Tracking Many Solution Paths of a Polynomial Homotopy on a Graphics Processing Unit in Double Double and Quad Double Arithmetic

Jan Verschelde
joint work with Xiangcheng Yu

University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science
http://www.math.uic.edu/~jan
emails: janv@uic.edu and xiangchengyu@outlook.com

The 17th IEEE International Conference on High Performance Computing and Communications. IEEE HPCC 2015. August 24-26, 2015, New York.

Outline

1. Polynomial Homotopy Continuation
   - QD: compensating for the cost of extra precision
   - AD: evaluating and differentiating polynomials
2. SIMT Path Tracking
   - tracking paths in Single Instruction Multiple Threads mode
   - fine granularity in evaluation and differentiation
3. Applications and Computational Results
   - hardware and software
   - evaluation and differentiation
   - accelerated path tracking

Polynomial Homotopy Continuation on GPUs

1. Polynomial Homotopy Continuation
   - QD: compensating for the cost of extra precision
   - AD: evaluating and differentiating polynomials
2. SIMT Path Tracking
   - tracking paths in Single Instruction Multiple Threads mode
   - fine granularity in evaluation and differentiation
3. Applications and Computational Results
   - hardware and software
   - evaluation and differentiation
   - accelerated path tracking

polynomial homotopy continuation methods

$f(\mathbf{x}) = \mathbf{0}$ is a polynomial system we want to solve,
$g(\mathbf{x}) = \mathbf{0}$ is a start system (g is similar to f) with known solutions.

A homotopy $h(\mathbf{x}, t) = (1 - t)g(\mathbf{x}) + t\,f(\mathbf{x}) = \mathbf{0}$, $t \in [0, 1]$,
to solve $f(\mathbf{x}) = \mathbf{0}$ defines solution paths $\mathbf{x}(t)$: $h(\mathbf{x}(t), t) \equiv \mathbf{0}$.

Numerical continuation methods track the paths $\mathbf{x}(t)$, from $t = 0$ to 1.

Problem statement: when solving large polynomial systems, the hardware double precision may not be sufficient for accurate solutions.

Our goal: accelerate computations with general purpose Graphics Processing Units (GPUs) to compensate for the overhead caused by double double and quad double arithmetic.

Numerical continuation methods apply Newton's method.
In the HPCC 2014 proceedings we published our GPU implementation of Newton's method for polynomial systems.

quad double precision

A quad double is an unevaluated sum of 4 doubles,
improves working precision from 2.2×10^{-16} to 2.4×10^{-63}.

- Y. Hida, X.S. Li, and D.H. Bailey: **Algorithms for quad-double precision floating point arithmetic.** In the *15th IEEE Symposium on Computer Arithmetic*, pages 155–162. IEEE, 2001. Software at `http://crd.lbl.gov/~dhbailey/mpdist/qd-2.3.9.tar.gz`.

Predictable overhead: working with `double double` is of the same cost as working with complex numbers. Simple memory management.

The QD library has been ported to the GPU by

- M. Lu, B. He, and Q. Luo: **Supporting extended precision on graphics processors.** In the *Proceedings of the Sixth International Workshop on Data Management on New Hardware (DaMoN 2010)*, pages 19–26, 2010.
  Software at `http://code.google.com/p/gpuprec/`.

Polynomial Homotopy Continuation on GPUs

1. Polynomial Homotopy Continuation
   - QD: compensating for the cost of extra precision
   - AD: evaluating and differentiating polynomials
2. SIMT Path Tracking
   - tracking paths in Single Instruction Multiple Threads mode
   - fine granularity in evaluation and differentiation
3. Applications and Computational Results
   - hardware and software
   - evaluation and differentiation
   - accelerated path tracking

evaluating and differentiating polynomials

The data parallelism (needed for speedup on the GPU) comes from

- the reverse mode of algorithmic differentiation
- to evaluate polynomials (and their derivatives) in several variables.

The data that defines the instructions:

- Polynomials are sums of coefficients multiplied by monomials.
- Monomials are decomposed in two parts:
  1. the factor of products of powers, common in all derivatives;
  2. the product of all variables that occur in the monomial.

Example: evaluation and differentiation of $x_1 x_2 x_3 x_4 x_5$.

x_1	$x_1 \star x_2$	$x_1 x_2 \star x_3$	$x_1 x_2 x_3 \star x_4$	$x_1 x_2 x_3 x_4 \star x_5$
x_5	$x_5 \star x_4$	$x_5 x_4 \star x_3$	$x_5 x_4 x_3 \star x_2$	
	$x_1 \star x_5 x_4 x_3$	$x_1 x_2 \star x_5 x_4$	$x_1 x_2 x_3 \star x_5$	

some related work in algorithmic differentiation

- M. Grabner, T. Pock, T. Gross, and B. Kainz. Automatic differentiation for GPU-accelerated 2D/3D registration. In *Advances in Automatic Differentiation*, pages 259–269. Springer, 2008.
- G. Kozikowski and B.J. Kubica. Interval arithmetic and automatic differentiation on GPU using OpenCL. In *PARA 2012*, LNCS 7782, pages 489-503, Springer 2013.

some related work in polynomial system solving

- R.A. Klopotek and J. Porter-Sobieraj. Solving systems of polynomial equations on a GPU. In *Preprints of the Federated Conference on Computer Science and Information Systems, September 9-12, 2012, Wroclaw, Poland*, pages 567–572, 2012.
- M.M. Maza and W. Pan. Solving bivariate polynomial systems on a GPU. *ACM Communications in Computer Algebra*, 45(2):127–128, 2011.

some related work in numerical linear algebra

- D. Mukunoki and D. Takashashi. Implementation and evaluation of triple precision BLAS subroutines on GPUs. In *Proceedings of PDSEC 2012*, pages 1372–1380. IEEE Computer Society, 2012.

Polynomial Homotopy Continuation on GPUs

1. Polynomial Homotopy Continuation
   - QD: compensating for the cost of extra precision
   - AD: evaluating and differentiating polynomials

2. SIMT Path Tracking
   - tracking paths in Single Instruction Multiple Threads mode
   - fine granularity in evaluation and differentiation

3. Applications and Computational Results
   - hardware and software
   - evaluation and differentiation
   - accelerated path tracking

SIMT Path Tracking (SIMT = Single Instruction Multiple Threads)

We run the same arithmetic circuit at different data:

- threads are evaluating and differentiation the same polynomials,
- at different approximations for the solutions along the path.

path0	path1	path2
`predict`	`predict`	`predict`
`evaluate`	`evaluate`	`evaluate`
`correct`	`correct`	`correct`
`evaluate`		`evaluate`
`correct`		`correct`
		`evaluate`
		`correct`

The three stages in a predictor-corrector algorithm are:

1. `predict`: apply extrapolation to predict the next approximation,
2. `evaluate`: evaluate and differentiate the polynomials in the system,
3. `correct`: solve a linear system in Newton's method.

ordering the jobs to track paths

In order for approximations to reach the required accuracy,
some paths need two or three steps in Newton's method.

The labels to the jobs correspond to the indices of each path:

path0	path1	path2
predict	predict	predict
evaluate correct	evaluate correct	evaluate correct
evaluate correct		evaluate correct
		evaluate correct

job0	job1	job2
predict0	predict1	predict2
evaluate0 correct0	evaluate1 correct1	evaluate2 correct2
evaluate0 correct0	evaluate2 correct2	
evaluate2 correct2		

Every path has its own current value of the continuation parameter t.

The adaptive step size control is executed on the device.

The length of the total execution time is bounded from below
by the time required for the most difficult path.

launching kernels for the next round

The status of each evaluation and correction stage is either

$$-1 \text{ for failure, } 0 \text{ to continue, or } +1 \text{ for success,}$$

as determined by a check kernel.
The check kernel performs a a parallel scan to count the zeros in the status report for all paths, for example:

	path0	path1	path2	path3	path4	$\cdots$
status	0	$+1$	0	-1	0	$\cdots$
scan for 0	1	1	2	2	3	$\cdots$
job_idx $+ 1$	1		2		3	$\cdots$
path_idx	0		2		4	$\cdots$

The only number passed between the CPU and the GPU in each step is the total number of paths to continue.

Polynomial Homotopy Continuation on GPUs

1. Polynomial Homotopy Continuation
   - QD: compensating for the cost of extra precision
   - AD: evaluating and differentiating polynomials

2. SIMT Path Tracking
   - tracking paths in Single Instruction Multiple Threads mode
   - fine granularity in evaluation and differentiation

3. Applications and Computational Results
   - hardware and software
   - evaluation and differentiation
   - accelerated path tracking

pseudo code on host and device

Pseudo code on the host:

for each polynomial do
  for each monomial do
    1. compute the coefficient $c(t)$ for this monomial;
    2. evaluate the monomial and its derivative;
    3. add the values to the polynomials and to the Jacobian matrix.

Pseudo code on the device:

  launch the following three kernels
  1. compute the coefficient $c(t)$
     for all monomials in all polynomials;
  2. evaluate the monomial and its derivatives
     for all monomials in all polynomials;
  3. add to the value of the polynomial and to the Jacobian matrix
     for all monomials in all polynomials.

consecutive mode of evaluation and differentiation

For memory coalescing in the summation kernel, the matrix of evaluated and differentiated monomials is transposed.

For example, the steps in evaluating the same monomial and its derivatives at different points (a_1, a_2, a_3, a_4), (b_1, b_2, b_3, b_4), and (c_1, c_2, c_3, c_4) are illustrated below:

	$x_1 x_2 x_3 x_4$ and its four derivatives evaluated		
	path 0	path 1	path 2
0	a_1	b_1	c_1
1	$a_1 \star a_2$	$b_1 \star b_2$	$c_1 \star c_2$
2	$a_1 a_2 \star a_3$	$b_1 b_2 \star b_3$	$c_1 c_2 \star c_3$
7	$a_1 a_2 a_3 \star a_4$	$b_1 b_2 b_3 \star b_4$	$c_1 c_2 c_3 \star c_4$
6	$a_1 a_2 \star a_4$	$b_1 b_2 \star b_4$	$c_1 c_2 \star c_4$
3	$a_3 \star a_4$	$b_3 \star b_4$	$c_3 \star c_4$
4	$a_1 \star a_3 a_4$	$b_1 \star b_3 b_4$	$c_1 \star c_3 c_4$
5	$a_2 \star a_3 a_4$	$b_2 \star b_3 b_4$	$c_2 \star c_3 c_4$

layout of the memory work space

Instructions to evaluate and differentiate polynomials are defined by indices (idx) and coefficients (cff).

For each value of the continuation parameter t along a path, we store its evaluated coefficients $c(t)$, the evaluated monomials (mon), and the Jacobian matrix (Jac).

instructions		work space for multiple path					
idx	cff	evaluate-differentiate			correct		
		cff	mon	Jac	Jac^T	R	$\Delta\mathbf{x}$

Linear systems are solved with a QR decomposition.
For the corrector, we store the transpose of Jac, the upper triangular matrix R and the update $\Delta\mathbf{x}$ to the solution.

Polynomial Homotopy Continuation on GPUs

1. Polynomial Homotopy Continuation
   - QD: compensating for the cost of extra precision
   - AD: evaluating and differentiating polynomials
2. SIMT Path Tracking
   - tracking paths in Single Instruction Multiple Threads mode
   - fine granularity in evaluation and differentiation
3. Applications and Computational Results
   - hardware and software
   - evaluation and differentiation
   - accelerated path tracking

hardware and software

Our NVIDIA Tesla K20C, has 2496 cores with a clock speed of 706 MHz, is hosted by a Red Hat Enterprise Linux workstation of Microway, with Intel Xeon E5-2670 processors at 2.6 GHz.

We implemented the path tracker with the gcc compiler and version 6.5 of the CUDA Toolkit, compiled with the optimization flag `-O2`.

The code is free and open source, at github.

The benchmark data were prepared with `phcpy`, the Python interface to PHCpack.

applications: the polynomial systems

Summarizing the characteristics:

name	dim	#paths	#monomials
`cyclic10`	10	34,940	92
`nash8`	8	14,833	1,040
`pieri44`	16	24,024	3,936

Application areas:

- `cyclic10`: study of complex Hadamard matrices,
- `nash8`: totally mixed Nash equilibria in a game,
- `pieri44`: a problems from enumerative geometry.

Polynomial Homotopy Continuation on GPUs

1. Polynomial Homotopy Continuation
   - QD: compensating for the cost of extra precision
   - AD: evaluating and differentiating polynomials
2. SIMT Path Tracking
   - tracking paths in Single Instruction Multiple Threads mode
   - fine granularity in evaluation and differentiation
3. Applications and Computational Results
   - hardware and software
   - evaluation and differentiation
   - accelerated path tracking

memory bandwith

The memory bandwidth of 1,000 evaluations of the same polynomial system in complex arithmetic (GB/s):

	name	double	double double	quad double
Mon	cyclic10	190.41	124.78	25.70
	nash8	206.68	143.30	27.62
	pieri44	209.47	147.31	27.32
Sum	cyclic10	104.91	126.63	123.13
	nash8	121.38	128.52	126.56
	pieri44	87.26	80.41	77.56

where

- `Mon` is the kernel to evaluate and differentiate monomials; and
- `Sum` is the kernel to sum the evaluated monomials.

evaluation of the Jacobian matrix of `cyclic10`

Times in milliseconds for a number of evaluations
in **complex double** arithmetic (***memory bound***):

	CPU	GPU				
#evals	total	mon	sum	coeff	total	speedup
10	0.062	0.017	0.008	0.004	0.028	2.19
20	0.078	0.020	0.008	0.004	0.033	2.39
50	0.188	0.024	0.011	0.005	0.040	4.69
100	0.379	0.030	0.016	0.006	0.051	7.39
150	0.553	0.038	0.021	0.007	0.066	8.41
200	0.732	0.042	0.026	0.008	0.076	9.60
250	0.928	0.049	0.032	0.009	0.090	10.31
300	1.132	0.056	0.037	0.010	0.103	11.04
500	1.824	0.087	0.056	0.015	0.157	11.61
750	2.786	0.126	0.079	0.021	0.226	12.32
1000	3.748	0.155	0.101	0.026	0.282	13.30
1250	4.748	0.203	0.127	0.032	0.363	13.08
1500	5.563	0.235	0.149	0.039	0.423	13.14
2000	7.381	0.299	0.191	0.050	0.540	13.67
3000	11.148	0.459	0.284	0.082	0.826	13.50

evaluation of the Jacobian matrix of `cyclic10`

Times in milliseconds for a number of evaluations
in **complex double double** arithmetic (***compute bound***):

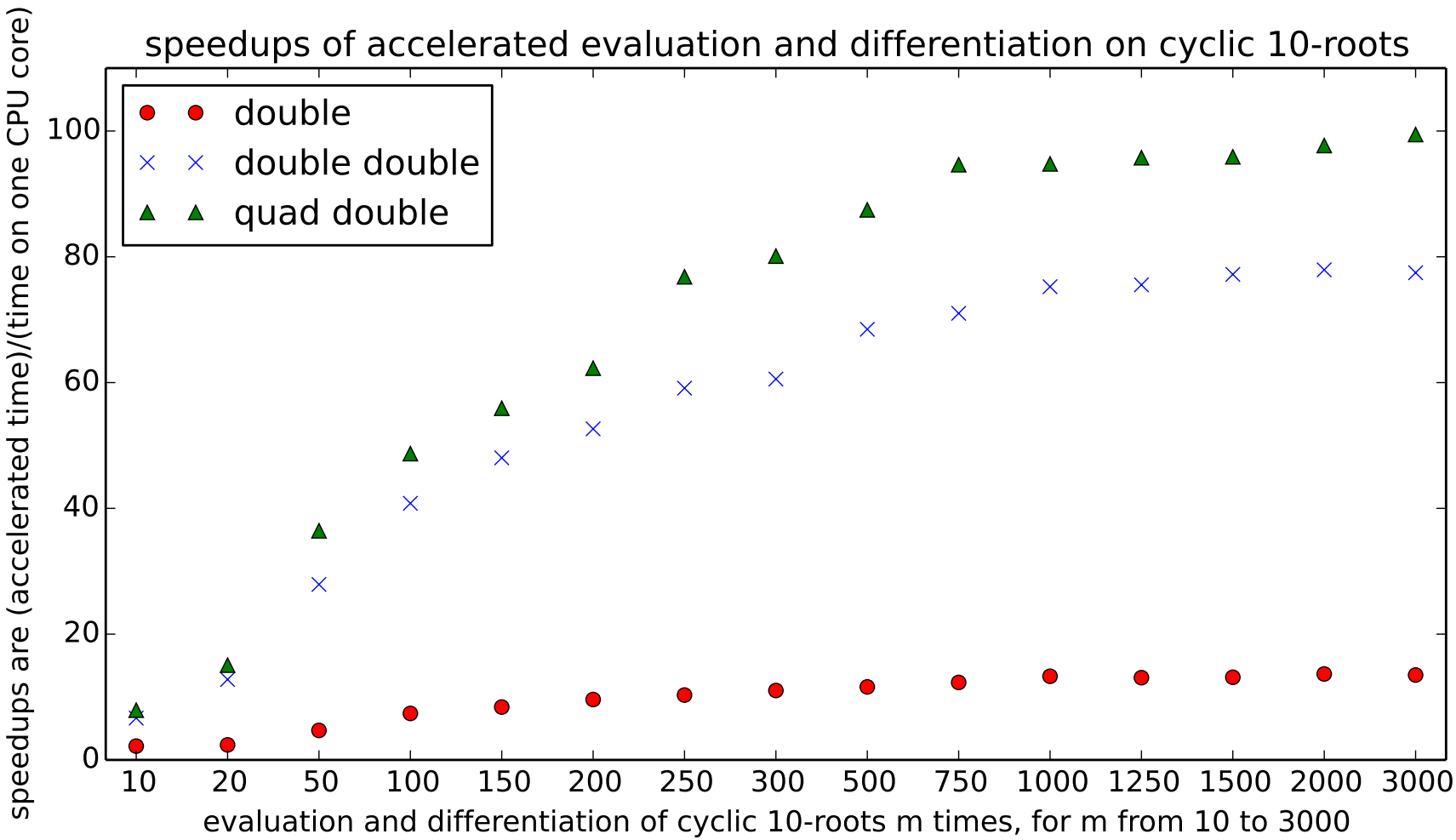
	CPU	GPU				
#evals	total	mon	sum	coeff	total	speedup
10	0.587	0.066	0.011	0.011	0.088	6.65
20	1.135	0.066	0.012	0.011	0.089	12.79
50	2.808	0.072	0.017	0.012	0.101	27.90
100	5.598	0.092	0.028	0.017	0.137	40.81
150	8.601	0.121	0.036	0.022	0.179	48.03
200	11.225	0.145	0.043	0.025	0.213	52.64
250	13.951	0.154	0.053	0.029	0.236	59.11
300	16.821	0.181	0.060	0.037	0.278	60.56
500	27.912	0.263	0.092	0.052	0.408	68.47
750	41.877	0.379	0.137	0.074	0.590	71.01
1000	55.871	0.472	0.175	0.096	0.743	75.24
1250	69.835	0.587	0.220	0.117	0.924	75.54
1500	83.920	0.691	0.257	0.139	1.087	77.20
2000	112.040	0.917	0.338	0.183	1.438	77.92
3000	167.568	1.383	0.502	0.278	2.163	77.47

evaluation of the Jacobian matrix of `cyclic10`

Times in milliseconds for a number of evaluations
in **complex quad double** arithmetic (***compute bound***):

	CPU	GPU				
#evals	total	mon	sum	coeff	total	speedup
10	5.572	0.632	0.042	0.072	0.705	7.91
20	11.129	0.622	0.043	0.073	0.738	15.07
50	27.769	0.633	0.054	0.075	0.762	36.44
100	55.566	0.931	0.080	0.130	1.141	48.70
150	83.369	1.213	0.098	0.179	1.491	55.92
200	111.027	1.438	0.120	0.224	1.782	62.29
250	138.872	1.428	0.144	0.235	1.808	76.82
300	166.546	1.641	0.161	0.277	2.079	80.11
500	277.978	2.486	0.257	0.436	3.178	87.46
750	416.268	3.435	0.369	0.594	4.398	94.64
1000	554.742	4.582	0.485	0.786	5.853	94.77
1250	694.084	5.715	0.591	0.943	7.249	95.75
1500	833.445	6.809	0.699	1.183	8.691	95.89
2000	1111.412	8.916	0.929	1.532	11.377	97.69
3000	1676.977	13.244	1.375	2.245	16.864	99.44

visualization of the speedups



Polynomial Homotopy Continuation on GPUs

1. Polynomial Homotopy Continuation
   - QD: compensating for the cost of extra precision
   - AD: evaluating and differentiating polynomials
2. SIMT Path Tracking
   - tracking paths in Single Instruction Multiple Threads mode
   - fine granularity in evaluation and differentiation
3. Applications and Computational Results
   - hardware and software
   - evaluation and differentiation
   - accelerated path tracking

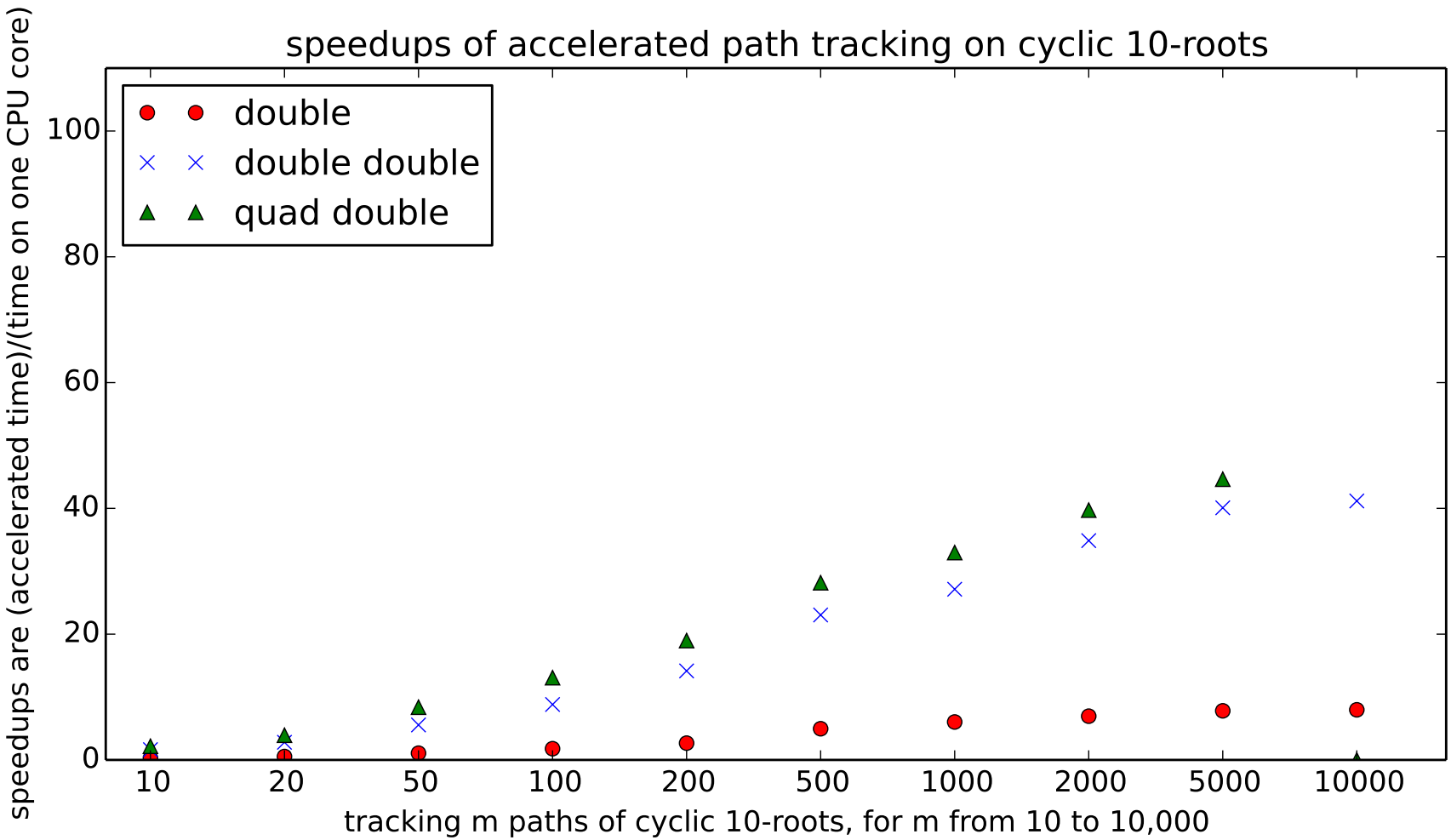
tracking paths for cyclic 10-roots

Times in seconds and speedups for tracking a number of paths of the cyclic 10-roots system.

	complex double			complex double double		
#paths	CPU	GPU	speedup	CPU	GPU	speedup
10	0.040	0.128	0.31	0.563	0.344	1.63
20	0.075	0.139	0.54	1.082	0.386	2.80
50	0.158	0.147	1.07	2.248	0.404	5.56
100	0.277	0.155	1.79	3.706	0.421	8.81
200	0.482	0.181	2.67	6.480	0.458	14.15
500	1.239	0.250	4.96	16.802	0.729	23.05
1000	2.609	0.432	6.03	35.683	1.315	27.14
2000	5.341	0.768	6.96	83.601	2.397	34.87
5000	**13.358**	1.711	7.81	210.287	5.246	40.09
10000	26.562	3.334	7.97	414.332	**10.063**	41.18

Quality Up!

visualization of the speedups



Conclusions and Outlook

When the number of solution paths in polynomial homotopies reaches several ten thousands, acceleration with GPUs achieves

1. double-digit speedup, relative to one CPU core; and
2. quality up: with acceleration we can
   - double the number of paths (the dimension of the problem),
   - double the precision (from double to double double),

and then with acceleration still compute faster than without GPU.

Future work includes

- the automatic determination of the level of precision;
- the integration with message passing and multicore parallelism.