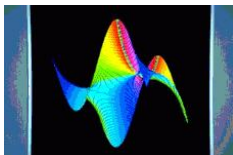


Fast Computation of Power Series Solutions of Systems of Differential Equations

Bruno Salvy

`Bruno.Salvy@inria.fr`

Algorithms Project, Inria



Joint work with

A. Bostan, F. Chyzak, F. Ollivier, É. Schost, A. Sedoglavic

Series Expansions to High Order

Motivations:

- combinatorics (generating functions of ordered structures);
- numerical analysis (e.g., Padé approximants);
- cryptography (e.g., morphisms between curves).

Non-linear differential systems can be reduced to linear ones.

Problem

Input: linear differential equation/system of order r with power series coefficients, positive integer $N \gg r$.

Output: N first terms of one solution or of a basis of solutions.

Complexity measure: number of arithmetic operations.

Aim: quasi-linear wrt N , good wrt r .

$M(N) = \mathcal{O}(N \log N)$ polynomial multiplication [Schönhage-Strassen71];

$MM(r, N) = \mathcal{O}(r^2 M(N) + r^\omega N)$ polynomial matrix product [Bostan-Schost05].

Results

input /output	power series coefficients	polynomial coefficients	constant coefficients	output size
equation /basis	$\mathcal{O}(r^2 N \log N)$ *	$\mathcal{O}(dr^2 N)$	$\mathcal{O}(rN)$ *	rN
equation /1 soln	$\mathcal{O}(rN \log^2 N)$ *	$\mathcal{O}(drN)$	$\mathcal{O}(\log rN)$ *	N
system /basis	$\mathcal{O}(r^2 N \log N)$ *	$\mathcal{O}(dr^\omega N)$	$\mathcal{O}(r^2 \log rN)$ *	$r^2 N$
system /1 soln	$\mathcal{O}(r^2 N \log^2 N)$ *	$\mathcal{O}(dr^2 N)$	$\mathcal{O}(r \log rN)$ *	rN

Previous results:

- $\mathcal{O}(r^2 N^2)$ undetermined coefficients;
- $\mathcal{O}(r^r N \log N)$ [Brent-Kung78];
- $\mathcal{O}(r^2 N \log^2 N)$ [van der Hoeven02].

Results

input /output	power series coefficients	polynomial coefficients	constant coefficients	output size
equation /basis	$\mathcal{O}(r^2 N)^*$	$\mathcal{O}(dr^2 N)$	$\mathcal{O}(rN)^*$	rN
equation /1 soln	$\mathcal{O}(rN)^*$	$\mathcal{O}(drN)$	$\mathcal{O}(N)^*$	N
system /basis	$\mathcal{O}(r^2 N)^*$	$\mathcal{O}(dr^\omega N)$	$\mathcal{O}(r^2 N)^*$	$r^2 N$
system /1 soln	$\mathcal{O}(r^2 N)^*$	$\mathcal{O}(dr^2 N)$	$\mathcal{O}(rN)^*$	rN

Previous results:

- $\mathcal{O}(r^2 N^2)$ undetermined coefficients;
- $\mathcal{O}(r^r N)$ [Brent-Kung78];
- $\mathcal{O}(r^2 N)$ [van der Hoeven02].

Results

input /output	power series coefficients	polynomial coefficients	constant coefficients	output size
equation /basis	$\mathcal{O}(r^2 N)^*$	$\mathcal{O}(dr^2 N)$	$\mathcal{O}(rN)^*$	rN
equation /1 soln	$\mathcal{O}(rN)^*$	$\mathcal{O}(drN)$	$\mathcal{O}(N)^*$	N
system /basis	$\mathcal{O}(r^2 N \log N)^*$	$\mathcal{O}(dr^\omega N)$	$\mathcal{O}(r^2 N)^*$	$r^2 N$
system /1 soln	$\mathcal{O}(r^2 N)^*$	$\mathcal{O}(dr^2 N)$	$\mathcal{O}(rN)^*$	rN

Previous results:

- $\mathcal{O}(r^2 N^2)$ undetermined coefficients;
- $\mathcal{O}(r^r N)$ [Brent-Kung78];
- $\mathcal{O}(r^2 N)$ [van der Hoeven02].

Basic Idea: Newton Iteration has Good Complexity

To solve $\phi(y) = 0$, iterate

$$y_{n+1} = y_n - u_{n+1}, \quad \phi'(y_n)u_{n+1} = \phi(y_n).$$

Example 1: $\phi(y) = 3y^2 + y - 1$

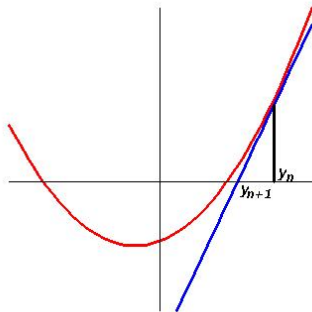
$$y_0 = 0.5$$

$$y_1 = 0.43750000000000000000000000000000$$

$$y_2 = 0.43426724137931034482758620$$

$$y_3 = 0.43425854597357627223447451$$

$$y_4 = 0.43425854591066488218983000$$



The cost is dominated by the last iteration.

Basic Idea: Newton Iteration has Good Complexity

To solve $\phi(y) = 0$, iterate

$$y_{n+1} = y_n - u_{n+1}, \quad \phi'(y_n)u_{n+1} = \phi(y_n).$$

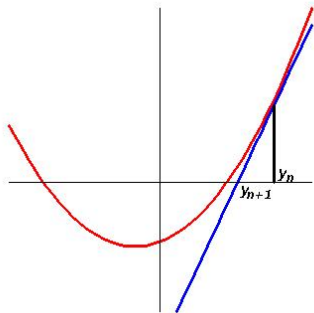
Example 2: $\phi(y) = ty^2 + y - 1$

$$y_0 = 0$$

$$y_1 = 1$$

$$y_2 = 1 - t + 2t^2 - 4t^3 + 8t^4 - 16t^5 + 32t^6 - 64t^7 + \dots$$

$$y_3 = 1 - t + 2t^2 - 5t^3 + 14t^4 - 42t^5 + 132t^6 - 428t^7 + \dots$$



The cost is dominated by the last iteration.

Newton for Power Series

$$y_{n+1} = y_n - u_{n+1}, \quad \phi'(y_n)u_{n+1} = \phi(y_n).$$

- Reciprocal [Sieveking72, Kung74]: $\phi(y) = a(t) - 1/y$;

$$y_{n+1} = y_n - u_{n+1}, \quad u_{n+1} = y_n \times (y_n \times a(t) - 1).$$

→ Same complexity $\mathcal{O}(M(N))$ as polynomial multiplication.

Newton for Power Series

$$y_{n+1} = y_n - u_{n+1}, \quad \phi'(y_n)u_{n+1} = \phi(y_n).$$

- Reciprocal [Sieveking72, Kung74]: $\phi(y) = a(t) - 1/y$;
- Logarithm [Brent75]: $\log a(t) = \int a'(t)/a(t)$;

→ Same complexity $\mathcal{O}(M(N))$ as polynomial multiplication.

Newton for Power Series

$$y_{n+1} = y_n - u_{n+1}, \quad \phi'(y_n)u_{n+1} = \phi(y_n).$$

- Reciprocal [Sieveking72, Kung74]: $\phi(y) = a(t) - 1/y$;
- Logarithm [Brent75]: $\log a(t) = \int a'(t)/a(t)$;
- Exponential [Brent75]: $\phi(y) = a(t) - \log y$.

$$y_{n+1} = y_n - u_{n+1}, \quad u_{n+1} = y_n \times (\log y_n - a(t)).$$

→ Same complexity $\mathcal{O}(M(N))$ as polynomial multiplication.

Newton for Power Series

$$y_{n+1} = y_n - u_{n+1}, \quad \phi'(y_n)u_{n+1} = \phi(y_n).$$

- Reciprocal [Sieveking72, Kung74]: $\phi(y) = a(t) - 1/y$;
- Logarithm [Brent75]: $\log a(t) = \int a'(t)/a(t)$;
- Exponential [Brent75]: $\phi(y) = a(t) - \log y$.
- 1st order LODE [Brent & Kung 78]: $y'(t) - a(t)y(t) = b(t)$
 - Homogeneous Case ($b = 0$): $y(t) = y_0(t) := \int \exp(a(t))$;
 - Non-homogeneous Case: variation of constants,
 $y(t) = y_0(t) \int b(t)/y_0(t)$.

→ Same complexity $\mathcal{O}(M(N))$ as polynomial multiplication.

Newton for Power Series

$$y_{n+1} = y_n - u_{n+1}, \quad \phi'(y_n)u_{n+1} = \phi(y_n).$$

- Reciprocal [Sieveking72, Kung74]: $\phi(y) = a(t) - 1/y$;
- Logarithm [Brent75]: $\log a(t) = \int a'(t)/a(t)$;
- Exponential [Brent75]: $\phi(y) = a(t) - \log y$.
- 1st order LODE [Brent & Kung 78]: $y'(t) - a(t)y(t) = b(t)$
 - Homogeneous Case ($b = 0$): $y(t) = y_0(t) := \int \exp(a(t))$;
 - Non-homogeneous Case: variation of constants,
 $y(t) = y_0(t) \int b(t)/y_0(t)$.

Natural idea: LDE of order r equivalent to 1st order **matrix** equation:

$$Y'(t) - A(t)Y(t) = B(t),$$

but $\int \exp A(t)$ is **not** a solution of $Y'(t) = A(t)Y(t)$.

Newton Iteration for Operators

Newton iteration for $\phi(Y) = 0$:

$$Y_{k+1} = Y_k - U_{k+1}, \quad \underbrace{D\phi|_{Y_k}}_{\text{differential}} \cdot U_{k+1} = \phi(Y_k).$$

Special cases:



Newton Iteration for Operators

Newton iteration for $\phi(Y) = 0$:

$$Y_{k+1} = Y_k - \underbrace{U_{k+1}}_{\text{differential}}, \quad \underbrace{D\phi|_{Y_k}}_{\text{differential}} \cdot U_{k+1} = \phi(Y_k).$$



Special cases:

- Matrix inversion [Schulz33]: $\phi(Y) = AY - I$, $D\phi \cdot U = AU$.

→ equation $AU_{k+1} = AY_k - I$; $U_{k+1} = Y_k(AY_k - I)$.

Complexity: $MM(r, N) = \mathcal{O}(r^2M(N) + r^\omega N)$.

Newton Iteration for Operators



Newton iteration for $\phi(Y) = 0$:

$$Y_{k+1} = Y_k - \underbrace{U_{k+1}}_{\text{differential}}, \quad \underbrace{D\phi|_{Y_k}}_{\text{differential}} \cdot U_{k+1} = \phi(Y_k).$$

Special cases:

- Matrix inversion [Schulz33]: $\phi(Y) = AY - I$, $D\phi \cdot U = AU$.
→ equation $AU_{k+1} = AY_k - I$; $U_{k+1} = Y_k(AY_k - I)$.
Complexity: $\text{MM}(r, N) = \mathcal{O}(r^2M(N) + r^\omega N)$.
- **Our problem:** $\phi(Y) = Y' - AY$. Note that $D\phi = \phi!$

New Algorithm and its Complexity

To iterate $Y_{k+1} = Y_k - U_{k+1}$, we need to solve

$$U'_{k+1} - A(t)U_{k+1} = Y'_k - A(t)Y_k.$$

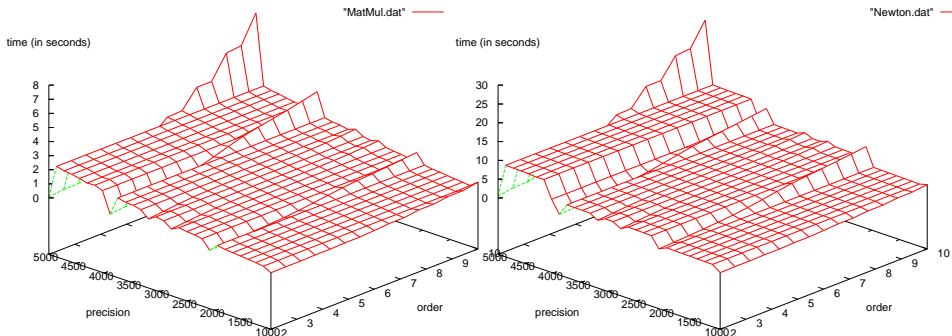
Variation of constants:

$$U_{k+1} = Y_k Z_{k+1},$$
$$Y_k Z'_{k+1} + \underbrace{Y'_k Z_{k+1} - A(t)Y_k Z_{k+1}}_0 = Y'_k - A(t)Y_k,$$

$$Y_{k+1} = Y_k - Y_k \int Y_k^{-1} (Y'_k - A(t)Y_k) \text{ mod } t^{2^{k+1}}, \quad \det Y_0 \neq 0$$

- Quadratic convergence;
- a whole basis of solutions is computed;
- its inverse is computed simultaneously;
- complexity $\mathcal{O}(r^2 M(N) + r^\omega N)$;
- good for $r = 1$ ([Hanrot-Zimmermann02] for $\exp(a(t))$).

Experimental results



Polynomial matrix multiplication vs. solving $Y' = AY$ by our method.

Non-Linear Differential Equations by Linearization

Newton again: $y_{k+1} = y_k - u_{k+1}$, $\underbrace{D\phi|_{y_k}}_{\text{differential}} \cdot u_{k+1} = \phi(y_k)$.

A **1st order** example from number theory (related to \wp):

$$\phi(y) = (x^3 + ax + b)y'^2 - y^3 - Ay - B = 0.$$

Differential:

$$\phi(y_k + u_{k+1}) - \phi(y_k) = \underbrace{(x^3 + ax + b)2y'_k u'_{k+1} - (3y_k^2 - A)u_{k+1}}_{D\phi|_{y_k} \cdot u_{k+1}} + \dots$$

→ **Linear** differential equation with **power series** coefficients

→ fast algorithm for isogenies between elliptic curves [BoMoSaSc06].

Higher order → a linear differential system at each step.

Have a look at the paper for

- constant coefficients
(including nice and fast algorithm for $\exp(tA)$);
- a divide-and-conquer algorithm for only one solution (instead of a basis) in the general case.

Still to be saved:

- a factor r for (1 equation, a basis of solutions);
- a factor $\log N$ for only one solution.