PARALANCZOS GPU LIBRARY (VERSION 0.1)

BRIAN TUOMANEN

Abstract. This is a preliminary document describing the usage, implementation, and underlying theory of the Lanczos GPU Library. The author would appreciate any feedback on this document or the library; moreover, he is interested in possible collaboration and discussion with researchers in Compressive Sensing. He can be reached at `btuomanen@outlook.com` .

1. Introduction

The purpose of this software library is to perform fast massively parallel singular value decompositions (SVDs) of large quantities of comparably small non-singular submatrices within a larger positive-symmetric matrix in CUDA; notably, this library has been developed to assist in the study of Reduced Isometry Property (RIP) sensing matrices in the field of Compressive Sensing. To this end, several Matlab front-end examples are included, as well as standard command-line C examples.

As per the name, the main CUDA kernel function performs a Lanczos tridiagonalization on each submatrix, and then a QL decomposition to extract the eigenvalues of the resulting similar submatrix. The user indicates which submatrices of the larger matrix are to be analyzed by a parameter given to the kernel function, which is then run directly on the GPU.

Due to the extensive usage of CUDA "shuffle" commands to speed up communication between GPU threads and avoid race conditions, this library requires at least a Kepler-level NVidia GPU to run (2012-); note that ParaLanczos was written and tested on Maxwell-level (2014) devices. Compilation parameters and Matlab parameters might have to be tested and tweaked for some systems.

Note that the Matlab examples are reliant on the Mathworks Parallel Computing Toolbox to be installed; only Matlab 2015A has been tested.

Also note this library has only been tested on Linux Mint 17.2. Modifications may be duly needed for it to work properly on other operating systems.

2. Installation

Please ensure that you have CUDA 7.5 or later, and the appropriate Linux paths are set up before proceeding.

To compile the library for a default Maxwell level (or later) GPU, type `make`. To compile the library for a Kepler level GPU, modify the `Makefile` for the appropriate architecture.

A basic example showing usage for command-line C programming are given in the file `example1.cu` file; examples showing usage for Matlab are given in `MATLAB` directory.

3. Usage

Note that this is a preliminary document to show basic usage; many of the library components are undocumented. The reader is thus encouraged to look at the library code (particularly `example1.cu`) to infer usage from within C++, and in particular the Matlab wrapper functions specifically how the GPU kernel functions work. Thus, we only show usage in Matlab.

The ParaLanczos kernel works by performing eigendecompositions of a specified set of submatrices of a larger real matrix, stored in an array of doubles in the GPU memory.

Currently, ParaLanczos only supports real, nonsingular submatrices of a given dyadic size (i.e., $2^x \times 2^x$), ranging from size 4×4 to size 128×128. Technical information regarding the use of these kernels follows in this section, while theoretical discussion of the Lanczos algorithm and its implemention are in the following section.

3.1. Matlab Wrappers Usage.

As stated, please make sure that you are using Matlab 2015A and have the Parallel Computing Library installed for proper usage. Note the outputs will all be -1 if there is an error. (i.e., the user tries to extract eigenvalues from a singular matrix or submatrix.)

To see a basic, concrete usage of the following functions, please see the Matlab example file `how2use.m`.

- `gpu_svd.m` : This function extracts the eigenvalues from one nonsingular positive-definite, symmetric real matrix G of dyadic size, with the specified outputs on the left-hand side:
  `[Eigenvalues, Determinant, Error] = gpu_svd(G)` .
- `gpu_subsvd.m` : This function extracts the eigenvalues from one nonsingular dyadic sized submatrix of a given positive-semidefinite matrix G; the indices are given in the variable `subIndex`:
  `[Eigenvalues, Determinant, Error] = gpu_svd(G, subIndex)` .
- `gpu_multi_subsvd.m` : This acts similarly to the prior function, only multiple submatrices can be analyzed; each index is stored in `subIndices`, one after another; and each submatrix must be of the same size. The size of the submatrices is given by minor, which should be a dyadic integer. The eigenvalues for each submatrix are then stored in `Eigenvalues`, each following another and corresponding to the matrices given in `subIndices`:
  `[Eigenvalues, Determinants, Errors] = gpu_multi_subsvd(G, subIndices, minor)` .

Acknowledgments

This work was supported by the National Science Foundation. (NSF ATD 1321779) The author expresses his gratitude to Professor Michela Becchi of the University of Missouri Department of Electrical and Computer Engineering for her invaluable assistance and advice with CUDA and GPU programming, and also thanks Professor Stephen Montgomery-Smith of the Department of Mathematics for his feedback.

Department of Mathematics, University of Missouri, Columbia, MO 65211-4100
E-mail address: btuomanen@outlook.com