URSS Project

Luke Raybould

September 18, 2023

1 Introduction

Hey! I'm Luke, a Physics undergraduate student. I'm writing this report to document my summer project between the 3rd and 4th years of my degree. I've found particular interest in computational physics, so aimed to find some experience while considering my options for after my degree. I hope to cover my experiences with starting research in an unfamiliar area, and how I approached learning and staying organised.

I'm taking a summer project through Warwick's Undergraduate Research Support Scheme (URSS), which provides support and funding for summer research projects with academics at the University. My department held a Postgraduate and URSS research evening where I spoke to a few people from the HetSys group. From there I ended up speaking to Albert Bartok-Partay from the Warwick Centre for Predictive Modelling (WCPM) and HetSys who had a high performance computing (HPC) focused project proposal that seemed perfect for me! I had to complete an application through the URSS webpage, which covered the details of the project and what I hope to gain from doing the project. We decided that the project would be split into two parts: a first part in June/July and then a second part in September/October, with August as a break in between.

2 Project Overview

My project is about writing code for efficient computation of spherical harmonics. Spherical harmonics and their derivatives are an essential tool in quantum mechanics, and hence are frequently needed in simulations of physical phenomena, particularly in the atomistic simulations that are the focus of WCPM. Additionally, spherical harmonics show up in signal processing, geometric deep learning, astronomy, and computer graphics.

One of the goals of the project is to have the ability to handle the calculations on a Graphical Processing Unit (GPU). The use of a GPU has major advantages in floating point calculations and is likely to make the evaluation of spherical harmonics significantly faster.

With traditional calculation methods the cartesian coordinates (x, y, z) of a point are converted into spherical coordinates (r, θ, ϕ), and a recurrence relation is used to find the Legendre polynomials in the SH equation. A more effective approach will calculate directly using the cartesian coordinate and by avoiding numerical instabilities in the Legendre polynomial iteration. A similar approach can be used for finding derivatives, which could be an additional goal for the project if successful.

3 Weeks 1-2 : Getting organised

3.1 Week 1

My first steps on the project were to sort my organisation and to decide on the tools (i.e. programming language, any software) being used with my project supervisor. I started by buying a notebook to keep notes about the theory and meeting with my supervisor. This ended up being really helpful! I tend to not work well with online notes so getting one let me keep track of the project progression and

kept all the information in one place.

After this was my first project meeting. I met with my supervisor to speak about starting and the URSS process. We discussed which programming language the code should be written in, some useful existing research on the area, and set up Slack for communicating on the project. We also talked about getting remote access from Warwick's Scientific Computing RTP (SCRPT) to a remote desktop for smaller tests/compiling, and a more powerful HPC desktop for larger tests and benchmarking later.

3.2 Fortran

In the meeting we decided on using Fortran over other options (such as C) as the language for the project. WCPM work primarily in Fortran, so this meant I could get more support and keep it consistent with the rest of their work. Fortran is very well suited to HPC and scientific applications and has easy to use multidimensional arrays built in. However, I had not used Fortran before this, so had to learn it for this project. Fortran is older and less widely used than C (which I also had previous experience using), so high quality learning resources were harder to find.

I spent much of the first week of the project getting familiar with the language and found it to be nowhere near as difficult as expected. The language was intuitive to learn, and I found a textbook in the University library which helped with the basics. Having experience with another programming language (especially one popular in Physics) is a very valuable skill and something I can take away from doing the project.

3.3 Week 2

During our second project meeting, we discussed Git and accessing the package's code from the remote desktop. I had never used Git before, so signed up for a GitHub account for this! We briefly covered the basics in this meeting and got a fork of the QUIP package made and cloned to the remote desktop.

For the rest of week I had a look at the existing code for spherical harmonics in the QUIP package. This ended up being a great way to improve at Fortran and get used to the conventions used in the package. I also looked into using Nano for quickly editing code files straight from the command prompt, another basic tool I wasn't familiar with that taking this project prompted me to learn.

4 Weeks 3-5 : Python Prototype

In our 3rd meeting we covered compiling the QUIP package on the remote desktop. "I was also recommended to use VS Code as my primary IDE for the project. So I installed some addons to make connecting to the remote desktop and coding in Fortran easier.

My first steps were producing a simple prototype. I chose Python for this. My goal was to familiarize myself with the theory and create a basis for future comparisons. Python's ease of use and my experience with it made this a lot easier to do than working in Fortran from the start, and I believe it saved me a lot of time overall.

I started with making a function for the prefactor. This doesn't depend on the coordinates, so in the final project can be run once and not computed again, greatly improving efficiency. For now, I just focused on getting the function to work correctly. Next were the iterative parts. I spent a few days making a version but was unable to get the function to produce the correct values. In an attempt to troubleshoot, I manually worked through the iterations for simpler cases. Surprisingly, the results matched my code, which suggested a deeper issue.

Until now I'd tried to work independently, but after hitting this problem I knew I needed support so organised a meeting. In the meantime, I made some functions for converting between the complex and real forms of spherical harmonics, as my program gave the real form and many other implementations (such as the existing QUIP function and SciPy) used the complex form. When we met we discussed

the issue for an hour, but weren't able to find the cause of the problem. We managed to schedule a 2nd meeting later that afternoon. It was between these two meetings (in fact, just 15 minutes before we'd scheduled our follow up) that I managed to spot the error – a single missing minus sign! My entire function was now giving the correct spherical harmonic values.

5 Weeks 5-7 : Writing the program in Fortran

My next steps were to begin writing the code in Fortran. This should be a lot more straightforward now I have a working prototype, the main challenges will be in where Python and Fortran handle variables and arrays differently. My aim is to calculate as much of the function that is independent of the coordinate before the main loop of the function and store these in an array, which will save significant time when dealing with a large number of coordinates.

I started by working on the inputs and outputs of the function - making sure the function expected the right arguments and gave an output in the correct form. I also created a test file, which calls the function and prints its output for the purpose of testing. I then began to work on the prefactor loop. This is at the start of the function and stores its output in an array.

After a week, I managed to have this function properly working. I decided to spend some time testing and researching to find some possible efficiency improvements. I also did some rewriting to try keep the style of code consistent with the rest of QUIP.

6 Next steps

I had a final meeting with my supervisor before August to discuss what the focus of the 2nd half of the project will be. In terms of the existing code, the next steps include starting to incorporate GPU acceleration in the form of parallelising the main for loop, creating an additional 2 functions to handle initialising and finalising the function (essentially moving allocations, freeing memory and the prefactor to separate functions), and looking into adding the option to also calculate the SH derivatives.