A GPU parallelized spectral method for elliptic equations in rectangular domains

Feng Chen a,∗, Jie Shen b

a Division of Applied Mathematics, Brown University, Providence, RI 02912, United States
b Department of Mathematics, Purdue University, West Lafayette, IN 47907, United States

1. Introduction

General-purpose computing on graphics processing units (GPGPU) has drawn much attention recently from the scientific computing community. The new list of top 500 supercomputers shows that more than 10% of them are now powered by NVIDIA Tesla GPUs (cf. [4]). Thanks to the high core density and the wide vector width SIMD architecture, using GPGPU can yield performance that is very hard for a conventional CPU to achieve, especially for high and regular throughput workloads (cf. [3]). Therefore, researchers in computational sciences are more and more interested in exploring parallel efficient strategies on GPUs.

There exist many successful GPU implementations of numerical methods for partial differential equations, to name a few, fast multipole methods [10], modal discontinuous Galerkin methods [12], finite difference methods [6], finite element methods [24,14,13], Fourier spectral methods [5,5], etc. However, to the best of the authors’ knowledge, not much effort, if not at all, are devoted to implementing, on GPUs, non-Fourier-based spectral methods for problems with non-periodic boundary conditions.

The aim of this paper is to exploit how to efficiently implement spectral methods with GPGPU. In particular, we shall design and implement an efficient matrix diagonalization method with spectral discretizations on the GPU, for solving elliptic and parabolic type equations with non-periodic boundary conditions on a d-dimensional rectangular domain (d = 2, 3).

It is well-known that, for separable elliptic equations, one can use the so-called matrix diagonalization technique (cf. [16,11,18]) to solve the linear systems from a spectral discretization in O(N d) operations, where N is the number of points in each dimension. For time-dependent problems, these solvers are called repeatedly. Fortunately, the core part of this