Accelerating Polynomial Homotopy Continuation on a Graphics Processing Unit with 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 7th International Workshop on Parallel Symbolic Computation
the University of Bath, Bath, UK, 10-11 July 2015
Outline

1 Polynomial Homotopy Continuation
   - compensating for the cost of extra precision
   - the problems with path tracking

2 Accelerated Path Tracking
   - monomial evaluation and differentiation
   - accelerated predictor-corrector methods

3 Applications and Computational Results
   - hardware and software
   - matrix completion with Pieri homotopies
   - monodromy loops on cyclic $n$-roots
GPU Accelerated Path Tracking

1. Polynomial Homotopy Continuation
   - compensating for the cost of extra precision
   - the problems with path tracking

2. Accelerated Path Tracking
   - monomial evaluation and differentiation
   - accelerated predictor-corrector methods

3. Applications and Computational Results
   - hardware and software
   - matrix completion with Pieri homotopies
   - monodromy loops on cyclic $n$-roots
polynomial homotopy continuation methods

\[ f(x) = 0 \] is a polynomial system we want to solve,

\[ g(x) = 0 \] is a start system (\( g \) is similar to \( f \)) with known solutions.

A homotopy \( h(x, t) = (1 - t)g(x) + tf(x) = 0, \ t \in [0, 1] \),
to solve \( f(x) = 0 \) defines solution paths \( x(t) \):

\[ h(x(t), t) \equiv 0. \]

Numerical continuation methods track the paths \( x(t) \), 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.

Our first results (jointly with Genady Yoffe) on this goal with multicore computers are in the PASCO 2010 proceedings.
quad double precision

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


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

GPU Accelerated Path Tracking

1. Polynomial Homotopy Continuation
   - compensating for the cost of extra precision
   - the problems with path tracking

2. Accelerated Path Tracking
   - monomial evaluation and differentiation
   - accelerated predictor-corrector methods

3. Applications and Computational Results
   - hardware and software
   - matrix completion with Pieri homotopies
   - monodromy loops on cyclic $n$-roots
one coordinate of a solution path
Why is this difficult?

Tracking of one single path with the predictor-corrector method is a strictly sequential process.

Although we compute many points on a solution path, we cannot compute those points in parallel, independently from each other.

In order to move to the next point on the path, the correction for the previous point must be completed.

This difficulty requires

- a fine granularity in the parallel algorithm; and
- a sufficiently high enough threshold on the dimension.
GPU Accelerated Path Tracking

1. Polynomial Homotopy Continuation
   - compensating for the cost of extra precision
   - the problems with path tracking

2. Accelerated Path Tracking
   - monomial evaluation and differentiation
   - accelerated predictor-corrector methods

3. Applications and Computational Results
   - hardware and software
   - matrix completion with Pieri homotopies
   - monodromy loops on cyclic $n$-roots
polynomial evaluation and differentiation

We distinguish three stages:

1. Common factors and tables of power products:

\[
x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n} = x_{i_1} x_{i_2} \cdots x_{i_k} \times x_{j_1}^{e_{j_1}} x_{j_2}^{e_{j_2}} \cdots x_{j_\ell}^{e_{j_\ell}}
\]

The factor \( x_{j_1}^{e_{j_1}} x_{j_2}^{e_{j_2}} \cdots x_{j_\ell}^{e_{j_\ell}} \) is common to all partial derivatives. The factors are evaluated as products of pure powers of the variables, computed in shared memory by each block of threads.

2. Evaluation and differentiation of products of variables:

*Computing the gradient of \( x_1 x_2 \cdots x_n \) with the reverse mode of algorithmic differentiation requires \( 3n - 5 \) multiplications.*

3. Coefficient multiplication and term summation.

Summation jobs are ordered by the number of terms so each warp has the same amount of terms to sum.
monomial evaluation and differentiation

Evaluating four monomials $x_0 x_1 x_2$, $x_3 x_4 x_5$, $x_2 x_3 x_4 x_5$, $x_0 x_1 x_3 x_4 x_5$. The tidx in the tables below stands for thread index.

<table>
<thead>
<tr>
<th>tidx</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_{tidx}$</td>
<td>$x_0 x_1 x_2$</td>
<td>$x_3 x_4 x_5$</td>
<td>$x_2 x_3 x_4 x_5$</td>
<td>$x_0 x_1 x_3 x_4 x_5$</td>
</tr>
<tr>
<td>$\frac{\partial m_{tidx}}{\partial x_j}$</td>
<td>$x_0$</td>
<td>$x_3$</td>
<td>$x_2$</td>
<td>$x_1$</td>
</tr>
<tr>
<td></td>
<td>$x_0 \ast x_1$</td>
<td>$x_3 \ast x_4$</td>
<td>$x_2 \ast x_3$</td>
<td>$x_1 \ast x_2$</td>
</tr>
</tbody>
</table>

the forward calculation, from the top to bottom row

<table>
<thead>
<tr>
<th>tidx</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_{tidx}$</td>
<td>$x_0 x_1 x_2$</td>
<td>$x_3 x_4 x_5$</td>
<td>$x_2 x_3 x_4 x_5$</td>
<td>$x_0 x_1 x_3 x_4 x_5$</td>
</tr>
<tr>
<td>$\frac{\partial m_{tidx}}{\partial x_j}$</td>
<td>$x_1 \ast x_2$</td>
<td>$x_3 \ast x_4 x_5$</td>
<td>$x_2 \ast (x_4 \ast x_5)$</td>
<td>$x_2 \ast x_3 x_4 x_5$</td>
</tr>
<tr>
<td></td>
<td>$x_0 \ast x_2$</td>
<td>$x_3 \ast x_5$</td>
<td>$x_2 \ast (x_4 \ast x_5)$</td>
<td>$x_1 \ast (x_3 \ast x_4 x_5)$</td>
</tr>
<tr>
<td></td>
<td>$x_0 x_1$</td>
<td>$x_3 x_4$</td>
<td>$x_2 x_3 \ast x_5$</td>
<td>$x_1 x_2 x_3 \ast (x_5)$</td>
</tr>
</tbody>
</table>

backward and cross products, from bottom to top row
arithmetic circuits in tree mode

First to evaluate the product:

\[
x_1 x_2 \ast x_3 x_4
\]

and (storing \(x_1 x_2\) and \(x_3 x_4\)) then to compute the gradient:
computing the gradient of $x_1 x_2 \cdots x_8$

Denote by $x_{i:j}$ the product $x_i \cdots x_k \cdots x_j$ for all $k$ between $i$ and $j$.

The computation of the gradient of $x_1 x_2 \cdots x_n$ requires

- $2n - 4$ multiplications, and
- $n - 1$ extra memory locations.
collaborating threads

(a) Evaluate $x_0 x_1 x_2 x_3 x_4$

(b) Evaluate $x_0 x_1 x_2 x_3 x_4 x_5$
collaborating threads – continued

(c) Evaluate $x_0x_1x_2x_3x_4x_5x_6$

(d) Evaluate $x_0x_1x_2x_3x_4x_5x_6x_7$
GPU Accelerated Path Tracking

1. Polynomial Homotopy Continuation
   - compensating for the cost of extra precision
   - the problems with path tracking

2. Accelerated Path Tracking
   - monomial evaluation and differentiation
   - accelerated predictor-corrector methods

3. Applications and Computational Results
   - hardware and software
   - matrix completion with Pieri homotopies
   - monodromy loops on cyclic $n$-roots
accelerated predictor-corrector methods

A path tracker has three ingredients:

1. The predictor applies an extrapolation method for the next point. Each coordinate is predicted independently, linear cost in $n$.

2. The corrector applies a couple of steps with Newton’s method. Denote by $J_f$ the matrix of all partial derivatives of $f$,

\[ J_f(x) \Delta x = -f(x), \quad x := x + \Delta x. \]

3. The adaptive step length control sets the value for the step size.

When tracking one path, the step length control can be done on the host, as only some doubles are needed in the transfer.

- The device computes $\|\Delta x\|$ and $\|f(x)\|$; and then sends $\|\Delta x\|$ and $\|f(x)\|$ to the host.
- The host computes a new value for the step size $\Delta t$; and sends $\Delta t$ to device.
accelerated tracking of one single path

Input: \textit{Inst, Polynomial Instructions} \\
\hspace{1cm} \textit{W, GPU Workspace} \\
\hspace{1cm} \textit{P, parameters for path tracker} \\
Output: \textit{success or fail} \\
\hspace{1cm} \textit{W.x, solution for } t = 1 \text{ if success} \\
\hspace{1cm} t = 0 \\
\hspace{1cm} \Delta t = P.\text{max}\Delta t \\
\hspace{1cm} \#successes = 0 \\
\hspace{1cm} \#steps = 0 \\
\hspace{1cm} \text{while } t < 1 \text{ do} \\
\hspace{2cm} \text{if (}\#steps > P.\text{max}\#steps\text{) then return fail} \\
\hspace{2cm} t = \text{min}(1, t + \Delta t) \\
\hspace{2cm} \text{copy } t \text{ from host to GPU} \\
\hspace{2cm} \text{launch kernel } \text{predict}(W.x\_array, W.t\_array) \\
\hspace{2cm} \text{newton}\_success = \text{GPU}\_Newton(\textit{Inst, W, P})
adaptive step control

if (newton_success) then
    update pointer of \( W.x \) in \( W.x\_array \)
    \#successes = \#successes + 1
    if (\#successes > 2) then
        \( \Delta t = \min(2\Delta t, P.max\Delta t) \)
    else
        \#successes = 0
        \( \Delta t = \Delta t/2 \)
    \#steps = \#steps + 1
return success
accelerated Newton’s method

Input: \( Inst \), Polynomial Instructions 
\( W \), GPU Workspace 
\( P \), parameters for Newton’s method 
Output: success or fail
updated \( W \cdot x \)

\[
\text{last\_max\_eq\_val} = \text{P\_max\_eq\_val}
\]
for \( k \) from 1 to \( P\_max\_iteration \) do

\( \text{GPU\_Eval} (Inst, W) \)
\( \text{launch kernel Max\_Array} (W\_eq\_val, max\_eq\_val) \)
with single block

copy \( max\_eq\_val \) from GPU to host
if \( (max\_eq\_val > \text{last\_max\_eq\_val}) \) then return fail
if \( (max\_eq\_val < \text{P\_tolerance}) \) then return success
accelerated Newton’s method continued

GPU_Modified_Gram_Schmidt(W)
launch kernel Max_Array(W.Δx, max_Δx)
    with single block
copy max_Δx from GPU to host
launch kernel Update_x(W.x, W.Δx)
if (max_Δx < P.tolerance) then return success
last_max_eq_val = max_eq_val
return fail

A right-looking algorithm to implement the modified Gram-Schmidt method provides the most thread parallelism.
GPU Accelerated Path Tracking

1. Polynomial Homotopy Continuation
   - compensating for the cost of extra precision
   - the problems with path tracking

2. Accelerated Path Tracking
   - monomial evaluation and differentiation
   - accelerated predictor-corrector methods

3. Applications and Computational Results
   - hardware and software
   - matrix completion with Pieri homotopies
   - monodromy loops on cyclic $n$-roots
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.

The scripts `backelin.py` and `pierisystems.py` are in the examples folder of the phcpy distribution.
GPU Accelerated Path Tracking

1. Polynomial Homotopy Continuation
   - compensating for the cost of extra precision
   - the problems with path tracking

2. Accelerated Path Tracking
   - monomial evaluation and differentiation
   - accelerated predictor-corrector methods

3. Applications and Computational Results
   - hardware and software
   - matrix completion with Pieri homotopies
   - monodromy loops on cyclic $n$-roots
matrix completion with Pieri homotopies

The polynomial equations arise from minor expansions on

$$\det(A|X) = 0, \quad A \in \mathbb{C}^{n \times m},$$

and where $X$ is an $n$-by-$p$ matrix $(m + p = n)$ of unknowns.

For example, a 2-plane in complex 4-space (or equivalently, a line in projective 3-space) is represented as

$$X = \begin{bmatrix} 1 & 0 \\ x_{2,1} & 1 \\ x_{2,2} & x_{3,2} \\ 0 & x_{4,2} \end{bmatrix}.$$ 

To determine for the four unknowns in $X$ we need four equations, which via expansion results in four quadratic equations.
Pieri homotopies

Sequences of homotopies build up the solution:

\[
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
0 & x_{3,2} \\
0 & 0
\end{bmatrix},
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
0 & x_{3,2} \\
0 & x_{4,2}
\end{bmatrix},
\begin{bmatrix}
1 & 0 \\
x_{2,1} & 1 \\
0 & x_{3,2} \\
0 & x_{4,2}
\end{bmatrix},
\begin{bmatrix}
1 & 0 \\
x_{2,1} & 1 \\
x_{3,1} & x_{3,2} \\
0 & x_{4,2}
\end{bmatrix}.
\]

Pieri homotopies are defined as, for \( k \) from 1 to \( m \times p \):

\[
h(x, t) = \begin{cases} 
\det(A^{(i)} | X) = 0, & i = 1, 2, \ldots, k - 1, \\
t \det(A^{(k)} | X) + (1 - t) \det(S_X | X) = 0.
\end{cases}
\]

Tracking one single path, we start a linear system and gradually, the polynomials increase in degree and the cost of evaluation and differentiation dominates.
Pieri homotopies with double double arithmetic

\[ n \]: dimension
\[ m \]: number of predictor-corrector stages

Tracking one path of a 3-plane in 35-space, with complex double double arithmetic, times are in seconds:

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 32 | 25 | 0.05 | 0.05 | 1.1 | | | |
| 36 | 36 | 0.40 | 0.21 | 1.9 | | | |
| 40 | 19 | 0.39 | 0.15 | 2.6 | | | |
| 44 | 23 | 0.61 | 0.19 | 3.2 | | | |
| 48 | 53 | 1.69 | 0.45 | 3.8 | | | |
| 52 | 79 | 2.97 | 0.71 | 4.2 | | | |
| 56 | 28 | 1.37 | 0.29 | 4.7 | | | |
| 60 | 111 | 5.70 | 1.13 | 5.1 | | | |
| 64 | 63 | 3.36 | 0.62 | 5.4 | | | |
Tracking a 4-plane in 36-space with double doubles

\[ n \] : dimension

\[ m \] : number of predictor-corrector stages

Tracking one path of a 4-plane in 36-space, with complex double double arithmetic, times are in seconds:

<table>
<thead>
<tr>
<th>( n )</th>
<th>( m )</th>
<th>cpu</th>
<th>gpu</th>
<th>S</th>
<th>( n )</th>
<th>( m )</th>
<th>cpu</th>
<th>gpu</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
<td>0.03</td>
<td>0.03</td>
<td>1.1</td>
<td>68</td>
<td>51</td>
<td>11.3</td>
<td>0.7</td>
<td>15.6</td>
</tr>
<tr>
<td>36</td>
<td>18</td>
<td>0.20</td>
<td>0.11</td>
<td>1.9</td>
<td>72</td>
<td>39</td>
<td>19.4</td>
<td>0.8</td>
<td>23.4</td>
</tr>
<tr>
<td>40</td>
<td>12</td>
<td>0.25</td>
<td>0.09</td>
<td>2.6</td>
<td>76</td>
<td>63</td>
<td>38.8</td>
<td>1.5</td>
<td>26.4</td>
</tr>
<tr>
<td>44</td>
<td>10</td>
<td>0.28</td>
<td>0.08</td>
<td>3.2</td>
<td>80</td>
<td>157</td>
<td>115.0</td>
<td>4.0</td>
<td>28.8</td>
</tr>
<tr>
<td>48</td>
<td>16</td>
<td>0.54</td>
<td>0.14</td>
<td>3.8</td>
<td>84</td>
<td>19</td>
<td>17.3</td>
<td>0.6</td>
<td>30.0</td>
</tr>
<tr>
<td>52</td>
<td>40</td>
<td>1.52</td>
<td>0.36</td>
<td>4.3</td>
<td>88</td>
<td>574</td>
<td>431.2</td>
<td>13.3</td>
<td>32.4</td>
</tr>
<tr>
<td>56</td>
<td>36</td>
<td>1.71</td>
<td>0.36</td>
<td>4.7</td>
<td>92</td>
<td>250</td>
<td>219.8</td>
<td>6.1</td>
<td>36.2</td>
</tr>
<tr>
<td>60</td>
<td>16</td>
<td>0.95</td>
<td>0.19</td>
<td>5.0</td>
<td>96</td>
<td>190</td>
<td>230.8</td>
<td>6.2</td>
<td>37.1</td>
</tr>
<tr>
<td>64</td>
<td>75</td>
<td>4.54</td>
<td>0.84</td>
<td>5.4</td>
<td>100</td>
<td>242</td>
<td>1367.2</td>
<td>23.0</td>
<td>59.4</td>
</tr>
<tr>
<td>65</td>
<td>83</td>
<td>7.93</td>
<td>0.96</td>
<td>8.3</td>
<td>101</td>
<td>809</td>
<td>4655.7</td>
<td>70.7</td>
<td>65.9</td>
</tr>
<tr>
<td>66</td>
<td>195</td>
<td>27.20</td>
<td>2.56</td>
<td>10.6</td>
<td>102</td>
<td>1016</td>
<td>5231.7</td>
<td>75.6</td>
<td>69.2</td>
</tr>
<tr>
<td>67</td>
<td>154</td>
<td>26.43</td>
<td>2.07</td>
<td>12.8</td>
<td>103</td>
<td>375</td>
<td>2923.2</td>
<td>44.0</td>
<td>66.5</td>
</tr>
</tbody>
</table>

Jan Verschelde (UIC)  
GPU Accelerated Path Tracking  
PASCO 2015  
28 / 38
the speedups on the Pieri homotopies

speedups on 2.60 GHz CPU accelerated by K20C

accelerated Pieri homotopies in complex double double arithmetic
GPU Accelerated Path Tracking

1. Polynomial Homotopy Continuation
   - compensating for the cost of extra precision
   - the problems with path tracking

2. Accelerated Path Tracking
   - monomial evaluation and differentiation
   - accelerated predictor-corrector methods

3. Applications and Computational Results
   - hardware and software
   - matrix completion with Pieri homotopies
   - monodromy loops on cyclic $n$-roots
loops of solution paths

Consider a sequence of homotopies:

\[ h_\alpha(x, t) = \begin{cases} f(x) = 0 \\ \alpha(1 - t)L(x) + tK(x) = 0 \end{cases} \] \hspace{1cm} (1)

\[ h_\beta(x, t) = \begin{cases} f(x) = 0 \\ \beta(1 - t)K(x) + tL(x) = 0 \end{cases} \] \hspace{1cm} (2)

where

- \( K(x) = 0 \) is as \( L(x) = 0 \) another set of linear equations with different random coefficients; and
- \( \alpha \) and \( \beta \) are different random complex constants.

One loop consists in tracking one path defined by \( h_\alpha(x, t) = 0 \) and \( h_\beta(x, t) = 0 \). In both cases \( t \) goes from 0 to 1.
visualization of monodromy on a quadratic set
the cyclic $n$-roots problem

Denoted the system by $f(x) = 0$, $f = (f_1, f_2, \ldots, f_n)$, with

$$f_1 = x_0 + x_1 + \cdots + x_{n-1},$$
$$f_2 = x_0x_1 + x_1x_2 + \cdots + x_{n-2}x_{n-1} + x_{n-1}x_0,$$
$$f_i = \sum_{j=0}^{n-1-i} \prod_{k=j}^{n-i-1} x_k \mod n, \quad i = 3, 4, \ldots, n - 1,$$
$$f_n = x_0x_1x_2 \cdots x_{n-1} - 1.$$

Observe the increase of the degrees: $\deg(f_k) = k$.

High degrees are a likely cause of large roundoff errors.
Lemma (Tropical Version of Backelin’s Lemma (Adrovic-V.))

For $n = m^2 \ell$, where $\ell \in \mathbb{N} \setminus \{0\}$ and $\ell$ is no multiple of $k^2$, for $k \geq 2$, there is an $(m - 1)$-dimensional set of cyclic $n$-roots, represented exactly as

$$
\begin{align*}
  x_{km+0} &= u^k t_0 \\
  x_{km+1} &= u^k t_0 t_1 \\
  x_{km+2} &= u^k t_0 t_1 t_2 \\
  \vdots \\
  x_{km+m-2} &= u^k t_0 t_1 t_2 \cdots t_{m-2} \\
  x_{km+m-1} &= \gamma u^k t_0^{-m+1} t_1^{-m+2} \cdots t_{m-3}^{-1} t_{m-2}^{-1}
\end{align*}
$$

for $k = 0, 1, 2, \ldots, m - 1$, free parameters $t_0, t_1, \ldots, t_{m-2}$, constants $u = e^{\frac{i2\pi}{m\ell}}$, $\gamma = e^{\frac{i\pi \beta}{m\ell}}$, with $\beta = (\alpha \mod 2)$, and $\alpha = m(m\ell - 1)$. 
positive dimensional cyclic $n$-roots

Backelin’s Lemma states that there are cyclic $n$-roots of dimension $m - 1$ for $n = \ell m^2$, where $\ell$ is no multiple of $k^2$, for $k \geq 2$.

To compute the degree of the sets, we add to $f$ as many linear equations $L$ (with random complex coefficients) as the dimension of the set and count the number of solutions of the system $f(x) = 0$:

\[
\begin{align*}
\left\{ \begin{array}{c}
f(x) = 0 \\
L(x) = 0.
\end{array} \right.
\end{align*}
\]

The degree $d = m$ for $n = m^2$ and this result extends for $n = \ell m^2$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>16</th>
<th>32</th>
<th>48</th>
<th>64</th>
<th>80</th>
<th>96</th>
<th>128</th>
<th>144</th>
<th>160</th>
<th>176</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d$</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$n$</td>
<td>192</td>
<td>208</td>
<td>240</td>
<td>256</td>
<td>272</td>
<td>288</td>
<td>304</td>
<td>320</td>
<td>336</td>
<td>352</td>
</tr>
<tr>
<td>$d$</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>12</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
times in seconds on tracking one cyclic $n$-roots path

| $n$ | $m$ | CPU | GPU | s   | $n$ | $m$ | CPU | GPU | s   | $n$ | $m$ | CPU | GPU | s   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 16  | 32  | 0.00| 0.03| 0.14| 32  | 20  | 0.04| 0.06| 0.65| 32  | 0.48| 0.52| 0.92|
| 32  | 100 | 0.06| 0.16| 0.35| 79  | 1.03| 0.41| 2.53| 100 | 12.66| 3.62| 3.50|
| 48  | 103 | 0.17| 0.24| 0.72| 78  | 3.23| 0.61| 5.29| 103 | 39.46| 5.39| 7.32|
| 64  | 987 | 4.48| 4.15| 1.08| 225 | 22.94| 2.57| 8.92| 987 | 229.99| 17.93| 12.83|
| 80  | 99  | 0.73| 0.42| 1.74| 75  | 14.96| 1.15| 13.01| 99  | 180.37| 10.13| 17.81|
| 96  | 95  | 1.23| 0.52| 2.36| 69  | 23.17| 1.34| 17.26| 95  | 289.38| 12.64| 22.90|
| 112 | 171 | 3.42| 1.17| 2.92| 121 | 68.07| 2.98| 22.86| 171 | 813.91| 28.36| 28.70|
| 128 | 162 | 5.66| 1.47| 3.85| 123 | 102.94| 3.88| 26.54| 162 | 1253.82| 37.75| 33.21|
| 144 | 214 | 12.58| 2.67| 4.72| 1500| 1487.86| 61.59| 24.16| 214 | 15898.67| 479.18| 33.18|
| 160 | 68  | 4.84| 0.87| 5.53| 49  | 83.11| 2.84| 29.31| 68  | 998.43| 23.96| 41.67|
| 176 | 160 | 15.65| 2.52| 6.21| 118 | 259.80| 8.06| 32.24| 160 | 3179.81| 70.58| 45.05|
| 192 | 246 | 30.92| 9.27| 3.34| 150 | 419.16| 13.03| 32.16| 246 | 5054.70| 105.69| 47.83|
| 208 | 231 | 39.51| 5.22| 7.57| 168 | 628.46| 16.33| 38.48| 231 | 7529.02| 147.09| 51.19|
| 224 | 96  | 19.39| 2.46| 7.88| 71  | 319.27| 7.88| 40.54| 96  | 3925.33| 73.76| 53.22|
| 240 | 140 | 34.04| 4.04| 8.42| 96  | 531.01| 12.49| 42.50| 140 | 6714.01| 119.86| 56.01|
| 256 | 0   | 0.00| 1.00| 0.00| 0   | 0   | 0.00| 1.00| 0.00| 0   | 0   | 0.00| 1.00| 0.00|
| 272 | 160 | 58.19| 7.19| 8.09| 118 | 914.24| 19.12| 47.82| 160 | 10829.36| 183.12| 59.14|
| 288 | 0   | 0.00| 1.00| 0.00| 0   | 0   | 0.00| 1.00| 0.00| 0   | 0   | 0.00| 1.00| 0.00|
| 304 | 142 | 81.04| 8.05| 10.07| 103 | 1176.29| 22.87| 51.44| 142 | 13992.60| 226.78| 61.70|
| 320 | 0   | 0.00| 1.00| 0.00| 0   | 0   | 0.00| 1.00| 0.00| 0   | 0   | 0.00| 1.00| 0.00|
| 336 | 157 | 105.30| 11.12| 9.47| 114 | 1772.97| 33.26| 53.31| 157 | 20807.27| 327.25| 63.58|
| 352 | 121 | 93.89| 9.78| 9.60| 90  | 1621.15| 28.75| 56.39| 121 | 18881.13| 290.36| 65.03|
speedups on one path of cyclic $n$-roots

In double precision, the dimensions are too small for good speedups. Double digits speedups in double double and quad double precision are achieved once $n = 64$. 
conclusions

We can compensate for the cost of double double arithmetic when tracking one solution path with GPU acceleration.

Double double and quad double arithmetic (using QD):

- **memory bound** for double and (real) double double arithmetic,
- **compute bound** for complex double doubles and quad doubles.

*Double digit speedups $\Rightarrow$ double the precision, compute twice as fast.*

We achieve not only speedup, but also *quality up*, and in some hard cases double precision is insufficient for a successful path tracking.

For *many* solution paths:

Tracking many solution paths of a polynomial homotopy on a graphics processing unit with double double and quad double arithmetic.