Computers and Mathematics with Applications 67 (2014) 1864–1883

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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

Computational cost estimates for parallel shared memory isogeometric multi-frontal solvers

M. Woźniak [a], K. Kuźnik [a], M. Paszyński [a,*], V.M. Calo [b], D. Pardo [c]

[a] AGH University of Science and Technology, Krakow, Poland
[b] King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
[c] University of the Basque Country (UPV/EHU), BCAM, and Ikerbasque, Bilbao, Spain

ARTICLE INFO

Article history:
Received 9 April 2013
Received in revised form 18 March 2014
Accepted 31 March 2014
Available online 18 April 2014

Keywords:
Isogeometric finite element method
Multi-frontal direct solver
Computational cost
NVIDIA CUDA GPU

ABSTRACT

In this paper we present computational cost estimates for parallel shared memory isogeometric multi-frontal solvers. The estimates show that the ideal isogeometric shared memory parallel direct solver scales as $O(p^2 \log(N/p))$ for one dimensional problems, $O(Np^2)$ for two dimensional problems, and $O(N^{4/3}p^2)$ for three dimensional problems, where N is the number of degrees of freedom, and p is the polynomial order of approximation. The computational costs of the shared memory parallel isogeometric direct solver are compared with those corresponding to the sequential isogeometric direct solver, being the latest equal to $O(Np^2)$ for the one dimensional case, $O(N^{1.5}p^3)$ for the two dimensional case, and $O(N^2p^3)$ for the three dimensional case. The shared memory version significantly reduces both the scalability in terms of N and p. Theoretical estimates are compared with numerical experiments performed with linear, quadratic, cubic, quartic, and quintic B-splines, in one and two spatial dimensions.

1. Introduction

Classical higher order finite element methods (FEM) [1,2] maintain only C^0-continuity at the element interfaces, while isogeometric analysis (IGA) utilizes B-splines as basis functions, and thus, it delivers C^k global continuity [3]. The higher continuity obtained across elements allows IGA to attain optimal convergence rates for any polynomial order, while using fewer degrees of freedom [4,5]. Nevertheless, this reduced count in the number of degrees of freedom may not immediately correlate with a computational cost reduction, since solution time per degree of freedom augments as the continuity is increased [6,7]. In spite of the increased cost of higher-continuous spaces, they have proven very popular and useful. For example, higher-continuous spaces have allowed the solution of higher-order partial differential equations with elegance [8–13] as well as several non-linear problems of engineering interest [14–21]. Thus, efficient multi-frontal solvers for higher-continuous spaces are important.

The multi-frontal solver is one of the state-of-the art algorithm for solving linear systems of equations [22,23]. It is a generalization of the frontal solver algorithm proposed in [24,25]. The multi-frontal algorithm constructs an assembly tree based on the analysis of the connectivity data or the geometry of the computational mesh. Finite elements are joined into pairs and fully assembled unknowns are eliminated within frontal matrices associated to multiple branches of the tree. The

* Corresponding author.
E-mail addresses: macwozni@agh.edu.pl (M. Woźniak), kmkuznik@gmail.com (K. Kuźnik), paszynsk@agh.edu.pl (M. Paszyński), victor.calo@kaust.edu.sa (V.M. Calo), dzubiaur@gmail.com (D. Pardo).

http://dx.doi.org/10.1016/j.camwa.2014.03.017
0898-1221/

process is repeated until the root of the assembly tree is reached. Finally, the common interface problem is solved and partial backward substitutions are recursively called on the assembly tree.

There exist parallel versions of the multi-frontal direct solver algorithm targeting distributed-memory, shared-memory, or hybrid architectures. The partition of data for distributed memory architecture may concern the redistribution of the computational mesh into sub-domains with overlapping or non-overlapping elements [26,27], the redistribution of the global matrix [28], or the redistribution of the elimination tree [29–32].

There also exist some versions of the multi-frontal solver algorithm for shared memory machines. These algorithms store the entire matrix in the shared memory and perform matrix operations concurrently [33–35]. The matrix is partitioned into blocks with BLAS operations performed concurrently over each block.

One limitation of the IGA is the fact that direct solvers work slower, due to *inconvenient* sparsity pattern of the resulting matrix [7]. In particular, the sequential IGA direct solvers delivers $O((N/p)p^2)$ computational cost for one dimensional problems (1D), $O(N^{1.5}p^3)$ for two dimensional problems (2D), and $O(N^2p^3)$ for three dimensional problems (3D) [7]. In those estimates, we assumed uniform p-order B-spline basis functions over the entire mesh delivering C^{p-1} global regularity of the solution. The direct solver algorithm for IGA delivering C^{p-1} global regularity with p-order B-splines is p^3 times slower than the direct solver with p-order polynomial with C^0 global regularity, for 2D and 3D problems, and p^2 times slower for 1D problems. Thus, for $p = 3$, the C^2 global continuity problem is 27 times more expensive to be solved than the C^0 counterpart in 2D and 3D, which implies that we may need to wait days instead of hours to obtain the solution.

The main contribution of this paper is to show theoretically and experimentally that the IGA sequential solver limitations can be overcome to some extent by utilizing shared memory GPU implementations. In this paper, we present the derivation of the computational cost for an ideal shared memory parallel direct solver, where we assume zero communication costs, and uniform grids in terms of h (element size) and p (polynomial order of approximation). The computational cost estimations imply that the ideal IGA shared memory parallel direct solver delivers $O(p^2 \log(N/p))$ for 1D problems, $O(Np^2)$ for 2D problems and $O(N^{4/3}p^2)$ for 3D problems.

With these results, not only the N dependence is significantly improved by using the shared memory version, but also the p-dependence, which makes IGA more competitive with respect to traditional C^0-FEM when using shared memory machines. In particular, for $p = 3$, the C^2 global continuity problem becomes *only* 9 times more expensive to be solved than the C^0 counterpart in 2D and 3D, as opposed to 27 in the sequential version.

The first part of the paper presents estimates of computational costs and scalability for any shared memory implementation of multi-frontal IGA solvers. The second part of the paper describes an implementation of the multi-frontal algorithm for graphics processors. The code is presented, its performance tested for several GPUs and the results are used for verifying the estimates derived in the first part. Among other things, this combination between theoretical and numerical results enables to show that the leading term (scalability) of our solver is not destroyed by implementation/architectural issues such as memory access. Notice that we are not assuming that the cost of memory access is zero, but rather that it is of lower (or equal) order that the number of floating point operations, which ultimately determines the scalability of the solver, as shown by the numerical results.

In some recent papers [36,37], the influence on execution time of the implementation factors, such as processor occupancy, thread synchronization, and organization of memory accesses is analyzed. However, in such a complex algorithm as the one presented here, a detailed analysis of memory access, synchronization, organization of memory accesses, etc. is extremely difficult to perform, and it is out of the scope of this paper. Rather, numerical results show that the theoretical scalability is indeed achievable in practice, thus, showing that other factors (including memory access) will not destroy the predicted scalability of the solver.

The theoretical computational cost estimates are compared with our parallel shared memory implementation. We target our solver NVIDIA's GPU architecture, which is a complex shared memory architecture. We tested our implementations on a GeForce GTX 560 Ti device with 8 multiprocessors, each one equipped with 48 cores. As well as on NVIDIA Tesla C2070 device, which has 14 multiprocessors with 32 CUDA cores per multiprocessor. We also performed tests for the 2D solver on a GeForce GTX 780 graphic card equipped with 3 gigabytes of memory and 2304 cores.

For the description of a B-spline based multi-frontal solver algorithm for 1D problems, we refer to [38]. The 1D IGA solver is similar to those used in finite difference methods [39]. For a detailed description of the B-spline based multi-frontal solver for 2D problems, we refer to [40].

The structure of the paper is the following. We start with the definition of our model problem in Section 2. Section 3 describes the basics of the isogeometric analysis using B-splines. Section 4 introduces the multi-frontal solvers algorithm for IGA. Section 5 presents the computational cost and memory usage estimates for the 1D, 2D, and 3D multi-frontal shared-memory parallel, direct solver algorithm. In Section 6, we describe the technical details related to the implementation of the 1D and 2D multi-frontal solver. Section 7 presents the numerical results for 1D and 2D models as well as a short discussion on the limitations of the 3D implementation. Finally, Section 8 depicts the main conclusions of this work and possible lines of research for the future.

2. Model problem

In this section, we present our model problem. We focus on the 2D conductive media equation

$$\nabla \cdot \sigma \nabla u = \nabla \cdot \mathbf{J}^{imp}, \tag{1}$$