Procedia Computer Science

Volume 80, 2016, Pages 865–874

ICCS 2016. The International Conference on Computational Science

Hybrid direct and iterative solver with library of multi-criteria optimal orderings for *h* adaptive finite element method computations

Hassan AbouEisha[1], Konrad Jopek[3], Bartłomiej Medygrał[3], Mikhail Moshkov[2], Szymon Nosek[3], Anna Paszyńska[4], Maciej Paszyński[3], Keshav Pingali[5*]

[1]Computer Science, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
[2]Applied Mathematics and Computational Science, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
[3]Department of Computer Science, AGH University of Science and Technology, Kraków, Poland
[4]Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, Kraków, Poland
[5]Institute for Computational and Engineering Sciences, The University of Texas at Austin, USA

Abstract

In this paper we present a multi-criteria optimization of element partition trees and resulting orderings for multi-frontal solver algorithms executed for two dimensional h adaptive finite element method. In particular, the problem of optimal ordering of elimination of rows in the sparse matrices resulting from adaptive finite element method computations is reduced to the problem of finding of optimal element partition trees. Given a two dimensional *h* refined mesh, we find all optimal element partition trees by using the dynamic programming approach. An element partition tree defines a prescribed order of elimination of degrees of freedom over the mesh. We utilize three different metrics to estimate the quality of the element partition tree. As the first criterion we consider the number of floating point operations(FLOPs) performed by the multi-frontal solver.

As the second criterion we consider the number of memory transfers (MEMOPS) performed by the multi-frontal solver algorithm. As the third criterion we consider memory usage (NONZEROS) of the multi-frontal direct solver. We show the optimization results for FLOPs vs MEMOPS as well as for the execution time estimated as FLOPs+100*MEMOPS vs NONZEROS. We obtain Pareto fronts with multiple optimal trees, for each mesh, and for each refinement level. We generate a library of optimal elimination trees for small grids with local singularities. We also propose an algorithm that for a given large mesh with identified local sub-grids, each one with local singularity. We compute Schur complements over the sub-grids using the optimal trees from the library, and we submit the sequence of Schur complements into the iterative solver ILUPCG.

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016

doi:10.1016/j.procs.2016.05.312

Keywords: finite element method; h adaptivity; multi-frontal direct solver; element partition tree; ordering algorithms; ILUPCG

1 Introduction

Multi-frontal solver [1,2] is the state-of-the-art direct solver algorithm for solving sparse systems of linear equations resulting from two dimensional h adaptive finite element method computations. The performance of the multi-frontal solver algorithm depends on the so-called ordering, that can be obtained from element partition tree (see chapter 8 in [3]) prescribing recursive divisions of the computational mesh. The nested-dissections [4] (equally weighted partitions) are optimal only for regular grids. When we switch to adaptive grids, the nested dissections algorithm is no longer optimal. The problem of finding optimal element partition tree defining optimal ordering, is NP-complete, as the problem of finding the minimum fill-in [5]. In this paper we focus on an automatic optimization of the element partition trees using dynamic programming. We augment the results of our previous automatic optimization [6,7,8] by considering three different cost functions.

First, we focus on estimations of the number of floating point operations (FLOPs) performed by the multi-frontal solver algorithm. Second, we focus on estimations of the number of memory transfers (MEMOPS) performed during the factorizations. Third, we focus on estimation on the memory usage (NONZEROS) of the solver. We generate Pareto fronts with optimal element partition trees by using the dynamic programming approach with minimization of FLOPs vs MEMOPS, as well as the execution time of the solver estimated as FLOPs+100*MEMOPS vs the NONZEROS.

Our dynamic programming approach allows us to build a library of optimal element partition trees, for a class of small grids with local singularities. Having a large grid with several local singularities, we run a greedy algorithm that localizes sub grids with local refinements. Finally, for each sub grid we compute Schur complement (we eliminate interior of the sub-grid with respect to its boundary). We do that using the ordering following the element partition tree, as described in chapter 8 of [3], and the GALOIS multi-frontal solver [7], that allows to pass the element partition tree as an input. Having the Schur complement computed, we collect them and call iterative solver ILUPCG from SLATEC library [9]. We show that our approach allows to significantly reduce the number of iterations of the iterative solver, if the local singularities are well separated.

2 Dynamic programming optimization

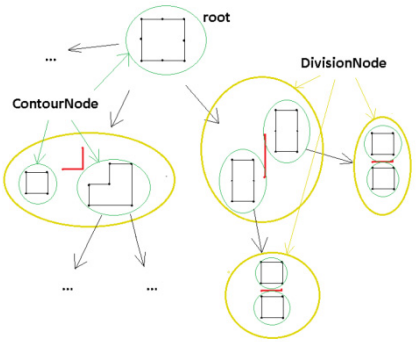

Figure 1. Contour Nodes and Division Nodes of the DAG

Following the ideas already presented in [8], we construct a dynamic programming approach that considers many recursive partitions of the computational mesh. It first constructs a Directed Acyclic