Parallel Multiple Double Precision to Solve Polynomial Systems

Jan Verschelde†

University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science

www.phcpack.org

HILT (High Integrity Language Technologies) 2020
16-17 November 2020, online

†Supported by the National Science Foundation, grant DMS 1854513.

Outline

1. Problem Statement
   - cost overhead of multiple double precision
   - solving polynomial systems
2. Parallel Newton
   - accuracy loss in a lower triangular block Toeplitz solver
   - medium grained parallelism
3. Computational Experiments
   - efficiency plots for random data

problem statement

+ Parallel computers are fast,
+ and can solve large problems,
− but the propagation of roundoff errors increases,
− and the hardware supports only double precision.

Quality Up: If we can afford the same time time as on a sequential run, how much can we improve the quality of the results on a parallel run?

Specific for this talk: compensate for the cost overhead of multiple double precision arithmetic with parallel computations.

software libraries for multiple double arithmetic

- Y. Hida, X. S. Li, and D. H. Bailey:

Algorithms for quad-double precision floating point arithmetic.
In the *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pages 155–162, 2001.

- M. Joldes, J.-M. Muller, V. Popescu, and W. Tucker:

CAMPARY: Cuda Multiple Precision Arithmetic Library and Applications.
In Mathematical Software – ICMS 2016, the 5th International Conference on Mathematical Software, pages 232-240, Springer-Verlag, 2016.

error free transformations

```
Computing the 2-norm of a vector of dimension 64
of random complex numbers on the unit circle equals 8.
Observe the second double of the multiple double 2-norm.

double double : 8.00000000000000E+00 - 6.47112461314111E-32
triple double : 8.00000000000000E+00 + 1.78941597340672E-48
  quad double : 8.00000000000000E+00 + 3.20475411419393E-65
 penta double : 8.00000000000000E+00 + 2.24021706293649E-81
  octo double : 8.00000000000000E+00 - 9.72609915198313E-129
  deca double : 8.00000000000000E+00 + 3.05130075600701E-161
```

multiple double operation count

for 2, 3, 4, 5, 8, and 10 double arithmetic

	double double				triple double				quad double			
	+	−	∗	/	+	−	∗	/	+	−	∗	/
add	8	12			13	22			35	54		
mul	5	9	9		83	84	42		99	164	73	
div	33	18	16	3	113	214	63	4	266	510	112	5

	penta double				octo double				deca double			
	+	−	∗	/	+	−	∗	/	+	−	∗	/
add	44	78			95	174			139	258		
mul	162	283	109		529	954	259		952	1743	394	
div	474	898	175	6	1599	3070	448	9	2899	5598	700	11

the software package PHCpack

- PHC stands for Polynomial Homotopy Continuation.
  `phc -b` is a blackbox solver for polynomial systems.
- ACM Transactions on Mathematical Software archived version 1.0 in 1999. The code is developed with the GNU Ada compiler.
- Ada works well for a large research software package.
  Its multitasking is effective and convenient to define in code.
- Use of multitasking on multicore processors was described in the proceedings of PASCO 2010 (PASCO = Parallel Symbolic Computation) in a joint paper with Genady Yoffe.
- NSF Award 1440534 of the SI2-SSE program led to the acquisition of a CentOS Linux workstation with 256 GB RAM and two 22-core 2.2 GHz Intel Xeon E5-2699 processors. Under this award, PHCpack was further developed into a Sustainable Software Element, available on github.

Parallel Newton

One step of Newton's method requires

1. evaluation and differentiation, and
2. the solution of a linear system.

Consider $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ as a system of polynomials in several variables, with coefficients as truncated power series in the variable t.

We compute $\mathbf{x}(t)$ a power series solution to $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, starting at a point $\mathbf{x}(0) = \mathbf{z}$, $\mathbf{x}(t) = \mathbf{z} + \mathbf{x}_1 t + \mathbf{x}_2 t^2 + \cdots$.

With linearization, instead of vectors and matrices of power series, we consider power series with vectors and matrices as coefficients. (Joint with Nathan Bliss, *Linear Algebra and Its Applications, 2018*.)

Now we can make our problem statement more specific ...

error analysis of a lower triangular block Toeplitz solver

joint with Simon Telen and Marc Van Barel, in the CASC 2020 proceedings

Solving $(A_0 + A_1 t + A_2 t^2 + \cdots + A_i t^i)(x_0 + x_1 t + x_2 t^2 + \cdots + x_i t^i)$
$= (b_0 + b_1 t + b_2 t^2 + \cdots + b_i t^i)$

leads to a lower triangular block system:

$$\begin{bmatrix} A_0 & & & & \\ A_1 & A_0 & & & \\ A_2 & A_1 & A_0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ A_i & A_{i-1} & A_{i-2} & \cdots & A_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_i \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_i \end{bmatrix}.$$

Let κ be the condition number of A_0. Let $\|A_0\| = \|x_0\| = 1$, $\|x_i\| \approx \rho^i$.
In our context, $\rho \approx 1/R$, where R is the convergence radius.

If $\|A_i\| \approx \rho^i$, then $\dfrac{\|\Delta x_i\|}{\|x_i\|} \approx \kappa^{i+1} \epsilon_{\text{mach}}$, and accuracy is lost.

With multiple double precision, a small ϵ_{mach} gives accurate results.

medium grained parallelism

With multiple double arithmetic, programs become compute bound.

One job takes at least several minutes.

A crew of worker tasks processes a queue of jobs.

Medium grained parallelism is applied:

1. To evaluate and differentiate a system of polynomials, one job is concerned with one polynomial.
2. The block triangular linear system is solved with pipelining.

computational experiments

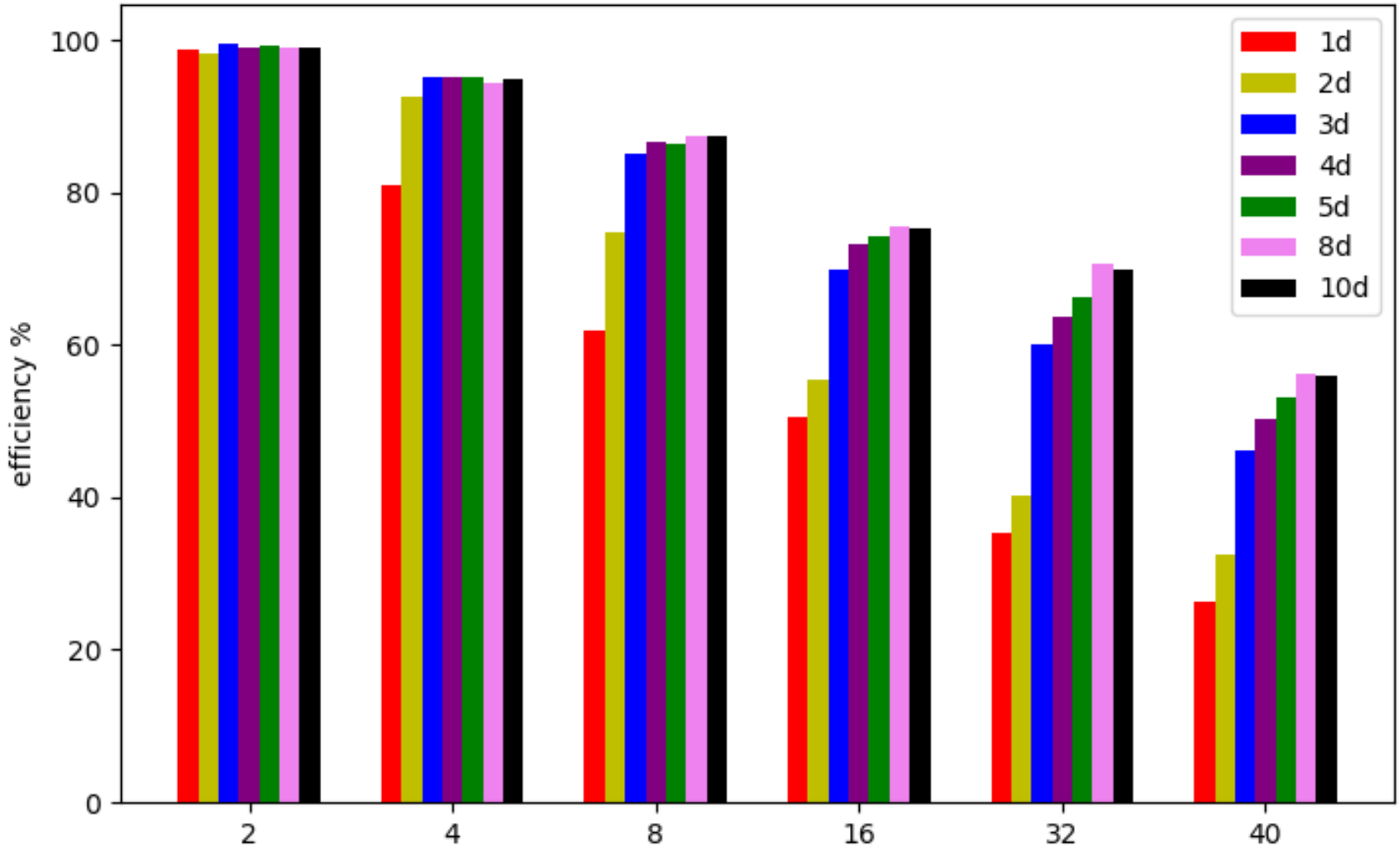
Runs done on a CentOS Linux workstation with 256 GB RAM and two 22-core 2.2 GHz Intel Xeon E5-2699 processors.

Generated polynomial systems of

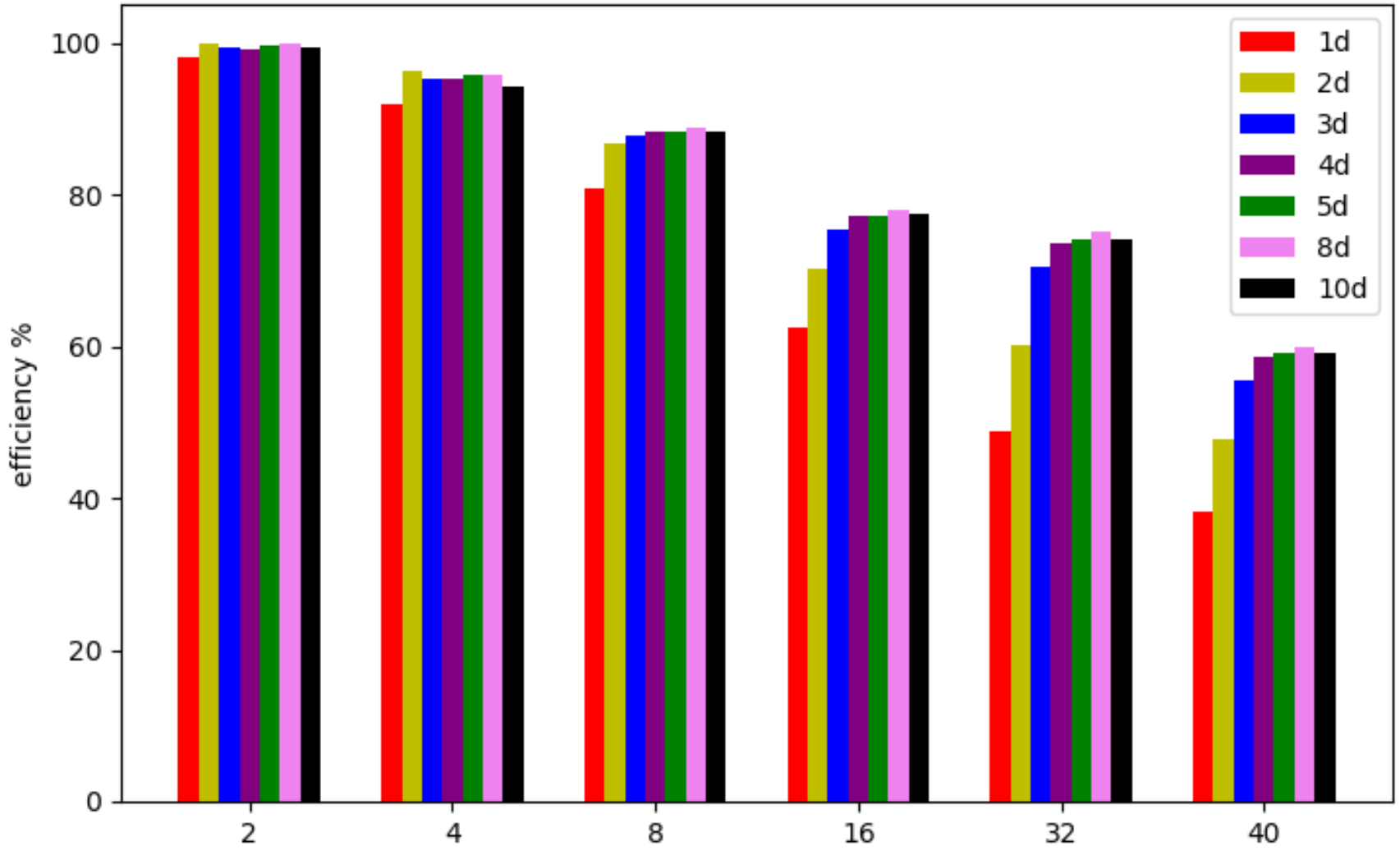
- 64 polynonomials with 64 monomials per polynomial,
- considered power series of degrees 8, 16, 32, and
- reported efficiencies for 2, 4, 8, 16, 32, and 40 worker threads.

Efficiency = speedup divided by the number of worker threads.

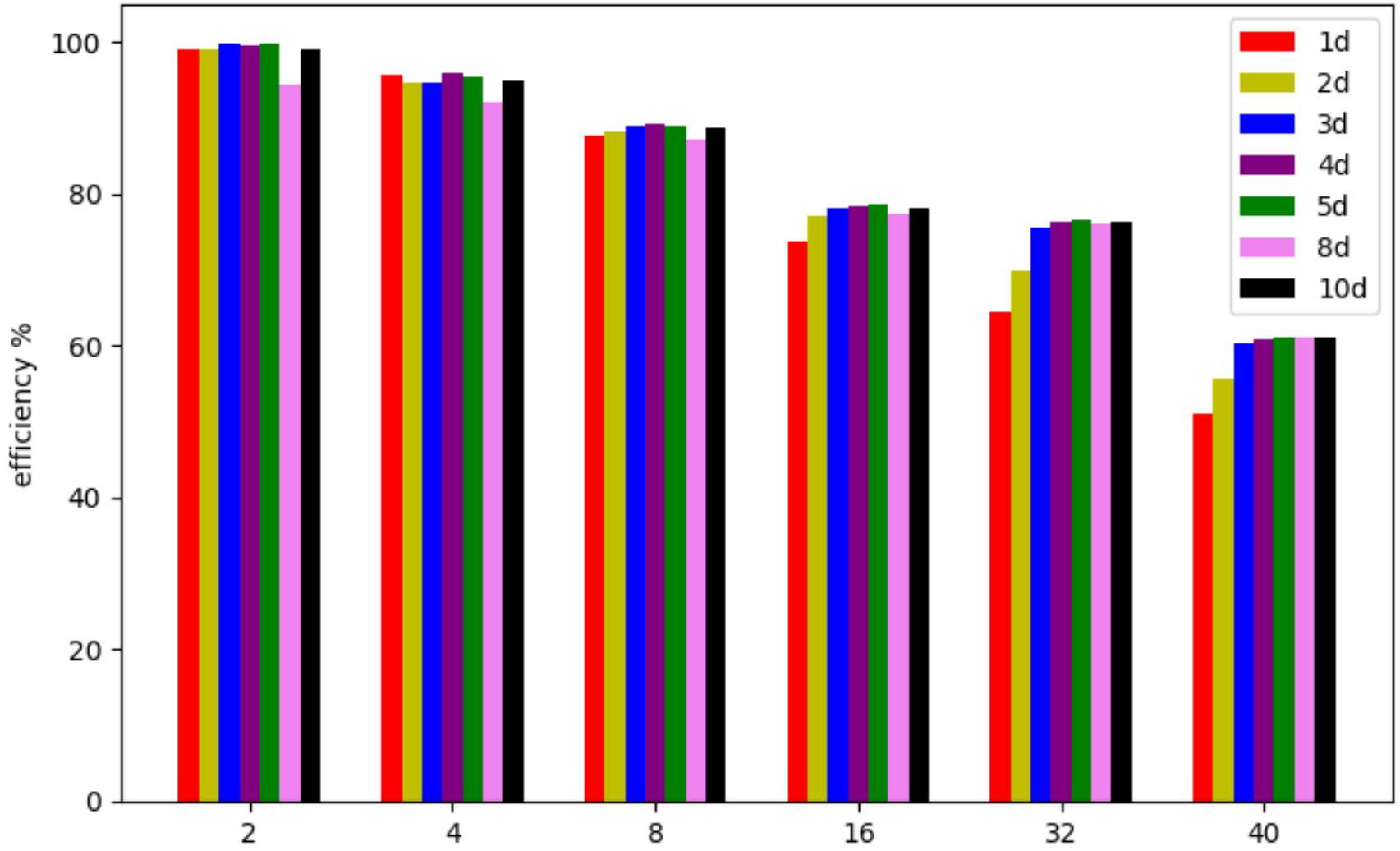
power series of degree 8



power series of degree 16



power series of degree 32



conclusions

As programs become compute bound with multiple double arithmetic, the efficiencies improve already significantly in triple double precision.

However, the efficiency is limited by the medium grained parallelism.

The Ada 202X parallel features look a promising development to overcome this limitation.