Polynomial Homotopy Continuation on GPUs*

Jan Verschelde and Xiangcheng Yu
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago
851 South Morgan (M/C 249)
Chicago, IL 60607, USA
emails: janv@uic.edu and xiangchengyu@outlook.com

30 August 2015

Abstract

The purpose of the software presentation is to announce a library to track many solution paths defined by a polynomial homotopy on a Graphics Processing Unit (GPU). Developed on NVIDIA graphics cards with CUDA SDKs, our code is released under the GNU GPL license. Via the C interface to PHCpack, we can call our GPU library from Python.

1 Polynomial Homotopy Continuation

Polynomial homotopy continuation methods apply predictor-corrector algorithms to track solution paths defined by a homotopy, a family of polynomial systems. As corrector, a path tracker applies Newton's method, which requires the the evaluation of the polynomials and all their derivatives. The evaluated and differentiated polynomials at the current approximation for the solution provide the input for a linear system that must be solved to update that approximation.

With this paper we want to announce the availability of a novel software library for tracking solution paths defined by a polynomial homotopy on graphics processing units. This coding project represents a successful application of the QD library [3, 5] which provides double double and quad double arithmetic to improve the accuracy of numerical computations. While related to interval arithmetic [6], the QD library seems to be a lesser known solution to increase the numerical robustness. Our original motivation, originating in [10], is to offset the cost of double double and quad double arithmetic by parallel computation.

The other contribution we want to highlight is the extension of the Python package `phcpy` [9] to call the GPU accelerated path trackers. Python is a popular scripting language in computational science and is frequently used to build web applications. In this interface, the input polynomials are given as strings of human readable polynomials in conventional symbolic form. The solutions on output are returned as strings and `phcpy` provides tools to parse those strings.

This document concerns release 2.4 of PHCpack [7, 8]. The code is available at
`https://github.com/janverschelde/PHCpack`.

*This material is based upon work supported by the National Science Foundation under Grant ACI-1440534.

2 Graphics Processing Units

With Graphics Processing Units (GPU) we aim to accelerate the computations in order to achieve significant speedups over unaccelerated code executed on traditional processors. Graphics cards offer a significantly higher memory bandwidth. Their scalable execution model launches thousands of threads that perform the same sequences of instructions on different data.

In [11] we showed there is enough regularity in the evaluation and differentiation of a polynomial system, using the reverse mode of algorithmic differentiation [2], to achieve a sufficient amount of data parallelism and to keep thousands of threads occupied. As in [10], with parallelism we aimed to compensate for the cost overhead caused by the double double and quad double arithmetic, and in [11, 12, 13], we applied the CUDA version [5] of the QD library [3].

For double and double double arithmetic, the computations are memory bound and speedups are achieved via the higher memory bandwidth. Double digit speedups are obtained in complex double double and quad double arithmetic, when the problem is compute bound, when we reach a higher compute to global memory access ratio.

Recently, we improved the arithmetic circuits to evaluate and differentiate polynomials and upgraded our linear systems solvers for integration in code to track one single path of a high dimensional system [14] and to track many paths of smaller homotopies [15]. In tracking many paths, all tasks of the path tracker (predictor, corrector, adaptive step control) are performed on the device (GPU). For one single path in high dimensions, the adaptive step control can be done separately on the host (CPU).

While the computations in [11, 12] ran on randomly generated data, in [13, 14, 15] we applied our GPU code on benchmark problems, such as the cyclic n-roots problems, Pieri hypersurface intersection conditions in the numerical Schubert calculus, and the computation of all Nash equilibria in multiplayer games. When tracking many paths, the benefits of the accelerated code start to show for problems with ten of thousands of solutions. When tracking one single path, the threshold dimension to experience the benefits of acceleration is about one hundred.

3 PHCpack, PHClib, and phcpy

The C interface to PHCpack [7] was developed [4] to write relatively short distributed memory message passing programs in various parallel versions of polynomial homotopies. This interface, called PHClib, gives the C programmer access to the data stored by PHCpack, without duplication of data structures. PHClib defines the layer between PHCpack and the Python package `phcpy` [9]. By design, any upgrade to `phcpy` increases the functionality for the C and C++ programmers.

The main program that launches the accelerated path trackers starts with the definition of the polynomial homotopy and initializes the solution(s) at the start of the path(s). With PHClib, the start system can be generated and its start solutions computed. PHCpack provides condition number estimators at standard double, double double, quad double, and arbitrary precision. These estimators can be applied to determine the precision required to reach a prescribed accuracy.

Unlike the main executable program `phc` of PHCpack, which is statically linked, our GPU software is released as source code on github. The workstation that hosts our NVDIA K20C GPUs is also a web server [1] to the solvers of PHCpack, at `https://kepler.math.uic.edu`. The path trackers that run in the cloud will also become accelerated.

4 GPU Acceleration in a Python session

To illustrate calling the library to accelerate the path trackers we use a script to track the 35,940 solution paths defined by an artificial-parameter homotopy between the cyclic 10-roots problem and a random coefficient start system for the cyclic 10-roots problem. The two most relevant lines in the script are below:

```
from phcpy.trackers import gpu_double_track
cyc10sols = gpu_double_track(cyc10, cyc10q, cyc10qsols, verbose=0)
```

The first two arguments of the function call are the target and start system respectively, followed by a list of start solutions.

To measure the wall clock time, we apply the command `time`, when calling the script at the command prompt `$`, for example, as `$ time python runcyc10d.py > /tmp/output`. The output of `time` consists of three numbers. We see first the elapsed wall clock time (`real`), followed by the user CPU time (`user`), and then finally the system time (`sys`). Table 1 summarizes the runs.

precision	`real`	`user`	`sys`
double	14.980s	11.683s	3.192s
double double	45.266s	35.897s	9.228s
quad double	6m 57.368s	5m 23.224s	1m 33.266s

Table 1: Running times in minutes (m) and seconds (s) for the tracking of 35,940 paths of the cyclic 10-roots problem, in double, double double, and quad double precision on an Intel Xeon E5-2670 processor, accelerated by the NVIDIA K20C.

References

[1] N. Bliss, J. Sommars, J. Verschelde, and Xiangcheng Yu. Solving polynomial systems in the cloud with polynomial homotopy continuation. In V.P. Gerdt, W. Koepf, E.W. Mayr, and E.V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing, 17th International Workshop, CASC 2015, Aachen, Germany*, volume 9301 of *Lecture Notes in Computer Science*, pages 87–100. Springer-Verlag, 2015.

[2] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, second edition, 2008.

[3] Y. Hida, X. S. Li, and D. H. Bailey. Algorithms for quad-double precision floating point arithmetic. In *15th IEEE Symposium on Computer Arithmetic (Arith-15 2001), 11-17 June 2001, Vail, CO, USA*, pages 155–162. IEEE Computer Society, 2001. Shortened version of Technical Report LBNL-46996, software at `http://crd.lbl.gov/~dhbailey/mpdist`.

[4] A. Leykin and J. Verschelde. Interfacing with the numerical homotopy algorithms in PHCpack. In N. Takayama and A. Iglesias, editors, *Proceedings of ICMS 2006*, volume 4151 of *Lecture Notes in Computer Science*, pages 354–360. Springer-Verlag, 2006.

[5] M. Lu, B. He, and Q. Luo. Supporting extended precision on graphics processors. In A. Ailamaki and P.A. Boncz, editors, *Proceedings of the Sixth International Workshop on Data Management on New Hardware (DaMoN 2010), June 7, 2010, Indianapolis, Indiana*, pages 19–26, 2010. Software at `http://code.google.com/p/gpuprec/`.

[6] S.M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287449, 2010.

[7] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.*, 25(2):251–276, 1999.

[8] J. Verschelde. Polynomial homotopy continuation with PHCpack. *ACM Communications in Computer Algebra*, 44(4):217–220, 2010.

[9] J. Verschelde. Modernizing PHCpack through phcpy. In P. de Buyl and N. Varoquaux, editors, *Proceedings of the 6th European Conference on Python in Science (EuroSciPy 2013)*, pages 71–76, 2014.

[10] J. Verschelde and G. Yoffe. Polynomial homotopies on multicore workstations. In M.M. Maza and J.-L. Roch, editors, *Proceedings of the 4th International Workshop on Parallel Symbolic Computation (PASCO 2010), July 21-23 2010, Grenoble, France*, pages 131–140. ACM, 2010.

[11] J. Verschelde and G. Yoffe. Evaluating polynomials in several variables and their derivatives on a GPU computing processor. In *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops (PDSEC 2012)*, pages 1391–1399. IEEE Computer Society, 2012.

[12] J. Verschelde and G. Yoffe. Orthogonalization on a general purpose graphics processing unit with double double and quad double arithmetic. In *Proceedings of the 2013 IEEE 27th International Parallel and Distributed Processing Symposium Workshops (PDSEC 2013)*, pages 1373–1380. IEEE Computer Society, 2013.

[13] J. Verschelde and X. Yu. GPU acceleration of Newton's method for large systems of polynomial equations in double double and quad double arithmetic. In *Proceedings of the 16th IEEE International Conference on High Performance Computing and Communication (HPCC 2014)*, pages 161–164. IEEE Computer Society, 2014.

[14] J. Verschelde and X. Yu. Accelerating polynomial homotopy continuation on a graphics processing unit with double double and quad double arithmetic. In J.-G. Dumas and E.L. Kaltofen, editors, *Proceedings of the 7th International Workshop on Parallel Symbolic Computation (PASCO 2015), July 10-11 2015, Bath, United Kingdom*, pages 109–118. ACM, 2015.

[15] J. Verschelde and X. Yu. Tracking many solution paths of a polynomial homotopy on a graphics processing unit in double double and quad double arithmetic. In *Proceedings of the 17th IEEE International Conference on High Performance Computing and Communication (HPCC 2015)*, pages 371–376. IEEE Computer Society, 2015.