Robust Numerical Tracking of One Path of a Polynomial Homotopy on Parallel Shared Memory Computers

Jan Verschelde
joint work with Simon Telen and Marc Van Barel

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

The 22th Workshop on Computer Algebra in Scientific Computing
14-18 September 2020, Linz, Austria (online)

Outline

1. Problem Statement
   - cost overhead of our a priori adaptive step control
   - error analysis of a lower triangular block Toeplitz solver
2. Parallel Algorithms
   - algorithmic differentiation for Jacobians and Hessians
   - pipelined solution of matrix series
3. Computational Experiments
   - estimating the distance to the nearest different path
   - estimating the radius of convergence of the series

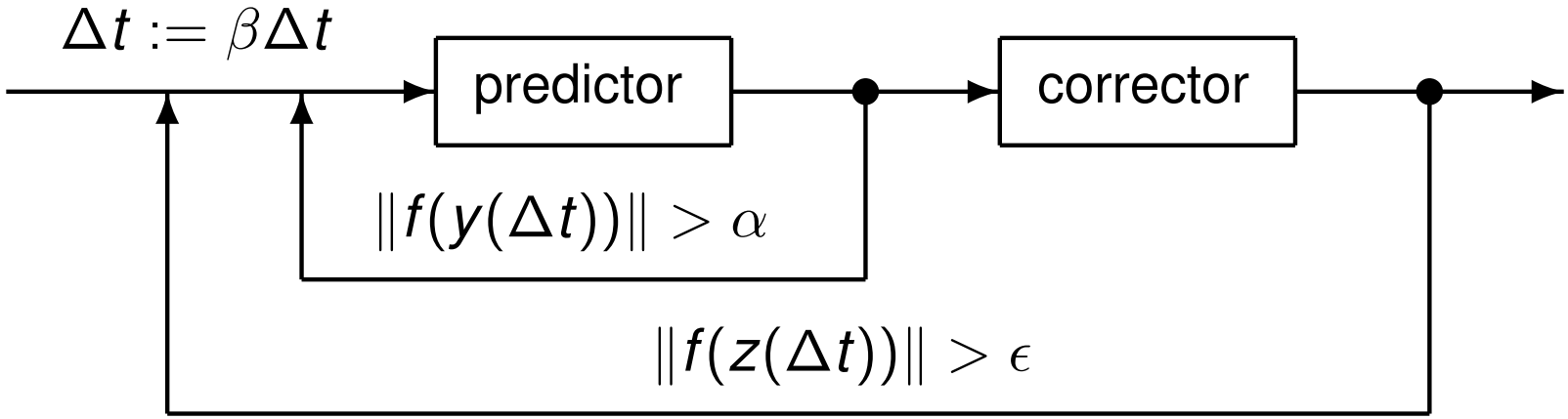
adaptive step control in predictor-corrector methods

To solve a polynomial system $f(x) = 0$, the homotopy

$$h(x,t) = \gamma(1-t)g(x) + t\,f(x) = 0, \quad t \in [0,1],$$

defines solution paths $x(t)$ satisfying $h(x(t),t) = 0$, starting at the start solutions of $g(x) = 0$. Paths $x(t)$ are regular, for a random γ, for $t < 1$.

An *a posteriori* step control uses feedback loops:



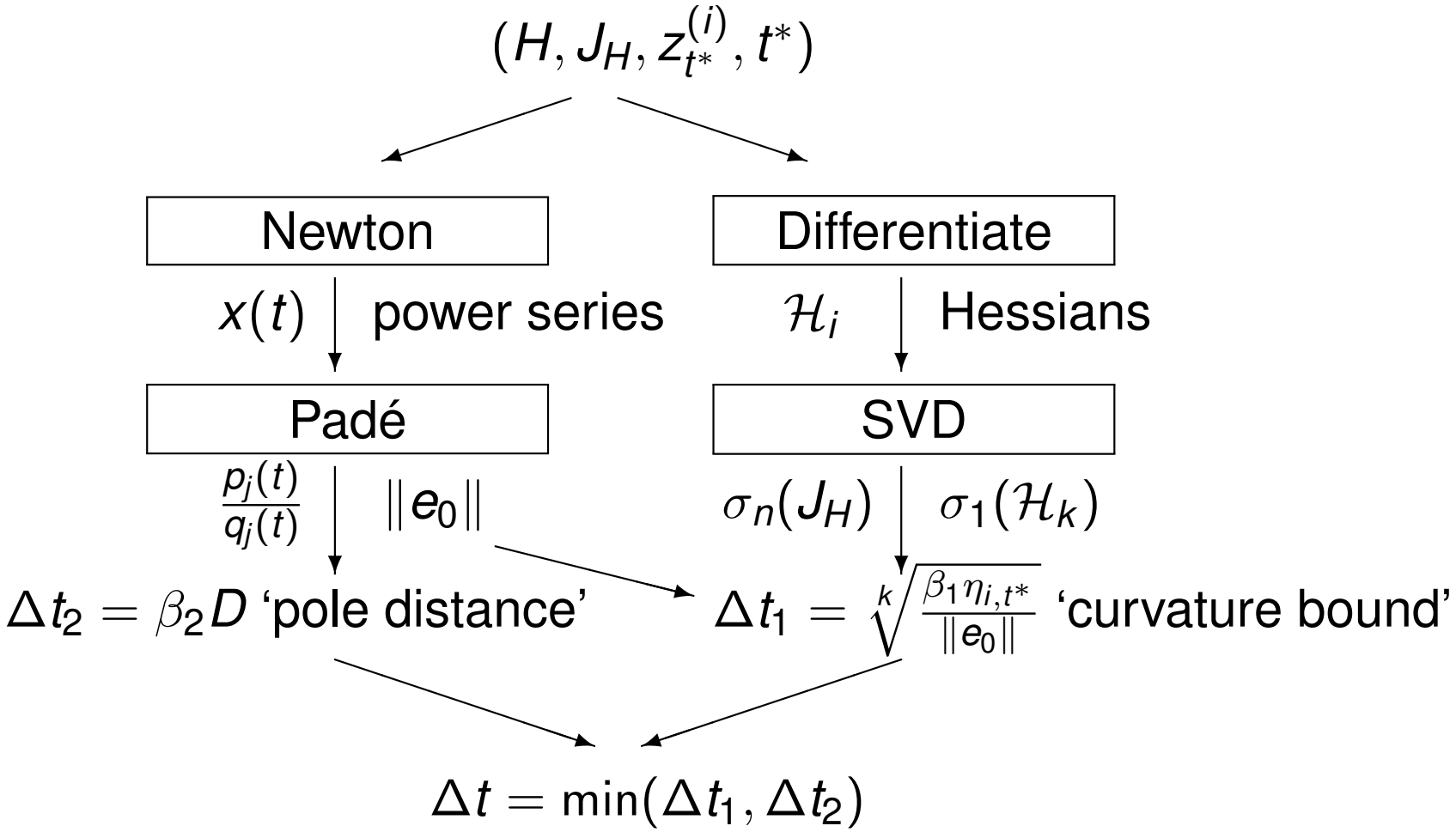
Variable step size Δt control in predictor-corrector methods:

- Δt is too large: divergence and/or path jumping.
- Δt is too small: inefficient, we care only about $x(1)$.

five most relevant papers

1. E. Fabry. Sur les points singuliers d'une fonction donnée par son développement en série et l'impossibilité du prolongement analytique dans des cas très généraux. *Annales scientifiques de l'École Normale Supérieure*, 13:367–399, 1896.
2. H. Schwetlick and J. Cleve. Higher order predictors and adaptive steplength control in path following algorithms. *SIAM Journal on Numerical Analysis*, 24(6):1382–1393, 1987.
3. 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.
4. N. Bliss and J. Verschelde. The method of Gauss–Newton to compute power series solutions of polynomial homotopies. *Linear Algebra and its Applications*, 542:569–588, 2018.
5. S. Telen, M. Van Barel, and J. Verschelde. A Robust Numerical Path Tracking Algorithm for Polynomial Homotopy Continuation. `arXiv:1909.04984v2`, accepted by *SIAM J. Sci. Computing*.

schematic of our a priori adaptive step control



Problem: compared to the a posteriori adaptive step control, the cost overhead factor is $O(n)$, for n-dimensional systems.

error analysis of a lower triangular block Toeplitz solver

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 $\|A_0\| = \|x_0\| = 1$, $\|x_i\| \approx \rho^i$.

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.

Problem: multiple double precision causes a cost overhead factor.

parallel algorithms to compute estimates C and R

our a priori adaptive set control algorithm is justified in arXiv:1909.04984v2

$$C = \frac{2\sigma_n(J)}{\sqrt{\sigma_{1,1}^2 + \sigma_{2,1}^2 + \cdots + \sigma_{n,1}^2}},$$

estimates the nearest different solution path, where

- $\sigma_n(J)$ is the smallest singular value of the Jacobian J,
- $\sigma_{k,1}$ is the largest singular value of the Hessian of the k-th polynomial, in a system with n equations.

R estimates the radius of convergence of the power series.

Applying the theorem of Fabry, R is computed as the ratio of the moduli of two consecutive coefficients in the series, truncated at d:

$$x(t) = c_0 + c_1 t + c_2 t^2 + \cdots + c_d t^d, \quad z = c_{d-1}/c_d, \quad R = |z|,$$

where z estimates the location of the nearest singular parameter value.

reverse mode of algorithmic differentiation

To evaluate and differentiate $f = x_1 x_2 x_3 x_4 x_5$, proceed as follows:

1) forward products:

$$\begin{aligned} x_1 x_2 &= x_1 \star x_2 \\ x_1 x_2 x_3 &= x_1 x_2 \star x_3 \\ x_1 x_2 x_3 x_4 &= x_1 x_2 x_3 \star x_4 = \partial f/\partial x_5 \\ x_1 x_2 x_3 x_4 x_5 &= x_1 x_2 x_3 x_4 \star x_5 = f \end{aligned}$$

2) backward products:

$$\begin{aligned} x_5 x_4 &= x_5 \star x_4 \\ x_5 x_4 x_3 &= x_5 x_4 \star x_3 \\ x_5 x_4 x_3 x_2 &= x_5 x_4 x_3 \star x_2 = \partial f/\partial x_1 \end{aligned}$$

3) cross products:

$$\begin{aligned} x_1 x_3 x_4 x_5 &= x_1 \star x_5 x_4 x_3 = \partial f/\partial x_2 \\ x_1 x_2 x_4 x_5 &= x_1 x_2 \star x_5 x_4 = \partial f/\partial x_3 \\ x_1 x_2 x_3 x_5 &= x_1 x_2 x_3 \star x_5 = \partial f/\partial x_4 \end{aligned}$$

For a product of n variables, with $3n - 5$ multiplications, we evaluate and compute all partial derivatives.

Hessians can be computed efficiently from the gradients.

medium grained parallelism

To evaluate and differentiate a system of n polynomial equations, we consider n jobs:

1. Evaluate each polynomial and its gradient.
2. Compute for each polynomial its Hessian matrix.

These n jobs can be computed independently from each other.

In case the polynomials differ much in size,
then dynamic load balancing improves the speedup.

The same type of job scheduling is applied in the other tasks:

1. Computing the rational approximations for the predictor.
2. Shifting the coefficients in the homotopy to reset to $t = 0$.

solving a matrix series

We solve $\mathbf{A}(t)\mathbf{x}(t) = \mathbf{b}(t)$ for series $\mathbf{x}(t)$,
given $\mathbf{A}(t) = A_0 + A_1 t + A_2 t^2 + \cdots$ and $\mathbf{b}(t) = b_0 + b_1 t + b_2 t^2 + \cdots$.

For example, for series truncated at degree 4:

$$\begin{aligned}
A_0 x_0 &= b_0 \\
A_0 x_1 &= b_1 - A_1 x_0 \\
A_0 x_2 &= b_2 - A_2 x_0 - A_1 x_1 \\
A_0 x_3 &= b_3 - A_3 x_0 - A_2 x_1 - A_1 x_2 \\
A_0 x_4 &= b_4 - A_4 x_0 - A_3 x_1 - A_2 x_2 - A_1 x_3.
\end{aligned}$$

First factor A_0, $F = \text{Factor}(A_0)$, and compute $x_0 = \text{Solve}(F, b_0)$.

1. update b_k with $b_k - A_k x_0$ simultaneously, $k = 1, 2, 3, 4$.
2. $x_1 = \text{Solve}(F, b_1)$
3. update b_k with $b_k - A_k x_1$ simultaneously, $k = 2, 3, 4$; etc...

a pipelined algorithm to solve a matrix series

Let $\mathbf{A}(t)\mathbf{x}(t) = \mathbf{b}(t)$ be a series truncated to degree d.

1. $F = \text{Factor}(A_0)$
2. $x_0 = \text{Solve}(F, b_0)$
3. for k from 1 to d do
   1. update b_ℓ with $b_\ell - A_\ell x_k$ simultaneously, for ℓ from k to d
   2. $x_k = \text{Solve}(F, b_k)$

With d threads, the speedup is then

$$\frac{d(d-1)/2 + 2(d+1)}{2(d+1)} = 1 + \frac{d(d-1)}{4(d+1)}.$$

As $d \to \infty$, this ratio equals $1 + d/4$.

one cyclic n-root, $n = 64, 96, 128$

Our computational experiments run on two 22-core 2.2 GHz Intel Xeon E5-2699 processors in a CentOS Linux workstation with 256 GB RAM.

The proceedings paper reports detailed experimental times on randomly generated problems data for all stages in the algorithm.

Let us focus on the cyclic n-roots problem:

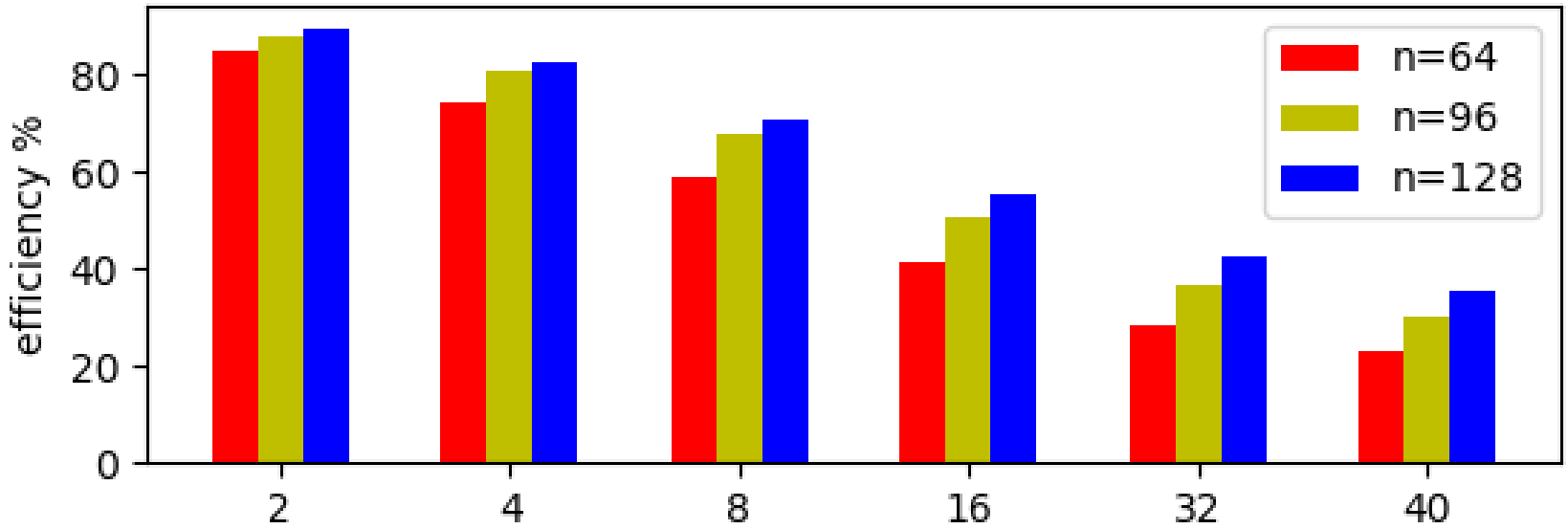
$$\begin{cases} x_0 + x_1 + \cdots + x_{n-1} = 0 \\ i = 2, 4, \ldots, n-1 : \sum_{j=0}^{n-1} \prod_{k=j}^{j+i-1} x_{k \bmod n} = 0 \\ x_0 x_1 x_2 \cdots x_{n-1} - 1 = 0. \end{cases}$$

Problem setup: selected a generic point on a 7-dimensional surface for $n = 64, 128$, and on a 3-dimensional surface for $n = 96$.

All reported computations happen in quad double precision.

estimating the distance to the nearest different path

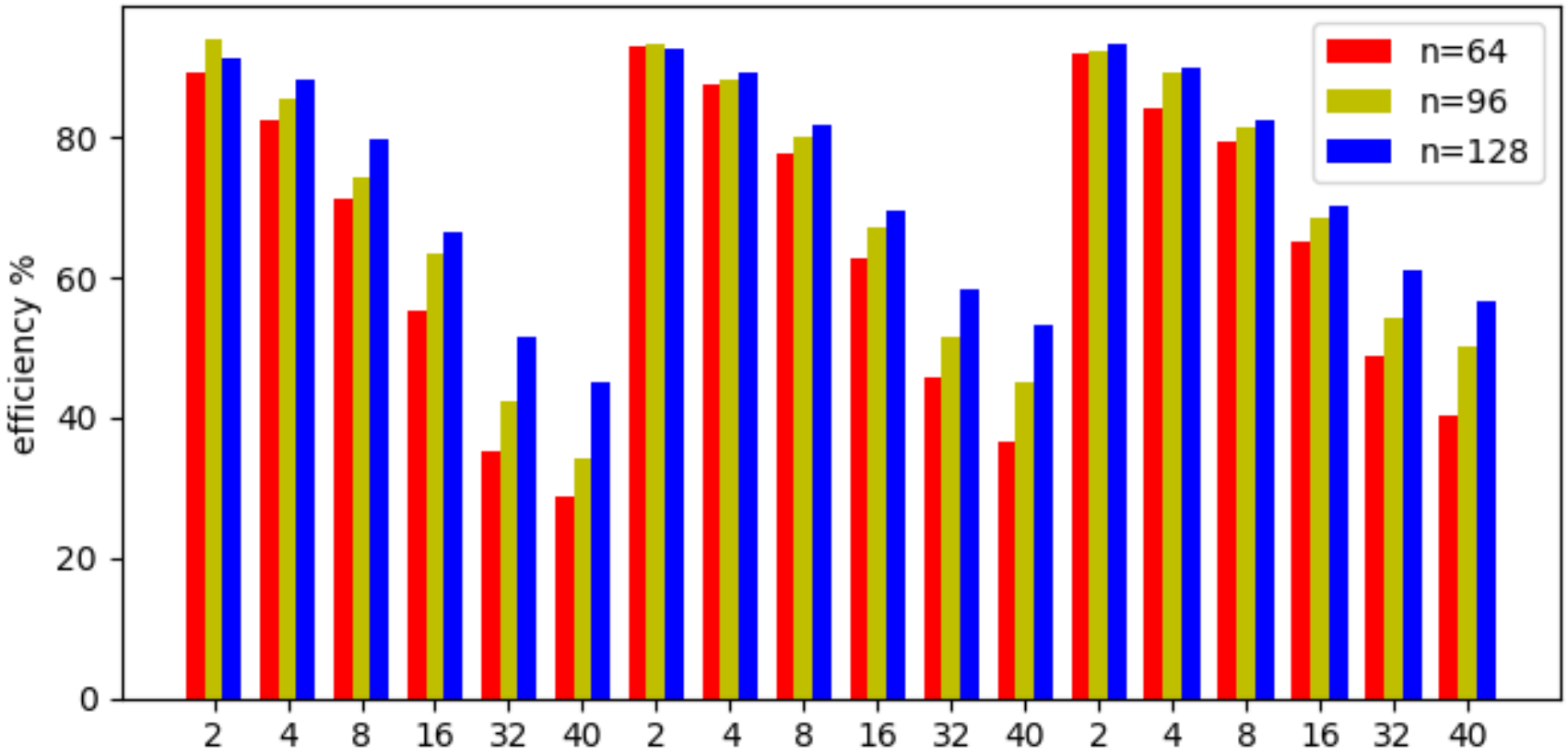
p is the number of threads, $S(p)$ is the speedup,
$E(p) = S(p)/p$ is the efficiency, shown for $p = 2, 4, 8, 16, 32, 40$:



Efficiency decreases for increasing p and increases for increasing n.

estimating the radius of convergence of the series

p is the number of threads, $S(p)$ is the speedup,
$E(p) = S(p)/p$ is the efficiency, shown for $p = 2, 4, 8, 16, 32, 40$,
in three groups, respectively for degrees $d = 8, 16, 24$:



observed convergence of Newton's method

Newton's method on a truncated series, truncated to degree d, converges if the norm of the update $\|\Delta x_d\|$ is sufficiently small.

Let κ denote the estimated inverse condition of the condition number of the Jacobian matrix A_0.

		$\|\Delta x_d\|$		
n	κ	$d = 8$	$d = 16$	$d = 24$
64	3.9E−05	4.6E−44	1.1E−24	4.1E−05
96	2.0E−04	1.4E−47	9.6E−31	7.3E−14
128	4.6E−06	2.2E−30	†	†

†For $n = 128$, for $d = 16$ and 24, the largest maximum norm of the update less than one occurs at the coefficients with t^{15} and equals about 1.1E−01. So quad double precision is insufficient.

conclusions

Parallel algorithms reduce the cost overhead factors caused by

1. our a priori adaptive step size control algorithm, and
2. multiple double precision.

The medium grained parallelism (at the polynomial level) is efficient

- for a modest number of threads, and
- improves for systems with a larger number of polynomials.

But quad double precision may not suffice
if too many terms in the power series are required.