46th Annual Iranian Mathematics Conference
25-28 August 2015
Yazd University

Talk Parallelization of the adaptive wavelet galerkin method for elliptic BVPs pp.: 1–4

Parallelization of the adaptive wavelet galerkin method for elliptic BVPs

Nabi Chegini
Tafresh University

Abstract

In this work, an adaptive wavelet galerkin method (`AWGM`) with optimal computational complexity is parallelized. The method is applied to the solution of the second order elliptic BVPs. With tensor product wavelet basis, the rate of the `AWGM` is dimension independent. The numerical results indicate the method converge with optimal rate. Our results demonstrate that the `AWGM` can be implemented in a multiprocessor environment and is scalable.

Keywords: Adaptive method, Tensor product wavelets, Parallel computation
Mathematics Subject Classification [2010]: 35K15, 65F50, 65Y05

1 Introduction

This paper deals with the implementation of the `AWGM` in shared-memory parallel programming. Recently multiprocessing platforms are available with multi-core processors sharing memory. An efficient way for performing the applications in high performance computing fields is to parallelize them in multiprocessing schemes. In order to achieve the best speed up as possible, *Synchronization* is a natural and essential part of parallel programs. We strongly notice that this main task of parallelization cannot be avoided in the adaptive methods. Shared-memory computing are rendered parallel with threading model extensions such as *OpenMP* and Pthreads. In this context, A *thread* is a sequence of such instructions within a program that can be executed independently of other code. In fact, *OpenMP* and Pthreads programming are two well known and dominant shared-memory programming models.

OpenMP is a portable interface for implementing fork-join parallelism on shared memory multiprocessor machines. It is a library which implemented with *"omp.h"*. *OpenMP* provides suitable level of abstraction to a programmer. It extends and defines a set of directives and library routines for Fortran and C/C++ [1] . Actually it consists of the set of directives, clauses and functions that enables creating, managing, communicating and synchronizing parallel threads.
One of the advantages of the programming in *OpenMP* is that the resulting parallel code is close to its sequential version. It explicitly decalres parallel regions but much of the synchronizations are managed implicitly. The execution performance of the program in *OpenMP* is highly dependent on the quality of the *OpenMP* implementation. An efficient way for designing the data structure of adaptive methods is hash table. In multi core