CM3202 - One Semester Project (40 Credits)

30th January 2016

Parallelisation of Matrix Exponentials in C/C++/CUDA for Quantum Control

---

-- Initial Plan--

Author: Peter Davison (C1310136)

Supervisor: Dr. Frank C Langbein

Moderator: Dr. Irena Spasic

1. Project Description:

1.1 - Introduction: Quantum technology is beginning to evolve faster and faster and quantum computing is evermore at the forefront of this research. As we improve this technology, the need for faster control systems increases. The matrix exponential function is used a lot quantum control systems and has been improved many times recently by parallelising its computation using CPU cores.

$$e^A = I_n + \frac{1}{1!}A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \cdots + \frac{1}{n!}A^n + \cdots\cdots$$ [1]

Above, is an equation for an approximation of a matrix exponential. This method uses the taylor series for exponentials where A is a matrix. This is one of many ways of solving the exponential of a matrix.[2]

1.2 - The Problem: Most matrix exponentials (with a few exceptions for special cases. eg. in diagonal matrices) are computationally very expensive to calculate. There are many methods of calculating a solution such as the Padé approximation, differential equations and eigendecomposition to name a few, but all of these still a require many operations to be performed; particularly on large matrices.

When you have a project which is performing operations on many large matrices, many times, tiny differences in code efficiency can make giant leaps in the grand scheme of things. With Moore's law soon to run out,[3] we can no longer rely on improvements in chip speed or transistor size and code optimisation is sure to take over in the pursuit of faster technology.

1.3 - My Solution: I believe that using a GPU to perform these tasks could greatly improve the runtime of this function and other matrix functions. A GPU is a specially designed CPU with a large number of cores and a shared memory layout. This makes them perfect for executing parallelised code. As I will be using Nvidia's GPUs for this project, I will be using CUDA

The purpose of this project is to investigate the efficiency of existing algorithms for performing matrix exponentials and to implement an improved algorithm using CUDA and C++. The improved algorithm's complexity should be better when compared to existing implementations.

---

[1] "Matrix Exponential - SOS Math." 31 Jan. 2016 <http://www.sosmath.com/matrix/expo/expo.html>
[2] Moler, C. "Nineteen Dubious Ways to Compute the Exponential of a ..." 2007. <http://www.cs.cornell.edu/cv/researchpdf/19ways+.pdf>
[3] "Moore's Law." 2006. 31 Jan. 2016 <http://www.mooreslaw.org/>

2. Project Aims and Objectives:

2.1 Goal: The aim of the project is to observe an increased efficiency in performing the matrix exponential function. To determine whether I have achieved my goal, I should perform several tests against existing implementations of the function (eg. in MatLab)[4] and also against my own sequential code. I hope to achieve either more accurate results in a similar time or a similar result in less time (or both) when compared to non parallel implementations.

2.2 - I Must:

- Write a sequential implementation of a matrix exponential function in C/C++.
- Write a parallel implementation of a matrix exponential function using C/C++ and CUDA.
    - This should perform all the same functions as the sequential code and should use the sequential class as a base.
- Compare the above implementations with regards to runtime, efficiency and accuracy against existing

2.3 - I Should:

- Write all my code to be as optimal as I can
- Parallelise sub-operations of the matrix exponential (eg. matrix multiplications)

2.4 - I Could:

- Create a complete matrix class with access to other functions (multiplications, additions, determinant, inverse, etc).
    - All of these should stick to the principles outlined for the matrix exponential function. ie. should be as efficient as possible and use the GPU for computation in parallel where possible.
- Parallelise any other operations in the class that are not essential to the matrix exponential.

[4] "Matrix Exponentials Three of the 19 ways to ... - MathWorks." 2014. 31 Jan. 2016 <http://www.mathworks.com/examples/matlab/1187-matrix-exponentials>

3. Work Plan: Below is a list of all the tasks that will need completing during this project and when they should be completed by. Note that everything in the section for week one has been completed before this plan has been submitted, but has been included for completeness.

Week 1:

- Complete initial plan.
- Research matrix exponentials (What they are, methods of calculation).
- Develop ideas on how to speed up performance of these methods.

Week 2

- Write a basic implementation in C/C++ to handle matrices. This will form the base for the rest of the code so everything should be well thought out and planned with expandability in mind.
- Implement some basic matrix functions such as multiplication. These will be used later on when performing more complex functions so should still be fast.

Week 3

- Implement a basic, sequential exponential function in C/C++ (Similar to existing sequential implementations).

Week 4

- Compare the runtime of my sequential function with existing implementations.
    - I expect other implementations to be faster as they may make use of CPU parallelisation. However, I plan to get as close as possible.
- Begin drafting final report with the results of the above tests.

Week 5

- Begin CUDA implementation of my code.
- Experiment with different ways of parallelising and record the results for evaluation.

Week 6

- Have half CUDA part written and have some form of parallel code working.

Week 7

- Complete CUDA implementation of the matrix exponential function.

Week 8

- Complete implementation of all code
- Make sure everything is as lightweight and efficient as possible including all sequential code.
- Make sure class is complete (ie. full of reusable methods)

EASTER BREAK

- Catch up on any work that is behind
- Any additional code that I want to add should be done here

Week 9

- Data gathering and putting together drafted parts of report.

Week 10

- 100% attention on final report.

Week 11

- Complete final report draft.

Week 12

- Complete final report and publish