Accelerating the Initial Value Problem of Ordinary Differential Equations using GPUs for following Magnetic Field lines in a Toroidal Domain

Anikait Singh (University of California, Berkeley), advised by Dr. Stephane Ethier, Garret Wright, and Dr. Samuel Lazerson (Princeton Plasma Physics Laboratory)

Goals

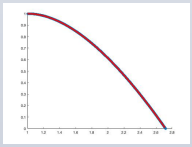
- Replacing the current LSODE solver with Sundial's CVODE solver which is compatible with CUDA
- Accelerating the Jacobian and Right Hand Side functions for the solver by converting the spline interpolation function to a CUDA kernel

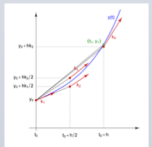
Background: Numerical Methods for Solving ODE's

- Ordinary Differential Equations is an equality involving a function and its derivatives of the form:

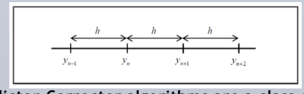
$$f\left(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \dots\right) = 0$$

- ODE's can also be expressed as a system of first order differential equations, which is used to numerically compute the solution to the initial value problem
- The Runge-Kutta-Fehlberg 4-5 algorithm is an embedded method to solve nonstiff ODE's which allows for an adaptive step size by generating a value for error by the difference between the value computed by taking intermediate steps and one larger step thus increasing the order of the method

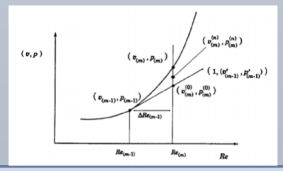




- The Adams-Moulton and Backward Differentiation algorithms are linear multistep methods that are efficient because they use n previous steps to determine the value of the dependent variable at a certain step. The Adams-Moulton method is an algorithm that works better on nonstiff equations whereas the Backward Differentiation method is better for stiff equations



- Predictor-Corrector algorithms are a class of algorithms that initially extrapolates the value for the next point and then refines the initial approximation using the predicted value and another method to interpolate the function at the same point.

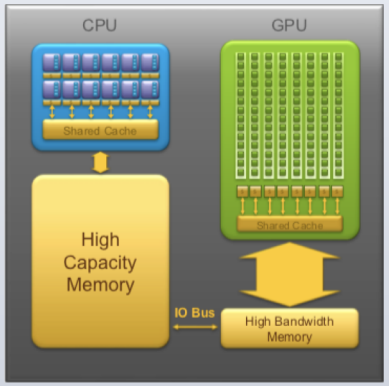


Numerical Method Chosen: Backward Differentiation with Newton-Raphson Predictor Corrector

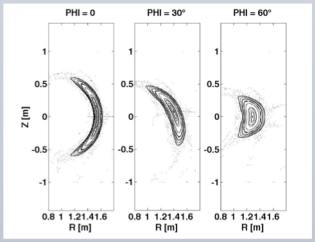
- Since the equations that are used to solve a system of two stiff first-order differential equations, the backward differentiation linear multistep method was chosen with a Newton-Raphson predictor-corrector, a variable-order, variable-step multistep method. Thus the libraries LSODE was chosen in the prior implementation of the FEILDLINES Package of the Stellarator Optimizer Library created by Dr. Samuel Lazerson.
- Since LSODE was not compatible with CUDA, the parallel computing platform for NVIDIA GPUs, the CVODE library was used to replace it (from the sundials suite of Lawrence Livermore National Labs). To interface with solver, the Spline Interpolation, Jacobian and Right Hand Side functions and kernels were written in C. CUDA was used for the transfer of memory between the CPU and GPU and the acceleration of the spline interpolation routine in the form of a kernel.

Serial vs GPU Programming

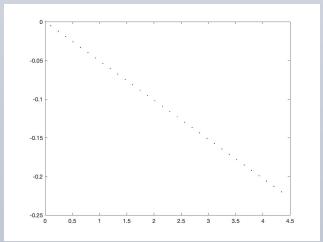
- Traditional programming is done in a serial manner where one statement is executed one after the other. This can be accelerated by having multiple threads that have distributed/independent memory which allows for multiple serial tasks to execute in a parallel manner
- GPU computing utilizes the GPU (Graphics Processing Unit) as a coprocessor. This is advantageous because of the hundreds of cores that the GPU which is much larger than 8-16 cores that a CPU has.
- The tradeoff for using a coprocessor is the copying of memory between host and device which is quite slow and can result in errors if not done properly. However, by providing read-only data, the transfer is much faster since it is unidirectional.



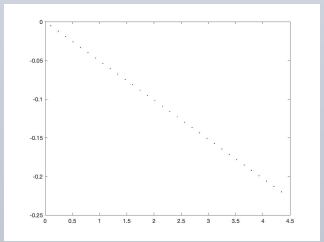
Results: Poincare Trajectories for Magnetic Field Lines



Full Field line Trajectories



LSODE



CVODE

Results: Speed Differences between LSODE and CVODE

- There was a negligible increase in speed between the multicore LSODE implementation and GPU CVODE implementation
- Some of the time that was saved from parallelizing the spline interpolation was spent on initially moving data between the device and host memory (from the CPU to GPU)
- Could see a larger amount of difference if used for a larger set of parallel calculations

Tools Utilized

- CUDA -> a parallel computing platform that can be used for general computing on GPU's
- Fortran C interface -> was able to invoke a C function from Fortran and utilize the value returned from the function
- Matlab -> was used to produce Poincare plots from the HDF5 files that were outputted from the FIELDLINES package of the STELLOPT Library along with experimenting with RKF-45
- Makefiles -> a useful method for compiling large amounts of program files in a consistent manner
- Unix Commands -> utilized emacs and other Linux tools for a more efficient programming style

Conclusions

- LSODE and CVODE had similar results to each other both in terms of the value that was outputted by the solver and the speed at which the problem was evaluated. Running CVODE serially also had similar performance as running CVODE on the GPU for the scope of this problem.
- CVODE would be a valid replacement for LSODE in the Stellarator optimizer package as an ODE solver
- Larger speedup would be possible if there was more possibility for parallelization in the Jacobian and Right Hand Side functions

REFERENCES

"CVODE." Computing, Lawrence Livermore National Laboratory, 28 June 2019, https://computing.llnl.gov/projects/sundials/cvode

Lazerson, S. A., & Chapman, I. T. (2013). STELLOPT modeling of the 3D diagnostic response in ITER. Plasma Physics and Controlled Fusion, 55(8), 084004. doi: 10.1088/0741-3335/55/8/084004

Press, W. H., Teukolsky, S. A., & Vetterling, W. T. (1992). Numerical Recipes in C The Art of Scientific Computing Second Edition. Cambridge: Cambridge University Press.

CONTACT

- Anikait Singh: asap7772@berkeley.edu
- Stephane Ethier: ethier@pppl.gov