain advantages compared to the EW approach, the most important one being its amenability to parallelism. In the element-wise approach, each element contributes to different stiffness coefficients and the coefficients are continuously updated with the contributions of each element. The final value of a particular coefficient is formed once all contributions have been considered. Therefore, parallelizing the element-wise approach involves scatter parallelism, which is schematically shown in Fig. 5 for two elements C and D. Each part of the sum can be calculated in parallel but there are conflicting updates to the same coefficients of the stiffness matrix. These race conditions can be avoided with proper synchronization but in massively parallel systems like the GPU where thousands of threads may be working concurrently it is very detrimental to performance because all updates are serialized with atomic operations [23].

In the CPPW approach, the final values for the submatrix of each interacting control point pair are calculated and appended to the matrix, instead of constantly updating the matrix. For the calculation of a submatrix, all contributions of the Gauss points belonging to the intersection of the the domains of influence of two interacting control points should be summed together. Thus, the interacting control point pairs approach utilizes gather parallelism as shown schematically in Fig. 6.

In a parallel implementation, each working unit, i.e. a thread or group of threads, prepares a submatrix $\mathbf{K}_{ij}$ related to a specific interacting control point pair ij. It gathers all contributions from the Gauss points and writes to a specific memory location accessed by no other thread. Thus, this method requires no synchronization or atomic operations.

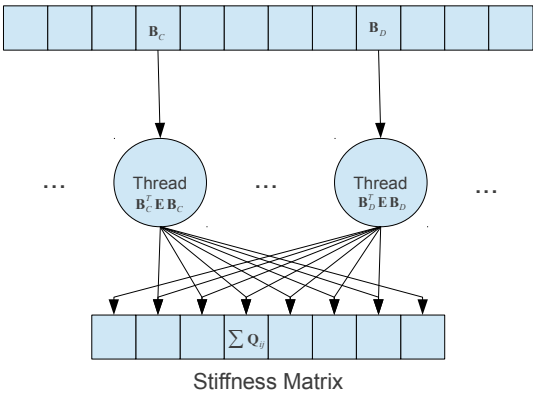

Fig. 5. Scatter parallelism required for the element-wise approach.

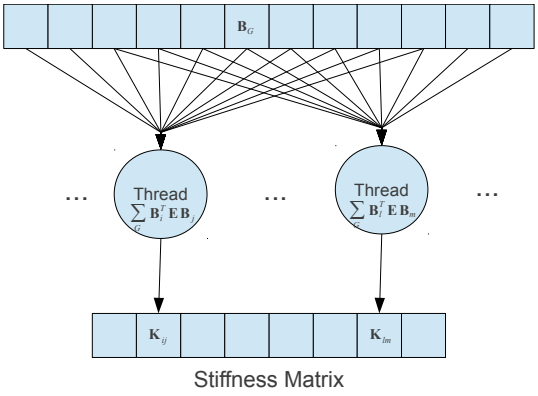

Fig. 6. Gather parallelism implemented in the interacting control point pairs approach.

4.5 Performance of the GPU implementations of the control point pair-wise approach

Table 4 shows the time needed for the GPU implementation. The results utilize one GPU, a GeForce GTX680 with 1536 CUDA cores and 2GB GDDR5 memory.

	Example	dof	CPPW GPU Time (seconds)		
			Kernel 1	Kernel 2	Total
2D	P2-1	101,250	0.05	0.08	0.1
	P2-2	500,000	0.23	0.34	0.6
	P2-3	801,378	0.35	0.53	0.9
	P3-1	101,250	0.13	0.18	0.3
	P3-2	204,800	0.23	0.35	0.6
	P3-3	301,088	0.36	0.51	0.9
	P4-1	51,200	0.21	0.20	0.4
	P4-2	101,250	0.38	0.38	0.8
	P4-3	151,250	0.60	0.54	1.1
3D	P2-1	20,577	0.07	0.10	0.2
	P2-2	52,728	0.18	0.25	0.4
	P2-3	107,811	0.38	0.51	0.9
	P3-1	20,577	0.37	0.56	0.9
	P3-2	27,783	0.51	0.78	1.3
	P3-3	52,728	1.03	1.61	2.6
	P4-1	10,125	0.49	1.07	1.6
	P4-2	14,739	0.80	1.79	2.6
	P4-3	20,577	1.19	2.68	3.9

Table 4. Computing time for the formulation of the stiffness matrix in the coalesced GPU implementation of the interacting control point-pair approach with a GTX 680.

5 NUMERICAL RESULTS FOR 2D AND 3D ELASTICITY PROBLEMS

Throughout this work, the two approaches for the computation of the stiffness matrix are tested in 2D and 3D elasticity problems. The geometric domains and parameters of these problems maximize the number of correlations and consequently the computational cost for the given number of control points . The examples are run on the following hardware. CPU: Core i7-980X which has 6 physical cores (12 logical cores) at 3.33 GHz and 12MB cache. GPU: GeForce GTX680 with 1536 CUDA cores and 2GB GDDR5 memory.

The performance of the serial element-wise (EW) and control point pair-wise (CPPW) approaches in the CPU is given in Table 5. These implementations have been explored in Sections 3.1 and 4.3 for the two approaches, respectively.

Example		dof	CPU Time (seconds)		
			Conventional EW	Proposed EW	Proposed CPPW
2D	P2-1	101,250	14	5	4
	P2-2	500,000	60	20	17
	P2-3	801,378	96	32	27
	P3-1	101,250	41	14	12
	P3-2	204,800	83	27	23
	P3-3	301,088	124	39	34
	P4-1	51,200	56	19	16
	P4-2	101,250	113	36	31
	P4-3	151,250	171	57	47
3D	P2-1	20,577	37	8	7
	P2-2	52,728	98	21	17
	P2-3	107,811	212	43	36
	P3-1	20,577	305	59	42
	P3-2	27,783	430	83	58
	P3-3	52,728	900	168	119
	P4-1	10,125	635	131	84
	P4-2	14,739	1,055	218	141
	P4-3	20,577	1,599	333	211

Table 5. Computing time for the formulation of the stiffness matrix in the serial CPU implementations of the element-wise (EW) and node pair-wise (CPPW) approaches

The performance of the GPU implementations of the control point pair-wise method is shown in Table 4. Speedup ratios of the GPU implementation compared to the CPU implementations is given in Table 6.

Example		dof	Speedup ratios of GPU implementation		
			Conventional EW	Proposed EW	Proposed CPPW
2D	P2-1	101,250	107	39	33
	P2-2	500,000	106	36	30
	P2-3	801,378	109	36	30
	P3-1	101,250	134	46	39
	P3-2	204,800	143	46	40
	P3-3	301,088	143	46	39
	P4-1	51,200	141	47	40
	P4-2	101,250	150	49	42
	P4-3	151,250	151	50	42
3D	P2-1	20,577	222	46	41
	P2-2	52,728	234	49	40
	P2-3	107,811	241	49	41
	P3-1	20,577	329	63	45
	P3-2	27,783	334	64	45
	P3-3	52,728	340	64	45
	P4-1	10,125	407	84	54
	P4-2	14,739	409	84	54
	P4-3	20,577	413	86	54

Table 6. Relative speedup ratios of GPU (GTX 680) implementation compared to the CPU (Core i7-980X) implementations.

6 CONCLUDING REMARKS

The proposed control point pair-wise approach has several benefits over the element-wise approach. The most important one is its amenability to parallelism especially in massively parallel systems like the GPUs. Each control point pair can be processed separately by any available processor in order to compute the corresponding stiffness submatrix. The control point pair approach can be characterized as “embarrassingly parallel” since it requires no synchronization whatsoever between node pairs.

A GPU implementation is applied to the control point pair-wise approach offering significant speedups compared to CPU implementations. The granularity of the control point pair-wise approach offers ample parallelism and results in high hardware utilization which is evidenced by speedup ratios achieved with just one GPU in the test examples presented. The control point pair-wise approach can be applied as is to any available hardware achieving even lower computing times. This includes using many GPUs, hybrid CPU(s)/GPU(s) implementations and generally any available processing unit. The importance of this portability becomes apparent when considering contemporary and future developments like heterogeneous systems architecture (HSA).

In conclusion, the parametric tests performed in the framework of this study showed that with the proposed implementation along with the exploitation of currently available low cost hardware, the expensive formulation of the stiffness matrix in IGA methods can be reduced by orders of magnitude. The presented control point pair-approach enables the efficient utilization of any available hardware and can accomplish high speedup ratios, which convincingly addresses a shortcoming of isogemetric analysis, making it computationally competitive in solving large-scale problems in computational mechanics.

7 ACKNOWLEDGMENTS

This work has been supported by the European Research Council Advanced Grant "MASTER – Mastering the computational challenges in numerical modeling and optimum design of CNT reinforced composites" (ERC-2011-ADG_20110209). The first author has also been supported by the John Argyris Foundation and the second author by the Alexander S. Onassis Foundation.

8 REFERENCES

[1] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs, "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 39–41, pp. 4135–4195, 2005.

[2] F. Auricchio, F. Calabrò, T. J. R. Hughes, A. Reali, and G. Sangalli, "A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 249–252, pp. 15–27, 2012.

[3] T. J. R. Hughes, A. Reali, and G. Sangalli, "Efficient quadrature for NURBS-based isogeometric analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 199, no. 5–8, pp. 301–313, 2010.

[4] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs, *Isogeometric Analysis: Toward Integration of CAD and FEA*, 1st ed. Wiley, 2009.

[5] F. Auricchio, L. Beirão da Veiga, T. J. R. Hughes, A. Reali, and G. Sangalli, "Isogeometric collocation for elastostatics and explicit dynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 249–252, pp. 2–14, 2012.

[6] I. C. Kampolis, X. S. Trompoukis, V. G. Asouti, and K. C. Giannakoglou, "CFD-based analysis and two-level aerodynamic optimization on graphics processing units," *Computer Methods in Applied Mechanics and Engineering*, vol. 199, no. 9–12, pp. 712–722, 2010.

[7] E. Elsen, P. LeGresley, and E. Darve, "Large calculation of the flow over a hypersonic vehicle using a GPU," *Journal of Computational Physics*, vol. 227, no. 24, pp. 10148–10161, 2008.

[8] J. C. Thibault and I. Senocak, "Accelerating incompressible flow computations with a Pthreads-CUDA implementation on small-footprint multi-GPU platforms," 2010.

[9] M. de la Asunción, J. M. Mantas, and M. J. Castro, "Simulation of one-layer shallow water systems on multicore and CUDA architectures," 2010.

[10] H. Zhou, G. Mo, F. Wu, J. Zhao, M. Rui, and K. Cen, "GPU implementation of lattice Boltzmann method for flows with curved boundaries," *Computer Methods in Applied Mechanics and Engineering*, vol. 225–228, pp. 65–73, 2012.

[11] A. Sunarso, T. Tsuji, and S. Chono, "GPU-accelerated molecular dynamics simulation for study of liquid crystalline flows," *Journal of Computational Physics*, vol. 229, no. 15, pp. 5486–5497, 2010.

[12] J. A. Anderson, C. D. Lorenz, and A. Travesset, "General purpose molecular dynamics simulations fully implemented on graphics processing units," *Journal of Computational Physics*, vol. 227, no. 10, pp. 5342–5359, 2008.

[13] E. Wadbro and M. Berggren, "Megapixel topology optimization on a graphics processing unit," *SIAM Review*, vol. 51, no. 4, pp. 707–721, 2009.

[14] D. Komatitsch, G. Erlebacher, D. Göddeke, and D. Michéa, "High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster," *Journal of*

Computational Physics, vol. 229, no. 20, pp. 7692–7714, 2010.
[15] T. Takahashi and T. Hamada, "GPU-accelerated boundary element method for Helmholtz' equation in three dimensions," *International Journal for Numerical Methods in Engineering*, vol. 80, no. 10, pp. 1295–1321, 2009.
[16] G. R. Joldes, A. Wittek, and K. Miller, "Real-time nonlinear finite element computations on GPU - Application to neurosurgical simulation," *Computer Methods in Applied Mechanics and Engineering*, vol. 199, no. 49–52, pp. 3305–3314, 2010.
[17] S. Tomov, J. Dongarra, and M. Baboulin, "Towards dense linear algebra for hybrid GPU accelerated manycore systems," *Parallel Computing*, vol. 36, no. 5–6, pp. 232–240, 2010.
[18] O. Schenk, M. Christen, and H. Burkhart, "Algorithmic performance studies on graphics processing units," *Journal of Parallel and Distributed Computing*, vol. 68, no. 10, pp. 1360–1369, 2008.
[19] J. M. Elble, N. V. Sahinidis, and P. Vouzis, "GPU computing with Kaczmarz's and other iterative algorithms for linear systems," *Parallel Computing*, vol. 36, no. 5–6, pp. 215–231, 2010.
[20] A. Cevahir, A. Nukada, and S. Matsuoka, "High performance conjugate gradient solver on multi-GPU clusters using hypergraph partitioning," *Computer Science - Research and Development*, vol. 25, no. 1–2, pp. 83–91, 2010.
[21] M. Papadrakakis, G. Stavroulakis, and A. Karatarakis, "A new era in scientific computing: Domain decomposition methods in hybrid CPU-GPU architectures," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 13–16, pp. 1490–1508, 2011.
[22] A. Karatarakis, P. Metsis, and M. Papadrakakis, "GPU-Acceleration of stiffness matrix calculation and efficient initialization of EFG meshless methods," *Computer Methods in Applied Mechanics and Engineering*, vol. 258, pp. 63–80, 2013.
[23] W. W. Hwu and D. B. Kirk, "Parallelism Scalability," in *Programming and tUning Massively Parallel Systems (PUMPS)*, Barcelona, 2011.