Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

GPU accelerated computation of the isogeometric analysis stiffness matrix

A. Karatarakis [*], P. Karakitsios, M. Papadrakakis

Institute of Structural Analysis and Antiseismic Research, National Technical University of Athens, Zografou Campus, Athens 15780, Greece

A R T I C L E I N F O

Article history:
Received 27 March 2013
Received in revised form 25 October 2013
Accepted 2 November 2013
Available online 19 November 2013

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

A B S T R A C T

Due to high regularity across mesh elements, isogeometric analysis achieves higher accuracy per degree of freedom and improved spectrum properties, among others, compared with finite element analysis. However, this inherent feature of isogeometric analysis increases the density of the 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 for real-world applications. 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 graphics processing units to drastically accelerate computations.

© 2013 Elsevier B.V. All rights reserved.

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 shape 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 increases the cost of IGA in real-world applications. Due to its higher inter-element continuity, IGA produces quite more elements than FEA for the same number of degrees of freedom. This leads to an increase of the number of Gauss points and consequently of the computational cost for assembling the characteristic matrices. This drawback dramatically increases the computational cost in the multivariate domains, especially in 3D analysis.

It has been 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 the fewer the required number of Gauss points per element. However, recently proposed integration rules, although optimal or nearly optimal in terms of the number of function evaluations, are either cumbersome to implement [2] or need special consideration to be given to the boundary elements [3]. In an effort to address the increased effort in the computation of IGA characteristic matrices, collocation methods have been introduced, requiring a minimum number of quadrature points [5,6].

[*] Corresponding author.
E-mail addresses: alex@karatarakis.com (A. Karatarakis), pkarak@hotmail.com (P. Karakitsios), mpapadra@central.ntua.gr (M. Papadrakakis).

0045-7825/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cma.2013.11.008

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 [7–17]. Linear algebra applications have also been a topic of scientific interest for GPU implementations [18–21]. GPU accelerated assembly in finite elements methods is reported in [22–25]. A hybrid CPU–GPU implementation of domain decomposition methods is presented in [26] where speedups of the order of 40$\times$ 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 interaction-wise procedure recently proposed for the computation of the stiffness matrix in element-free Galerkin formulations [27]. This approach is amenable to parallel computations since it does not have race conditions or the need for 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 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 the exact geometry is always represented – even in the case of very coarse meshes – and thus there is no approximation in that regard. 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 take place in this space, 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 the projections of knot lines under the B-SPLine mapping. Fig. 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.

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:

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

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

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 NURBS, shape functions are produced from the following formula in 1D case:

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

in the 2D case:

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

Download English Version:

https://daneshyari.com/en/article/497913

Download Persian Version:

https://daneshyari.com/article/497913

Daneshyari.com