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

STIFFNESS MATRIX COMPUTATION FOR ELEMENT FREE GALERKIN METHODS ON GPU

Alexander Karatarakis[1], Panagiotis Metsis[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
pmetsis@gmail.com, mpapadra@central.ntua.gr

Keywords: Meshless methods, Element Free Galerkin (EFG), preprocessing, stiffness matrix assembly, parallel computing, GPU acceleration

Abstract. *Meshless methods have a number of virtues in problems concerning crack growth and propagation, large displacements, strain localization and complex geometries, among other. Despite the fact that they do not rely on a mesh, meshless methods require a preliminary step for the identification of the correlation between nodes and Gauss points before building the stiffness matrix. This is implicitly performed with the mesh generation in FEM but must be explicitly done in EFG methods and can be time-consuming. Furthermore, the resulting matrices are more densely populated and the computational cost for the formulation and solution of the problem is much higher than the conventional FEM. This is mainly attributed to the vast increase in interactions between nodes and integration points due to their extended domains of influence. For these reasons, computing the stiffness matrix in EFG meshless methods is a very computationally demanding task which needs special attention in order to be affordable in real-world applications. In this paper, we address the pre-processing phase, dealing with the problem of defining the necessary correlations between nodes and Gauss points and between interacting nodes, as well as the computation of the stiffness matrix. A novel approach is proposed for the formulation of the stiffness matrix which exhibits several computational merits, one of which is its amenability to parallelization which allows the utilization of graphics processing units (GPUs) to accelerate computations.*

1. INTRODUCTION

In meshless methods (MMs) there is no need to construct a mesh, as in finite element method (FEM), which is often in conflict with the real physical compatibility condition that a continuum possesses [1]. Moreover, stresses obtained using FEM are discontinuous and less accurate while a considerable loss of accuracy is observed when dealing with large deformation problems because of element distortion. Furthermore, due to the underlying structure of the classical mesh-based methods, they are not well suited for treating problems with discontinuities that do not align with element edges. MMs were developed with the objective of eliminating part of the above mentioned difficulties [2]. With MMs, manpower time is limited to a minimum due to the absence of mesh and mesh related phenomena. Complex geometries are handled easily with the use of scattered nodes.

One of the first and most prominent meshless methods is the element free Galerkin (EFG) method introduced by Belytschko et al. [3]. EFG requires only nodal data, no element connectivity is needed to construct the shape functions. However a global background cell structure is necessary for the numerical integration. Moreover, since the number of interactions between nodes and/or integration points is heavily increased, due to large domains of influence, the resulting matrices are more densely populated and the computational cost for the formulation and solution of the problem is much higher than in the conventional FEM [3].

To improve the computational efficiency of MMs, parallel implementations like the MPI parallel paradigm has been used in large scale applications[4], [5] and several alternative methodologies have been proposed concerning the formulation of the problem. The smoothed FEM (SFEM) [6] couples FEM with meshless methods by incorporating a strain smoothing operation used in the mesh-free nodal integration method. The linear point interpolation method (PIM) [7] obtains the partial derivatives of shape functions effortlessly due to the local character of the radial basis functions. A coupled EFG/boundary element scheme [8], taking advantage of both the EFG and the boundary element method. Furthermore, solvers which perform an improved factorization of the stiffness matrix and use special algorithms for realizing the matrix-vector multiplication are proposed in [9], [10]. Divo and Kassab [11] presented a domain decomposition scheme on a meshless collocation method, where collocation expressions are used at each subdomain with artificial created interfaces. Wang et al. [7] presented a parallel reproducing kernel particle method (RKPM), using a particle overlapping scheme which significantly increases the number of shared particles and the time for communicating information between them. Recently, a novel approach for reducing the computational cost of EFG methods is proposed by employing domain decomposition techniques on the physical as well as on the algebraic domains [12]. In that work the solution of the resulting algebraic problems is performed with the dual domain decomposition FETI method with and without overlapping between the subdomains. The non-overlapping scheme has led to a significant decrease of the overall computational cost.

The present work aims at a drastic reduction of the computational effort required for the initialization phase and for assembling the stiffness matrix by implementing a novel node pair-wise procedure. It is believed that with the proposed computational handling of the pre-processing phase and the accelerated formulation of the stiffness matrix, together with recent improvements on the solution of the resulting algebraic equations [12] , MMs are becoming computationally competitive and are expected to demonstrate their inherent advantages in solving real, large-scale engineering problems.

2. BASIC INGREDIENTS OF THE MESHLESS EFG METHOD

The approximation of a scalar function u in terms of Lagrangian coordinates in the meshless EFG method can be written as

$$u(\mathbf{x},t)=\sum_{i\in S}\Phi_i(\mathbf{x})u_i(t) \tag{1}$$

where Φ_i are the shape functions, u_i are the nodal values at particle i located at position $\mathbf{x}_i$, and S is the set of nodes i for which $\Phi_i(\mathbf{x})\neq 0$. The shape functions in eq.(1) are only approximants and not interpolants, since $u_i\neq u(\mathbf{x}_i)$.

The shape functions Φ_i are obtained from the weight coefficients w_i, which are functions of a distant parameter $r=\|\mathbf{x}_i-\mathbf{x}\|/d_i$ where d_i defines the domain of influence (doi) of node i. The domain of influence is crucial to solution accuracy, stability and computational cost, as it defines the degree of continuity between the nodes and the bandwidth of the system matrices.

The approximation u^h is expressed as a polynomial of length m with non-constant coefficients. The local approximation around a point $\bar{\mathbf{x}}$, evaluated at a point $\mathbf{x}$ is given by

$$u_L^h(\mathbf{x},\bar{\mathbf{x}})=\mathbf{p}^T(\mathbf{x})\mathbf{a}(\bar{\mathbf{x}}) \tag{2}$$

where $\mathbf{p}(\mathbf{x})$ is a complete polynomial of length m and $\mathbf{a}(\bar{\mathbf{x}})$ contains non-constant coefficients that depend on $\mathbf{x}$

$$\mathbf{a}(\bar{\mathbf{x}})=\begin{bmatrix}a_0(\mathbf{x}) & a_1(\mathbf{x}) & a_2(\mathbf{x}) & \dots & a_m(\mathbf{x})\end{bmatrix}^T \tag{3}$$

In two dimensional problems, the linear basis $\mathbf{p}(\mathbf{x})$ is given by

$$\mathbf{p}^T(\mathbf{x})=\begin{bmatrix}1 & x & y\end{bmatrix},\ m=3 \tag{4}$$

and the quadratic basis by

$$\mathbf{p}^T(\mathbf{x})=\begin{bmatrix}1 & x & y & x^2 & y^2 & xy\end{bmatrix},\ m=6 \tag{5}$$

The unknown parameters $a_j(\mathbf{x})$ are determined at any point $\mathbf{x}$, by minimizing a functional $J(\mathbf{x})$ defined by a weighted average over all nodes $i\in 1,\dots,n$:

$$J(\mathbf{x})=\sum_{i=1}^{n}w(\mathbf{x}-\mathbf{x}_i)\left[u_L^h(\mathbf{x}_i,\mathbf{x})-u_i\right]^2=\sum_{i=1}^{n}w(\mathbf{x}-\mathbf{x}_i)\left[\mathbf{p}^T(\mathbf{x}_i)\mathbf{a}(\mathbf{x})-u_i\right]^2 \tag{6}$$

where the parameters u_i are specified by the difference between the local approximation $u_L^h(\mathbf{x},\bar{\mathbf{x}})$ and the value u_i while the weight function satisfies the condition $w(\mathbf{x}-\mathbf{x}_i)\neq 0$. An extremum of $J(\mathbf{x})$ with respect to the coefficients $a_j(\mathbf{x})$ can be obtained by setting the derivative of J with respect to $\mathbf{a}(\mathbf{x})$ equal to zero. This condition gives the following relation

$$\mathbf{A}(\mathbf{x})\mathbf{a}(\mathbf{x})=\mathbf{W}(\mathbf{x})\mathbf{u} \tag{7}$$

where

$$\mathbf{A}(\mathbf{x})=\sum_{i=1}^{n}w(\mathbf{x}-\mathbf{x}_i)\mathbf{p}(\mathbf{x}_i)\mathbf{p}^T(\mathbf{x}_i) \tag{8}$$

$$\mathbf{W}(\mathbf{x})=\begin{bmatrix}w(\mathbf{x}-\mathbf{x}_1)\mathbf{p}(\mathbf{x}_1) & w(\mathbf{x}-\mathbf{x}_2)\mathbf{p}(\mathbf{x}_2) & \dots & w(\mathbf{x}-\mathbf{x}_n)\mathbf{p}(\mathbf{x}_n)\end{bmatrix} \tag{9}$$

Solving for $\mathbf{a}(\mathbf{x})$ in eq.(7) and substituting into eq.(2) the approximants u^h can be defined as follows:

$$u^h(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\left[[\mathbf{A}(\mathbf{x})]^{-1}\mathbf{W}(\mathbf{x})\mathbf{u}\right] \tag{10}$$

which together with eq.(1) leads to the derivation of the shape function Φ_i associated with node i at point $\mathbf{x}$:

$$\Phi_i(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})[\mathbf{A}(\mathbf{x})]^{-1}\mathbf{W}(\mathbf{x}_i) \tag{11}$$

A solution of a local problem $\mathbf{A}(\mathbf{x})\mathbf{z} = \mathbf{p}(\mathbf{x})$ of size $m \times m$ is performed whenever the shape functions are to be evaluated. This constitutes a drawback of moving least squares-based (MLS-based) MMs since the computational cost can be substantial and it is possible for the moment matrix $\mathbf{A}(\mathbf{x})$ to be ill conditioned [2].

The Galerkin weak form of the above formulation gives the discrete algebraic equation

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

with

$$\mathbf{K}_{ij} = \int_{\Omega} \mathbf{B}_i^T\,\mathbf{E}\,\mathbf{B}_j\,d\,\Omega \tag{13}$$

$$\mathbf{f}_i = \int_{\Gamma_t} \mathbf{\Phi}_i\,\bar{\mathbf{t}}\,d\,\Gamma + \int_{\Omega} \mathbf{\Phi}_i\,\mathbf{b}\,d\,\Omega \tag{14}$$

In 2D problems matrix $\mathbf{B}$ is given by

$$\mathbf{B}_i = \begin{bmatrix} \Phi_{i,x} & 0 \\ 0 & \Phi_{i,y} \\ \Phi_{i,y} & \Phi_{i,x} \end{bmatrix} \tag{15}$$

and in 3D problems by

$$\mathbf{B}_i = \begin{bmatrix} \Phi_{j,x} & 0 & 0 \\ 0 & \Phi_{j,y} & 0 \\ 0 & 0 & \Phi_{j,z} \\ \Phi_{j,y} & \Phi_{j,x} & 0 \\ 0 & \Phi_{j,z} & \Phi_{j,y} \\ \Phi_{j,z} & 0 & \Phi_{j,x} \end{bmatrix} \tag{16}$$

Due to the lack of the Kronecker delta property of shape functions, the essential boundary conditions cannot be imposed the same way as in FEM. Several techniques are available such as Lagrange multipliers, penalty and EFG and FEM coupling.

For the integration of eq. (13), virtual background cells are considered by dividing the problem domain into integration cells over which a Gaussian quadrature is performed:

$$\int_{\Omega} \mathbf{f}(\mathbf{x})\,d\Omega = \sum_{J} \mathbf{f}(\xi_J)\,\omega_{\Xi}\,det\,J^{\xi}(\xi) \tag{17}$$

where ξ are the local coordinates and $det\,J^{\xi}(\xi)$ is the determinant of the Jacobian.

3. GAUSS POINT-WISE FORMULATION OF THE STIFFNESS MATRIX

The stiffness matrix of eq. (13) is usually formed by adding the contributions of the products $\mathbf{B}_G^T\mathbf{E}\mathbf{B}_G$ of all Gauss points G to the stiffness matrix according to the formula:

$$\mathbf{K} = \sum_{G} \mathbf{B}_G^T\mathbf{E}\,\mathbf{B}_G = \sum_{G} \mathbf{Q}_G \tag{18}$$

where the deformation matrix $\mathbf{B}_G$ is computed at the corresponding Gauss point. The summation is performed for each Gauss point and affects all nodes within its domain of influence. Compared to FEM, the amount of calculations for performing this task is significantly higher since the domains of influence of Gauss points are much larger than the corresponding domains in FEM as is schematically shown in Fig. 1 for a domain discretized with EFG and FEM having equal number of nodes and Gauss points. Throughout this paper we do not address the issue of accuracy obtained by the two methods with the same number of nodes and Gauss points.

In FEM, each Gauss point is typically involved in element-level computations for the formation of the element stiffness matrix which is then added to the appropriate positions of the global stiffness matrix. Moreover, 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 each element. In EFG methods, however, the contribution of each Gauss point is directly added to the global stiffness matrix while the shape functions are not predefined and span across larger domains with a significantly higher amount of Gauss point-node interactions.

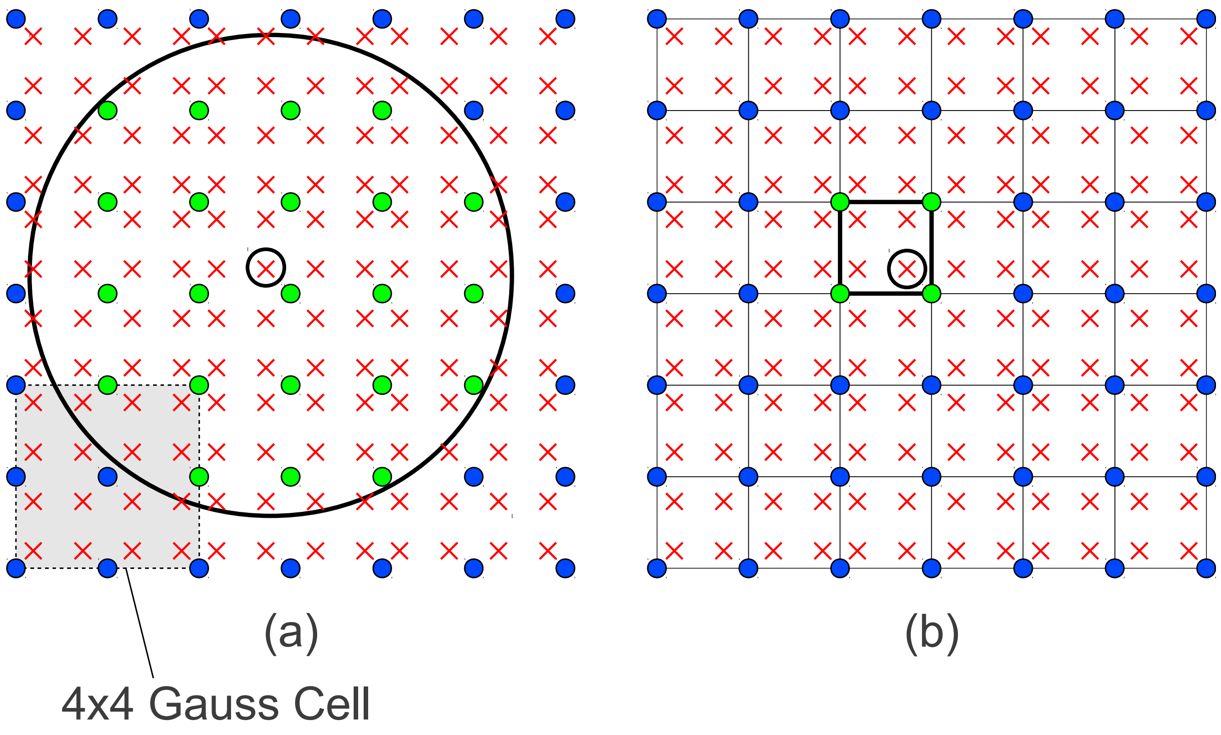

Fig. 1: Domain of influence of Gauss point ⊗ in (a) EFG; (b) FEM, for the same number of nodes and Gauss points

Although, in EFG methods there is no need to construct a mesh, the correlation between nodes and Gauss points needs to be defined. This preliminary step before building the stiffness matrix is implicitly performed with the mesh creation in FEM but must be explicitly done in EFG methods and can be time-consuming if not appropriately handled. For the aforementioned reasons, computing the stiffness matrix in EFG meshless methods is a very computationally demanding task which needs special attention in order to be affordable in real-world applications.

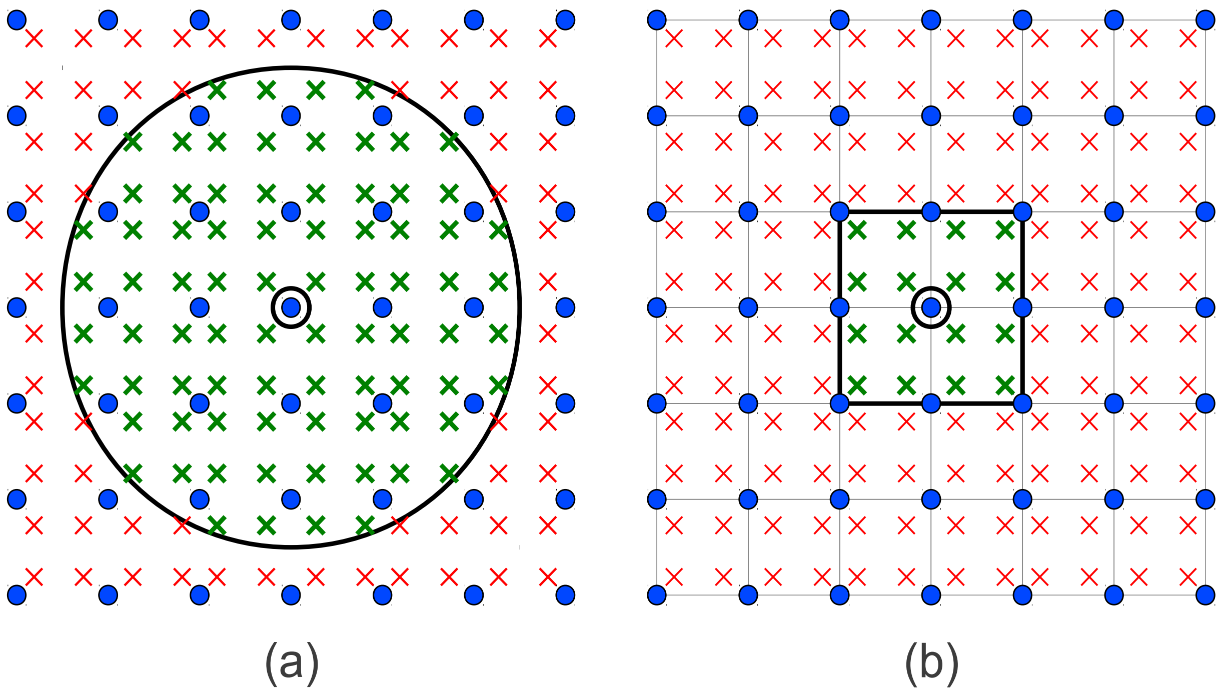

Fig. 2: Domain of influence of node ◉ (a) EFG; (b) FEM, for the same number of nodes and Gauss points

3.1. Node-Gauss point correlation

In the initialization step, the basic entities are created, namely the nodes and the Gauss points together with their domains of influence. The domains of influence define the correlation between nodes and Gauss points. With the absence of an element mesh, the correlation of Gauss points and nodes must be established explicitly at the initialization phase.

A first approach is to search on the global physical domain for the Gauss points belonging to the domain of influence of each node. This approach performs a large amount of unnecessary calculations since the domains of influence are localized areas. In order to reduce the time spent for identifying the interaction between Gauss points and nodes, the search can be performed on Gauss regions.

A rectangular grid is created and we refer to each of the regions defined as a Gauss region. Each Gauss region contains a group of Gauss points. Given the coordinates of a particular node, it is immediately known in which region it is located. The search per node is conducted over the neighboring Gauss regions only instead of the global domain. Thus, regardless of the size of the problem, the search per node is restricted on a small number of Gauss regions.

The time required to define correlations in three 2D and three 3D elasticity problems with varying number of degrees of freedom (dof) are shown in Table 1. The 2D problems correspond to square domains and the 3D to cubic domains, with rectangular domains of influence (doi) with dimensionless parameter 2.5. These domains maximize the number of correlations and consequently the computational cost for the given number of nodes. In these examples, each Gauss region is equivalent to a single Gauss cell. Thus, in the 2D examples each Gauss cell contains 16 Gauss points (4×4 rule) and in the 3D examples 64 Gauss points ($4\times4\times4$ rule). The examples are run on a Core i7-980X which has 6 physical cores (12 logical cores) at 3.33GHz and 12MB cache. Each node can define its correlation independently of other nodes, which is amenable to parallel computations.

Example	Nodes	Gauss points	Search Time (seconds)		
			Global	Regioned	Regioned
			Serial	Serial	Parallel
2D-1	25.921	102.400	23	1,3	0,5
2D-2	75.625	300.304	300	3,4	1,0
2D-3	126.025	501.264	836	5,4	1,4
3D-1	9.221	64.000	7	3,7	0,9
3D-2	19.683	140.608	45	7,8	1,7
3D-3	35.937	262.144	157	15,7	3,3

Table 1: Computing time required for all node-Gauss point correlations

With the implementation of Gauss regions, the initialization phase of EFG methods in complex domains takes less time than FEM, since the generation of a finite element mesh can sometimes be laborious and time consuming [13]. At the end of the initialization step each node has a list of influencing Gauss points and each Gauss point has a list of influenced nodes.

3.2. Performance of the Gauss point-wise approach

The performance of the Gauss point-wise approach in the CPU is shown in Table 2. The proposed Gauss point-wise (GP) approach is compared with the "conventional" one which is a first approach that does not include several optimizations.

Example	dof	Gauss points	Time (seconds)		Ratio
			Conventional GP	Proposed GP	
2D-1	51.842	102.400	107	12	9
2D-2	152.250	300.304	313	34	9
2D-3	252.050	501.264	502	53	9
3D-1	27.783	64.000	2.374	241	10
3D-2	59.049	140.608	6.328	616	10
3D-3	107.811	262.144	13.302	1.165	11

Table 2: Computing time for the formulation of the stiffness matrix in the CPU implementations of the Gauss-point wise approach

4. NODE PAIR-WISE FORMULATION OF THE STIFFNESS MATRIX

An alternative way to perform the computation of the global stiffness matrix is the proposed node pair-wise approach. The computation of the global stiffness coefficient $\mathbf{K}_{ij}$ is performed for all interacting $i-j$ nodes and is formed from contribution by the shared Gauss points of their domains of influence. Fig. 3 depicts two interacting nodes as a result of having common Gauss points in the intersection of their domains of influence and one node that is not interacting with the other two.

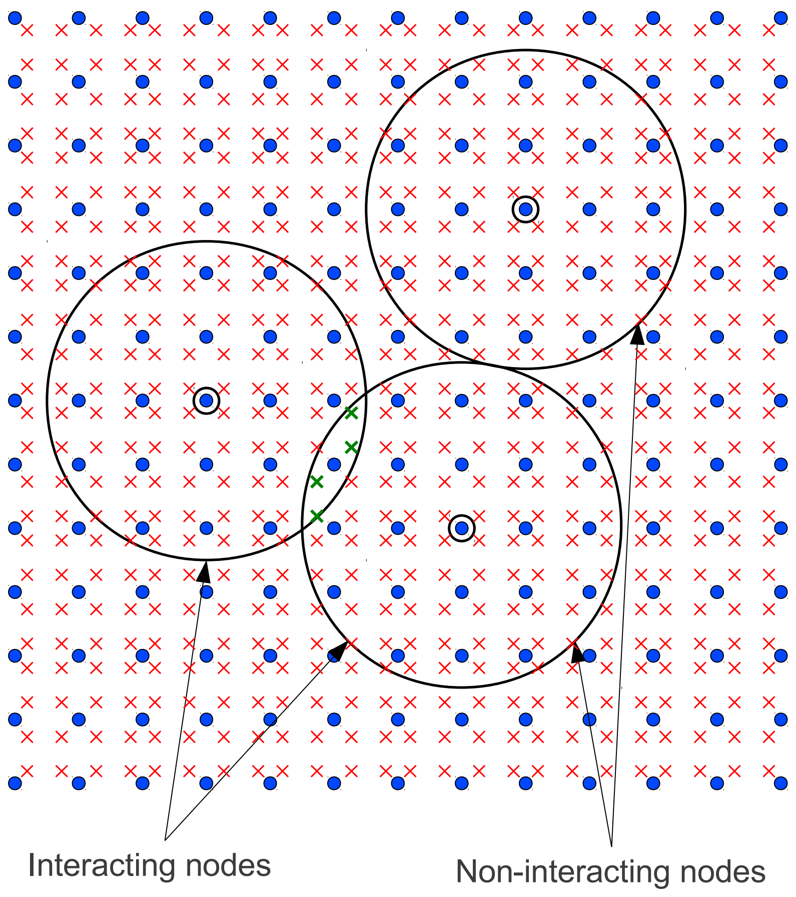

Fig. 3 Intersection of domains of influence

4.1. Computation of global stiffness coefficients for each interacting node pair

The computation of the stiffness elements for each interacting node pair is split in two phases. In the first phase, the shape function derivatives for each influenced node of every Gauss point are calculated as described in the Gauss point-wise method. Then, instead of continuing with the calculation of the stiffness matrix coefficients corresponding to a particular Gauss point, the shape function derivatives are stored for the calculation of $\mathbf{Q}_{ij}$ matrices in the next phase. The required storage of all shape function derivatives is small so storing them temporarily is not an issue.

In the second phase, the stiffness matrix coefficients of each interacting node pair is computed. For each interacting node pair ij, the matrix $\mathbf{Q}_{ij}$ is calculated over all shared Gauss points and summed to form the final values of the corresponding coefficients of the global matrix:

$$\mathbf{K}_{ij} = \sum_G \mathbf{Q}_{ij} = \sum_G \mathbf{B}_i^T \mathbf{E}\, \mathbf{B}_j \tag{19}$$

Both phases are amenable to parallelization, the first with respect to Gauss points and the second with respect to interacting node pairs, and involve no race conditions or the need for synchronization.

4.2. Parallelization features of the interacting node pairs approach

The interacting node pairs approach has certain advantages compared to the Gauss point-wise approach. The most important one is related to its amenability to parallelism, in contrast to the Gauss point-wise approach. Since in EFG methods each Gauss point affects a large number of nodes, each $\mathbf{K}_{ij}$ submatrix is formed by a large number of stiffness contributions. Parallelizing the Gauss point-wise approach involves scatter parallelism, which is schematically shown in Fig. 4 for two Gauss points C and D. Each part of the sum can be calculated in parallel but there are conflicting updates to the same element 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 [14].

In the interacting node pairs approach, instead of constantly updating the matrix, the final values for the submatrix of each interacting node pair are calculated and then appended to 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 nodes should be summed together. Thus, the interacting node pairs approach utilizes gather parallelism as shown schematically in Fig. 5. In a parallel implementation, each working unit, or thread, prepares a submatrix $\mathbf{K}_{ij}$ related to a specific interacting node 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. An important benefit of this approach is the indexing cost of the stiffness matrix elements. In the Gauss point-wise method each stiffness matrix element is updated a large number of times while in the proposed interacting node pair approach the final value is calculated and written only once.

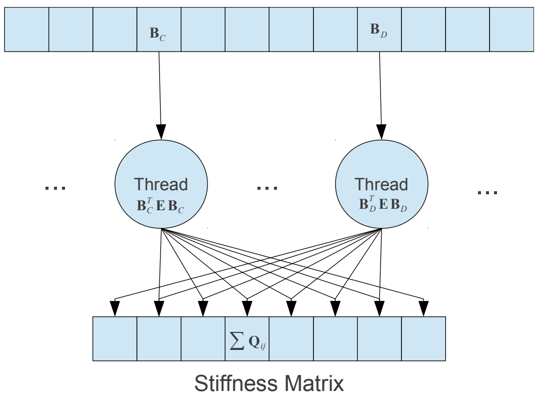
$\mathbf{B}_C$
$\mathbf{B}_D$
...
Thread $\mathbf{B}_C^T\mathbf{E}\mathbf{B}_C$
...
Thread $\mathbf{B}_D^T\mathbf{E}\mathbf{B}_D$
...
$\sum\mathbf{Q}_{ij}$
Stiffness Matrix

Fig. 4: Scatter parallelism required for the Gauss point-wise approach

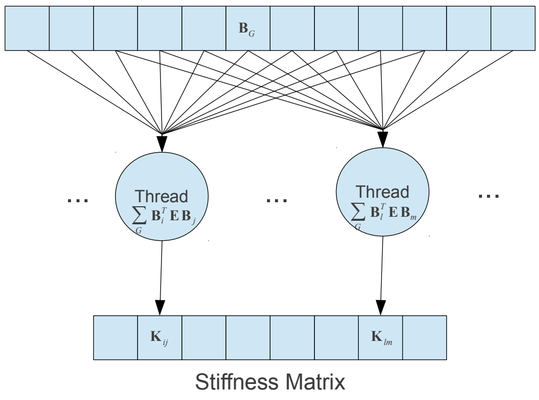
$\mathbf{B}_G$
...
Thread $\sum_G\mathbf{B}_i^T\mathbf{E}\mathbf{B}_j$
...
Thread $\sum_G\mathbf{B}_l^T\mathbf{E}\mathbf{B}_m$
...
$\mathbf{K}_{ij}$
$\mathbf{K}_{lm}$
Stiffness Matrix

Fig. 5: Gather parallelism implemented in the interacting node pairs approach

5. NUMERICAL RESULTS IN 2D AND 3D ELASTICITY PROBLEMS

The two procedures elaborated in this work for the computation of the stiffness matrix in large-scale EFG meshless simulations are tested for the same 2D and 3D elasticity problems already used for testing throughout this paper. The geometric domains of these problems maximize the number of correlations and consequently the computational cost for the given number of nodes. 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: is a GeForce GTX680 with 1536 CUDA cores and 2GB GDDR5 memory.

The performance of the Gauss point-wise (GP) and node pair-wise (NP) approaches in the CPU are given in Table 3. The proposed Gauss point-wise approach is compared with the

Example	dof	Gauss points	CPU Time (seconds)		
			Conventional GP	Proposed GP	Proposed NP
2D-1	51.842	102.400	107	12	11
2D-2	152.250	300.304	313	34	28
2D-3	252.050	501.264	502	53	47
3D-1	27.783	64.000	2.374	241	134
3D-2	59.049	140.608	6.328	616	328
3D-3	107.811	262.144	13.302	1.165	645

Table 3: Computing time for the formulation of the stiffness matrix in the serial CPU implementations of the Gauss point-wise (GP) and node pair-wise (NP) approaches

"conventional" one without the previously described improvements.

The performance of the GPU implementation of the node pair-wise method is shown in Table 4. Speedup ratios of the GPU implementation compared to the CPU implementations is given in Table 5.

Example	dof	Gauss points	NP GPU Time (seconds)		
			Kernel 1	Kernel 2	Total
2D-1	51,842	102,400	0.05	0.19	0,2
2D-2	152,250	300,304	0.13	0.56	0,7
2D-3	252,050	501,264	0.21	0.89	1,1
3D-1	27,783	64,000	0.17	2.41	2,6
3D-2	59,049	140,608	0.32	6.17	6,5
3D-3	107,811	262,144	0.62	12.31	12,9

Table 4: Computing time for the formulation of the stiffness matrix in the GPU implementation of the interacting node-pair approach

Example	Speedup ratios of GPU implementation		
	Conventional GP	Proposed GP	Proposed NP
2D-1	450	50	46
2D-2	457	50	41
2D-3	456	48	43
3D-1	921	93	52
3D-2	975	95	50
3D-3	1.028	90	50

Table 5: Relative speedup ratios of GPU implementation compared to the CPU implementations

6. CONCLUDING REMARKS

The proposed improvements on the initialization phase through the utilization of Gauss regions significantly reduces the time required to create the necessary correlations between the entities of the meshless methods. With Gauss regions, the process scales very well, in contrast to globally searching, and the initialization takes only a small percentage of the problem formulation time.

The improvements in the Gauss point-wise approach for assembling the stiffness matrix offer an order of magnitude speedup compared to the conventional approach. This is attributed to the reduced number of calculations in all parts of the process and the usage of an efficient sparse matrix format and an implementation specifically tailored for the formulation phase of the stiffness matrix. Indexing is a major factor affecting the computational cost. Therefore, the skyline format is faster due to its lower indexing cost, however the significantly higher memory requirement makes it problematic for larger problems where a sparse format is preferable or mandatory.

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

A GPU implementation is applied to the node pair-wise approach offering great speedups compared to CPU implementations. The node pairs keep the GPU constantly busy with calculations resulting in high hardware utilization which is evidenced by the high speedup ratios of approximately two orders of magnitude in the test examples presented. The node 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 the latter 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 meshless EFG methods can be reduced by orders of magnitude. The presented node pair-approach enables the efficient utilization of any available hardware and in conjunction with fast initialization and its inherently parallelization features can accomplish high speedup ratios, which convincingly addresses the main shortcoming of meshless methods making them computationally competitive in solving large-scale engineering problems.

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).

8. REFERENCES

[1] S. Li and W. K. Liu, “Meshfree and particle methods and their applications,” *Applied Mechanics Reviews*, vol. 55, no. 1, pp. 1–34, 2002.

[2] V. P. Nguyen, T. Rabczuk, S. Bordas, and M. Duflot, “Meshless methods: A review and computer implementation aspects,” *Mathematics and Computers in Simulation*, vol. 79, no. 3, pp. 763–813, 2008.

[3] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl, “Meshless methods: An overview and recent developments,” *Computer Methods in Applied Mechanics and Engineering*, vol. 139, no. 1–4, pp. 3–47, 1996.

[4] K. T. Danielson, S. Hao, W. K. Liu, R. A. Uras, and S. Li, “Parallel computation of meshless methods for explicit dynamic analysis,” *International Journal for Numerical Methods in Engineering*, vol. 47, no. 7, pp. 1323–1341, 2000.

[5] K. T. Danielson, R. A. Uras, M. D. Adley, and S. Li, “Large-scale application of some modern CSM methodologies by parallel computation,” *Advances in engineering software*, vol. 31, no. 8, pp. 501–509, 2000.
[6] G. R. Liu, K. Y. Dai, and T. T. Nguyen, “A smoothed finite element method for mechanics problems,” *Computational Mechanics*, vol. 39, no. 6, pp. 859–877, 2007.
[7] J. G. Wang and G. R. Liu, “A point interpolation meshless method based on radial basis functions,” *International Journal for Numerical Methods in Engineering*, vol. 54, no. 11, pp. 1623–1648, 2002.
[8] Y. T. Gu and G. R. Liu, “A coupled element free Galerkin/boundary element method for stress analysis of tow-dimensional solids,” *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 34, pp. 4405–4419, 2001.
[9] W.-R. Yuan, P. Chen, and K.-X. Liu, “High performance sparse solver for unsymmetrical linear equations with out-of-core strategies and its application on meshless methods,” *Applied Mathematics and Mechanics (English Edition)*, vol. 27, no. 10, pp. 1339–1348, 2006.
[10] S. C. Wu, H. O. Zhang, C. Zheng, and J. H. Zhang, “A high performance large sparse symmetric solver for the meshfree Galerkin method,” *International Journal of Computational Methods*, vol. 5, no. 4, pp. 533–550, 2008.
[11] E. Divo and A. Kassab, “Iterative domain decomposition meshless method modeling of incompressible viscous flows and conjugate heat transfer,” *Engineering Analysis with Boundary Elements*, vol. 30, no. 6, pp. 465–478, 2006.
[12] P. Metsis and M. Papadrakakis, “Overlapping and non-overlapping domain decomposition methods for large-scale meshless EFG simulations,” *Computer Methods in Applied Mechanics and Engineering*, vol. 229–232, pp. 128–141, 2012.
[13] R. Trobec, M. Šterk, and B. Robič, “Computational complexity and parallelization of the meshless local Petrov-Galerkin method,” *Computers and Structures*, vol. 87, no. 1–2, pp. 81–90, 2009.
[14] W. W. Hwu and D. B. Kirk, “Parallelism Scalability,” in *Programming and tUning Massively Parallel Systems (PUMPS)*, Barcelona, 2011.