Analytic Algorithms for Linear ODEs

Marc Mezzarobba

CNRS, École polytechnique

DART 2026, Île d'Oléron

Bridging Numeric and Symbolic Solvers

Setting.

$$a_r(x)\, y^{(r)}(x) + \cdots + a_1(x)\, y'(x) + a_0(x)\, y(x) = 0$$

Bridging Numeric and Symbolic Solvers

Setting.

$$a_r(x)\, y^{(r)}(x) + \cdots + a_1(x)\, y'(x) + a_0(x)\, y(x) = 0$$

$$y \colon U \subseteq \mathbb{C} \to \mathbb{C}$$

Bridging Numeric and Symbolic Solvers

Setting.

$$\begin{array}{ccccc} a_i \in \bar{\mathbb{Q}}[x] & & a_i \in \bar{\mathbb{Q}}[x] & & a_i \in \bar{\mathbb{Q}}[x] \\ \downarrow & & \downarrow & & \downarrow \\ a_r(x)\, y^{(r)}(x) + \cdots + & & a_1(x)\, y'(x) + & & a_0(x)\, y(x) = 0 \\ & & & & \uparrow \\ & & & & y \colon U \subseteq \mathbb{C} \to \mathbb{C} \end{array}$$

Bridging Numeric and Symbolic Solvers

Setting.

$$\begin{array}{ccccc} a_i \in \bar{\mathbb{Q}}[x] & & a_i \in \bar{\mathbb{Q}}[x] & & a_i \in \bar{\mathbb{Q}}[x] \\ \downarrow & & \downarrow & & \downarrow \\ a_r(x)\, y^{(r)}(x) + \cdots + & & a_1(x)\, y'(x) + & & a_0(x)\, y(x) = 0 \\ & & & & \uparrow \\ & & & & y \colon U \subseteq \mathbb{C} \to \mathbb{C} \end{array}$$

Question.

Suppose that we know how to solve this equation **numerically**.

Can we use that to recover **exact** solutions (of a certain type)?

Bridging Numeric and Symbolic Solvers

Setting.

$$\begin{array}{ccccc} a_i \in \bar{\mathbb{Q}}[x] & & a_i \in \bar{\mathbb{Q}}[x] & & a_i \in \bar{\mathbb{Q}}[x] \\ \downarrow & & \downarrow & & \downarrow \\ a_r(x)\, y^{(r)}(x) + \cdots + & & a_1(x)\, y'(x) + & & a_0(x)\, y(x) = 0 \\ & & & & \uparrow \\ & & & & y \colon U \subseteq \mathbb{C} \to \mathbb{C} \end{array}$$

Question.

Suppose that we know how to solve this equation **numerically**.

Can we use that to recover **exact** solutions (of a certain type)?

rule out

Bridging Numeric and Symbolic Solvers

Setting.

$$\begin{array}{ccccc} a_i \in \bar{\mathbb{Q}}[x] & & a_i \in \bar{\mathbb{Q}}[x] & & a_i \in \bar{\mathbb{Q}}[x] \\ \downarrow & & \downarrow & & \downarrow \\ a_r(x)\, y^{(r)}(x) + \cdots + & & a_1(x)\, y'(x) + & & a_0(x)\, y(x) = 0 \\ & & & & \uparrow \\ & & & & y\colon U \subseteq \mathbb{C} \to \mathbb{C} \end{array}$$

Question.

Suppose that we know how to solve this equation **numerically**.

Can we use that to recover **exact** solutions (of a certain type)?

rule out

This talk: numerical solver suitable for this
examples of its use to solve exact problems

General Idea

- Algebraic algorithms for "solving" linear differential equations extract partial information on the differential Galois group

Examples:
- algebraic solutions
- Liouvillian solutions
- operator factorization
- LCLM decomposition

[Kovacic 1977(?), 1986;
Singer 1979, 1981, 1996;
Ulmer 1991, 2003;
Singer–Ulmer 1995, 1998;
van Hoeij 1996, 1997;
van Hoeij–Ragot–Ulmer–Weil 1999;
Fredet 2003;
Compoint–Weil 2004; …]

General Idea

- Algebraic algorithms for "solving" linear differential equations extract partial information on the differential Galois group

  Examples:
  - algebraic solutions
  - Liouvillian solutions
  - operator factorization
  - LCLM decomposition

  [Kovacic 1977(?), 1986; Singer 1979, 1981, 1996; Ulmer 1991, 2003; Singer–Ulmer 1995, 1998; van Hoeij 1996, 1997; van Hoeij–Ragot–Ulmer–Weil 1999; Fredet 2003; Compoint–Weil 2004; …]

- Viewing solutions as analytic functions gives access to global information

  This means computing

  → Connection matrices between singular points
  → Monodromy matrices with a common base point
  → Stokes matrices

  } transcendental

Analytic Methods for the Solution of LODEs

19th century: Fuchsian equations

Jordan 1878 — finite monodromy

Picard 1883, 1887 — Galois invariance

Schlesinger 1895 — density of the monodromy group

Frobenius, Klein, Lie, Fuchs, …

20th century: Irregular singularities

Écalle 1981, 1985 — accelero-summation

Ramis 1985 — monodromy + Stokes

…

Algorithms

Kovacic 1977(?); Singer 1979; … — (algebraic) algorithms from Picard–Vessiot theory

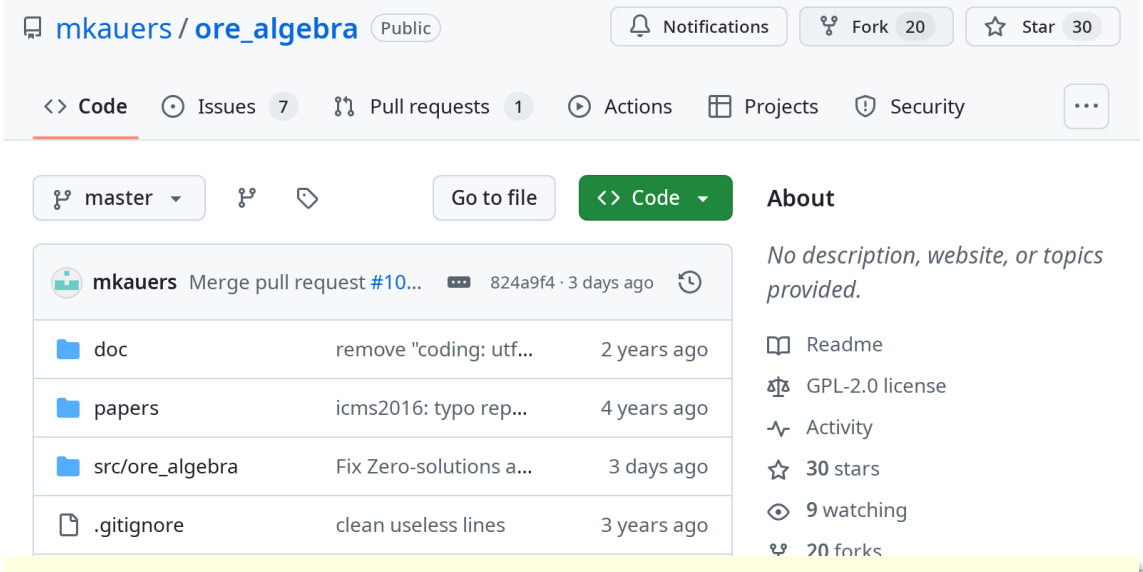
van der Hoeven 2007, 2022 — symbolic-numeric factorization and heuristic Galois

Llorente–Mozo-Fernández 2016 — Liouvillian solutions

Johansson-Kauers-M. 2013, Chyzak–Goyer–M. 2022, Goyer 2025 — special cases of factorization

A Rigourous Arbitrary-Precision Numerical Solver

The ore_algebra Package




```
$ sage -pip install --no-build-isolation git+https://github.com/mkauers/ore_algebra.git
```

Differential Operators in ore_algebra

$$a_r(x)\,y^{(r)}(x) + \cdots + a_1(x)\,y'(x) + a_0(x)\,y(x) = 0$$

$$\uparrow$$

$$y : U \subseteq \mathbb{C} \to \mathbb{C}$$

```
SageMath version 10.7, Release Date: 2025-08-09
sage: from ore_algebra import OreAlgebra
sage: POL.<x> = QQ[]
sage: DOP.<Dx> = OreAlgebra(POL)
sage: DOP
sage: Dx*x
sage: (x^2 + 1)*Dx^2 + Dx + 1
sage:
```

Differential Operators in ore_algebra

$$a_r(x)\,y^{(r)}(x) + \cdots + a_1(x)\,y'(x) + a_0(x)\,y(x) = 0$$

$$\uparrow$$

$$y : U \subseteq \mathbb{C} \to \mathbb{C}$$

```
SageMath version 10.7, Release Date: 2025-08-09
sage: from ore_algebra import OreAlgebra
sage: POL.<x> = QQ[]
sage: DOP.<Dx> = OreAlgebra(POL)
sage: DOP
Univariate Ore algebra in Dx over Univariate Polynomial Ring in x over Rational Fi
eld
sage: Dx*x
sage: (x^2 + 1)*Dx^2 + Dx + 1
sage:
```

Differential Operators in ore_algebra

$$a_r(x)\,y^{(r)}(x) + \cdots + a_1(x)\,y'(x) + a_0(x)\,y(x) = 0$$

$$y : U \subseteq \mathbb{C} \to \mathbb{C}$$

```
SageMath version 10.7, Release Date: 2025-08-09
sage: from ore_algebra import OreAlgebra
sage: POL.<x> = QQ[]
sage: DOP.<Dx> = OreAlgebra(POL)
sage: DOP
Univariate Ore algebra in Dx over Univariate Polynomial Ring in x over Rational Fi
eld
sage: Dx*x
x*Dx + 1
sage: (x^2 + 1)*Dx^2 + Dx + 1
sage:
```

Differential Operators in ore_algebra

$$a_r(x)\,y^{(r)}(x) + \cdots + a_1(x)\,y'(x) + a_0(x)\,y(x) = 0$$

$$\uparrow$$

$$y : U \subseteq \mathbb{C} \to \mathbb{C}$$

```
SageMath version 10.7, Release Date: 2025-08-09
sage: from ore_algebra import OreAlgebra
sage: POL.<x> = QQ[]
sage: DOP.<Dx> = OreAlgebra(POL)
sage: DOP
Univariate Ore algebra in Dx over Univariate Polynomial Ring in x over Rational Fi
eld
sage: Dx*x
x*Dx + 1
sage: (x^2 + 1)*Dx^2 + Dx + 1
(x^2 + 1)*Dx^2 + Dx + 1
sage:
```

Analytic Solutions

Theorem. [Cauchy, ~1839]

Assume $a_r(x) \neq 0$ for $|x| < \rho$.

For any choice of $y(0), \ldots, y^{(r-1)}(0) \in \mathbb{C}$, the differential equation

$$a_r(x)\, y^{(r)}(x) + a_{r-1}(x)\, y^{(r-1)}(x) + \cdots + a_0(x)\, y(x) = 0$$

has a unique analytic solution $y\colon \{|x| < \rho\} \to \mathbb{C}$.

Analytic Solutions

Theorem. [Cauchy, ~1839]

Assume $a_r(x) \neq 0$ for $|x| < \rho$.

For any choice of $y(0), \ldots, y^{(r-1)}(0) \in \mathbb{C}$, the differential equation

$$a_r(x)\, y^{(r)}(x) + a_{r-1}(x)\, y^{(r-1)}(x) + \cdots + a_0(x)\, y(x) = 0$$

has a unique analytic solution $y\colon \{|x| < \rho\} \to \mathbb{C}$.

"Local" basis of solutions:

$$\begin{cases} f_0(x) &= 1 + 0\,x + \cdots + 0\,x^{r-1} + \blacksquare\, x^r + \cdots \\ f_1(x) &= 0 + 1\,x + \cdots + 0\,x^{r-1} + \blacksquare\, x^r + \cdots \\ &\vdots \\ f_{r-1}(x) &= 0 + 0\,x + \cdots + 1\,x^{r-1} + \blacksquare\, x^r + \cdots \end{cases}$$

Analytic Solutions

Theorem. [Cauchy, ~1839]

Assume $a_r(x) \neq 0$ for $|x| < \rho$.

For any choice of $y(0), \ldots, y^{(r-1)}(0) \in \mathbb{C}$, the differential equation

$$a_r(x)\, y^{(r)}(x) + a_{r-1}(x)\, y^{(r-1)}(x) + \cdots + a_0(x)\, y(x) = 0$$

has a unique analytic solution $y\colon \{|x| < \rho\} \to \mathbb{C}$.

"Local" basis of solutions:

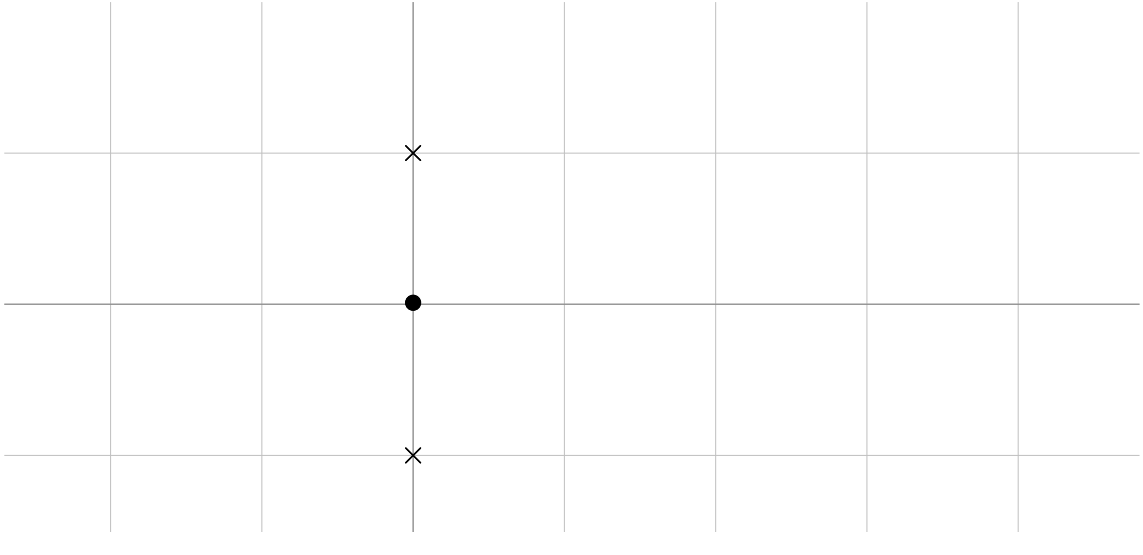
$$\begin{cases} f_0(x) &= 1 + 0\,x + \cdots + 0\,x^{r-1} + \blacksquare\, x^r + \cdots \\ f_1(x) &= 0 + 1\,x + \cdots + 0\,x^{r-1} + \blacksquare\, x^r + \cdots \\ &\vdots \\ f_{r-1}(x) &= 0 + 0\,x + \cdots + 1\,x^{r-1} + \blacksquare\, x^r + \cdots \end{cases}$$

Fundamental matrix:

$$F(x) = \begin{pmatrix} f_0(x) & \cdots & f_{r-1}(x) \\ f_0'(x) & & f_{r-1}'(x) \\ \vdots & & \vdots \\ \dfrac{f_0^{(r-1)}(x)}{(r-1)!} & \cdots & \dfrac{f_{r-1}^{(r-1)}(x)}{(r-1)!} \end{pmatrix}$$

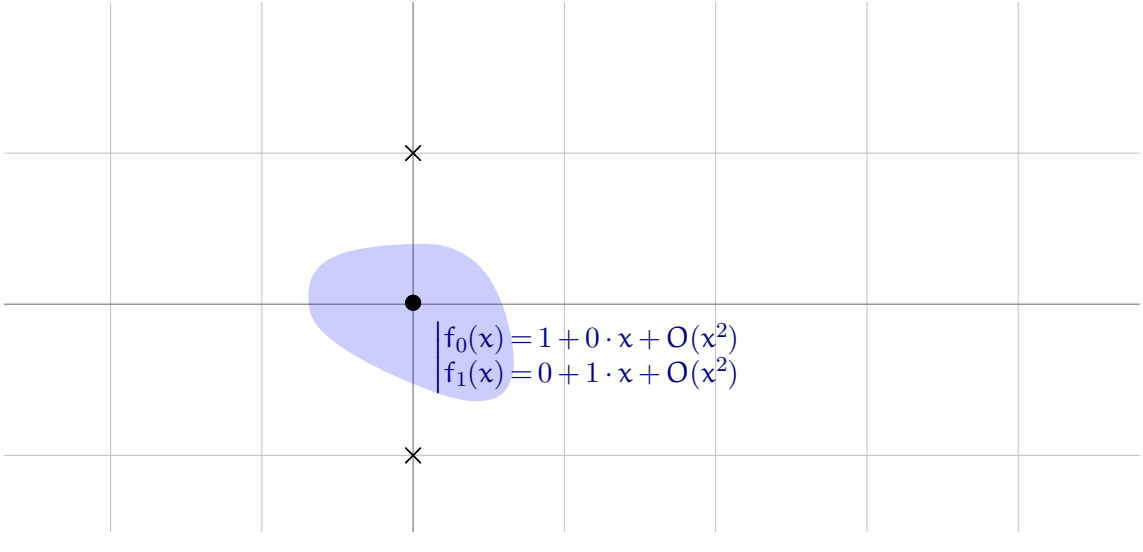
Analytic Continuation of Solutions

$$(x^2+1)\,y''(x)+2\,x\,y'(x)=0$$



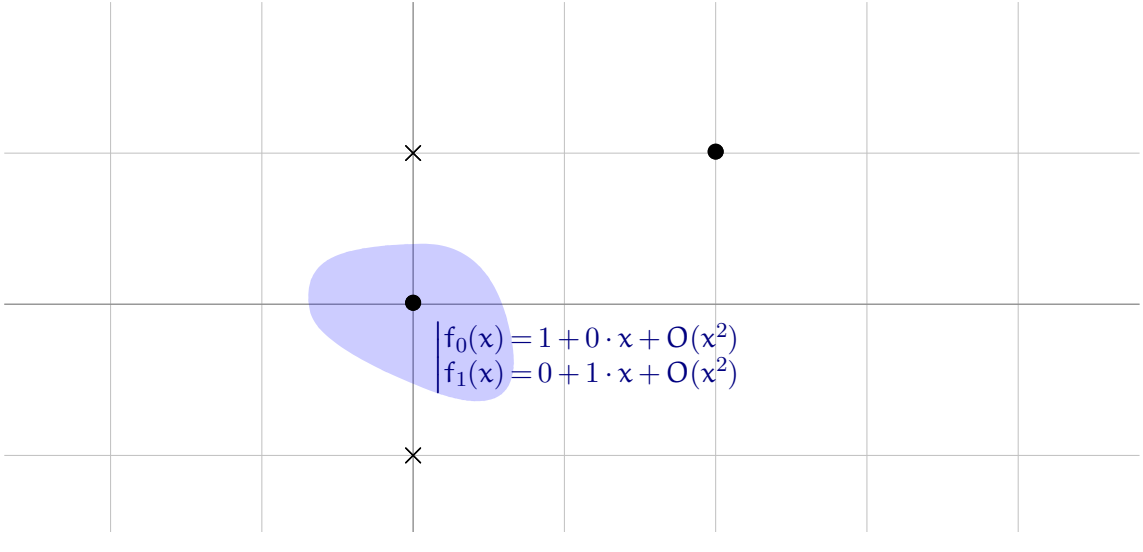
Analytic Continuation of Solutions

$$(x^2+1)\,y''(x)+2\,x\,y'(x)=0$$



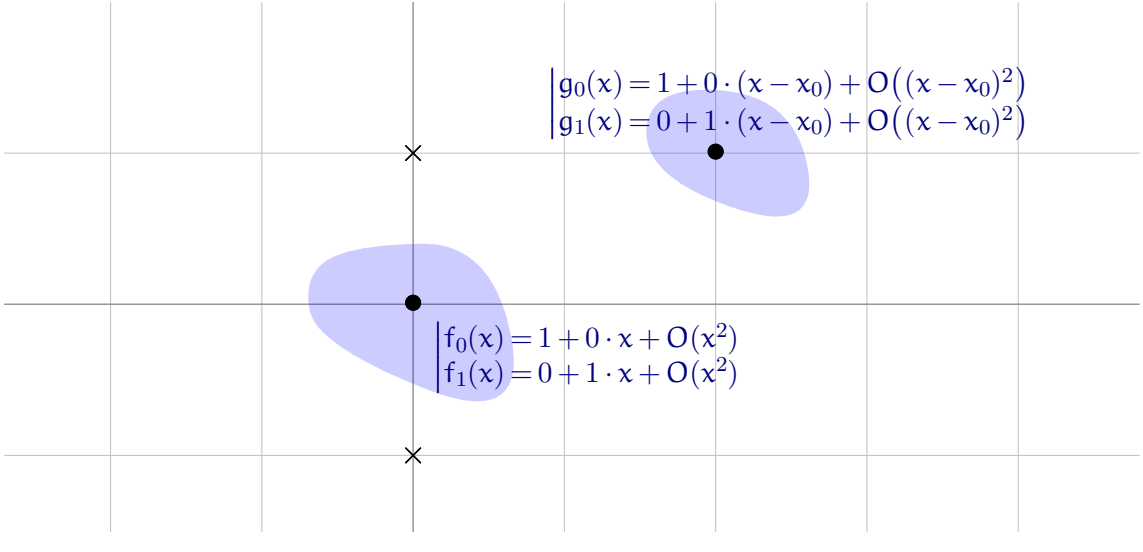
Analytic Continuation of Solutions

$$(x^2+1)\,y''(x)+2\,x\,y'(x)=0$$



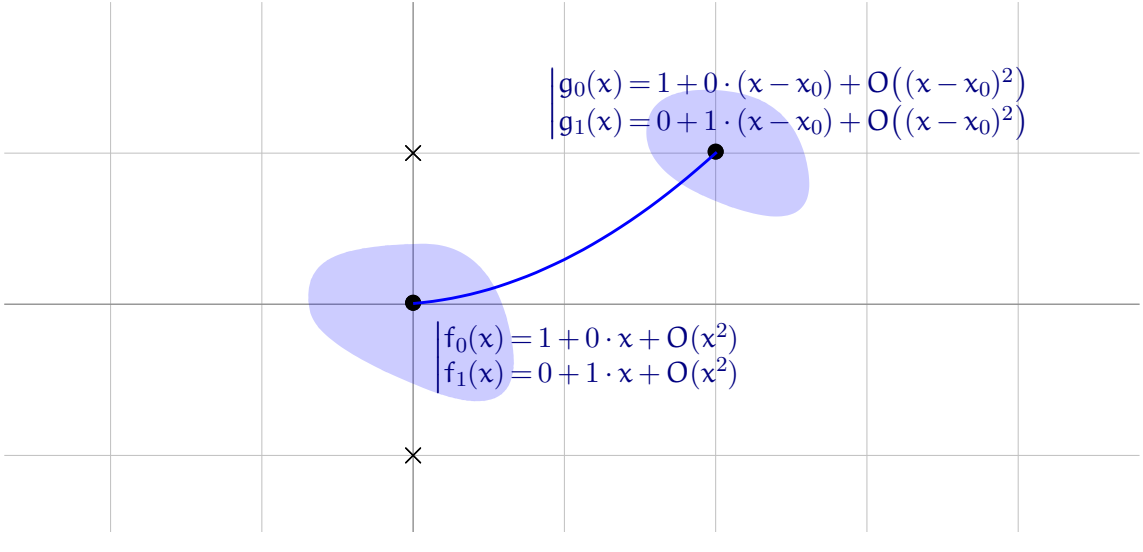
Analytic Continuation of Solutions

$$(x^2+1)\,y''(x)+2\,x\,y'(x)=0$$



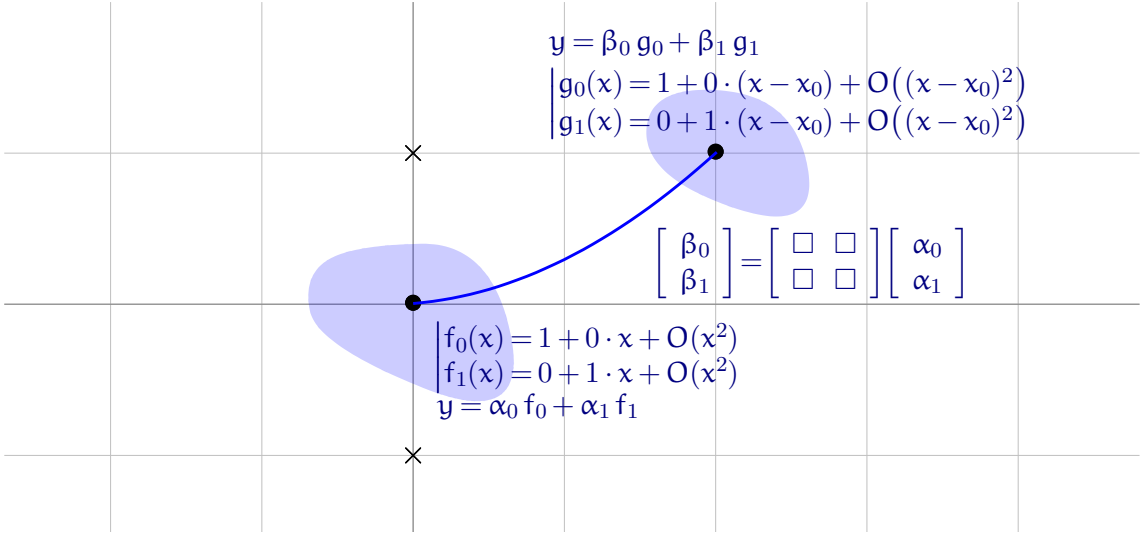
Analytic Continuation of Solutions

$$(x^2+1)\,y''(x)+2\,x\,y'(x)=0$$



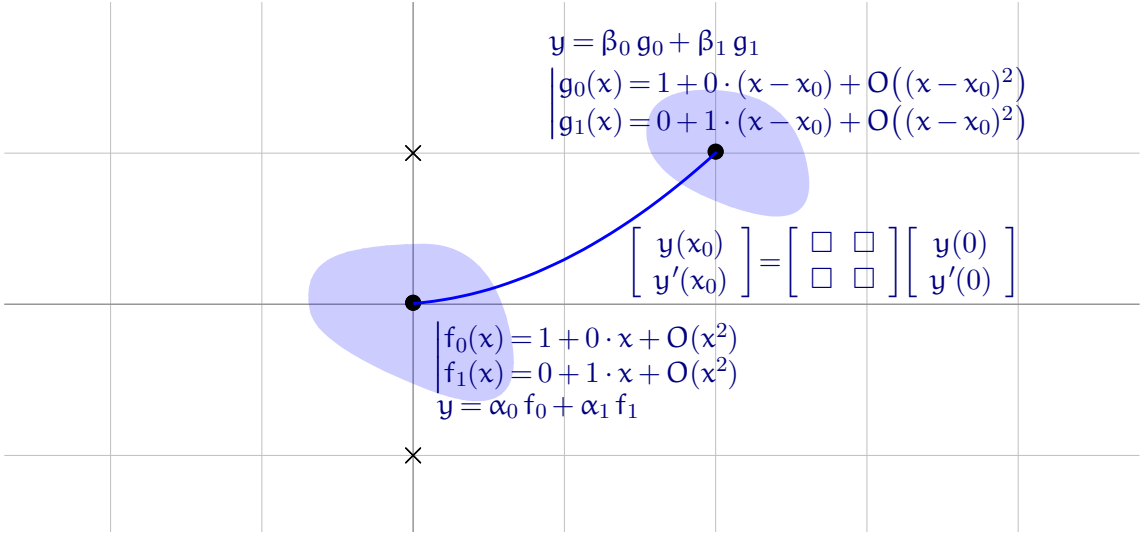
Analytic Continuation of Solutions

$$(x^2+1)\,y''(x)+2\,x\,y'(x)=0$$



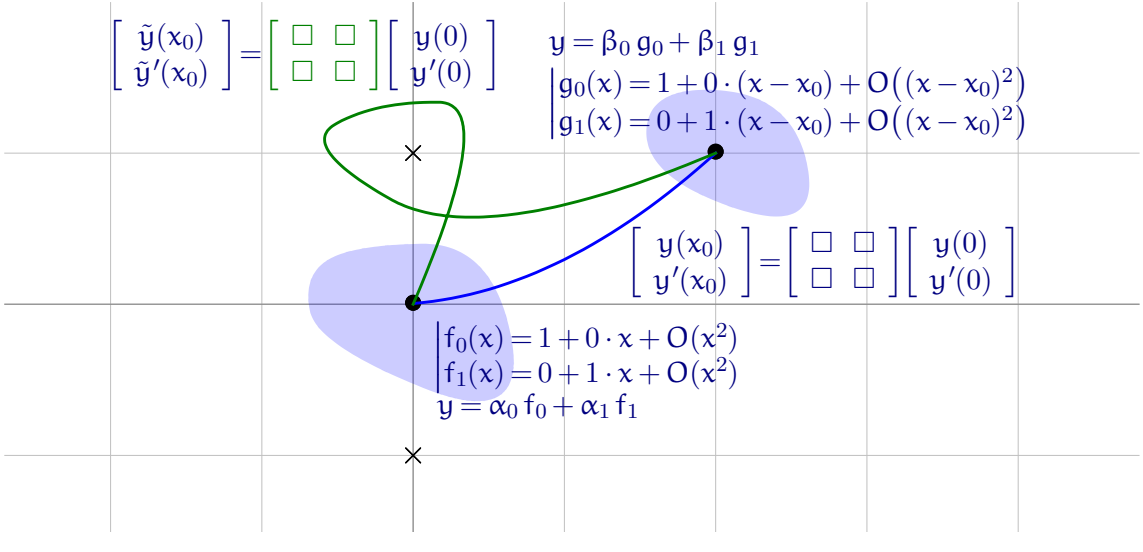
Analytic Continuation of Solutions

$$(x^2+1)\,y''(x)+2\,x\,y'(x)=0$$



Analytic Continuation of Solutions

$$(x^2+1)\,y''(x)+2\,x\,y'(x)=0$$



Analytic Continuation of Solutions


```
sage: (Dx - 1).numerical_solution(ini=[1], path=[0,1])
sage: dop = (x^2 + 1)*Dx^2 + 2*x*Dx
sage: dop.numerical_transition_matrix([0, 1])
sage: dop.numerical_solution(ini=[0,1], path=[0,1])
sage: dop.numerical_solution(ini=[0,1], path=[0,2+I])
sage: dop.numerical_solution(ini=[0,1], path=[0,1+I,2*I,I-1,0,2+I])
sage:
```

Automatically bound truncation errors [van der Hoeven 1999; M. 2018]

Interval arithmetic for error propagation and rounding errors

Analytic Continuation of Solutions


```
sage: (Dx - 1).numerical_solution(ini=[1], path=[0,1])
[2.7182818284590452 +/- 3.57e-17]
sage: dop = (x^2 + 1)*Dx^2 + 2*x*Dx
sage: dop.numerical_transition_matrix([0, 1])
sage: dop.numerical_solution(ini=[0,1], path=[0,1])
sage: dop.numerical_solution(ini=[0,1], path=[0,2+I])
sage: dop.numerical_solution(ini=[0,1], path=[0,1+I,2*I,I-1,0,2+I])
sage:
```

Automatically bound truncation errors [van der Hoeven 1999; M. 2018]

Interval arithmetic for error propagation and rounding errors

Analytic Continuation of Solutions


```
sage: (Dx - 1).numerical_solution(ini=[1], path=[0,1])
[2.7182818284590452 +/- 3.57e-17]
sage: dop = (x^2 + 1)*Dx^2 + 2*x*Dx
sage: dop.numerical_transition_matrix([0, 1])
[    [1.0000000000000000 +/- 2e-21] [0.78539816339744831 +/- 3.89e-19]]
[                    [+/- 1.02e-21]  [0.50000000000000000 +/- 2.9e-21]]
sage: dop.numerical_solution(ini=[0,1], path=[0,1])
sage: dop.numerical_solution(ini=[0,1], path=[0,2+I])
sage: dop.numerical_solution(ini=[0,1], path=[0,1+I,2*I,I-1,0,2+I])
sage:
```

Automatically bound truncation errors [van der Hoeven 1999; M. 2018]

Interval arithmetic for error propagation and rounding errors

Analytic Continuation of Solutions


```
sage: (Dx - 1).numerical_solution(ini=[1], path=[0,1])
[2.7182818284590452 +/- 3.57e-17]
sage: dop = (x^2 + 1)*Dx^2 + 2*x*Dx
sage: dop.numerical_transition_matrix([0, 1])
[    [1.0000000000000000 +/- 2e-21] [0.78539816339744831 +/- 3.89e-19]]
[                    [+/- 1.02e-21]  [0.50000000000000000 +/- 2.9e-21]]
sage: dop.numerical_solution(ini=[0,1], path=[0,1])
[0.78539816339744831 +/- 5.40e-19]
sage: dop.numerical_solution(ini=[0,1], path=[0,2+I])
sage: dop.numerical_solution(ini=[0,1], path=[0,1+I,2*I,I-1,0,2+I])
sage:
```

Automatically bound truncation errors [van der Hoeven 1999; M. 2018]

Interval arithmetic for error propagation and rounding errors

Analytic Continuation of Solutions


```
sage: (Dx - 1).numerical_solution(ini=[1], path=[0,1])
[2.7182818284590452 +/- 3.57e-17]
sage: dop = (x^2 + 1)*Dx^2 + 2*x*Dx
sage: dop.numerical_transition_matrix([0, 1])
[    [1.0000000000000000 +/- 2e-21] [0.78539816339744831 +/- 3.89e-19]]
[                    [+/- 1.02e-21]  [0.50000000000000000 +/- 2.9e-21]]
sage: dop.numerical_solution(ini=[0,1], path=[0,1])
[0.78539816339744831 +/- 5.40e-19]
sage: dop.numerical_solution(ini=[0,1], path=[0,2+I])
[1.1780972450961725 +/- 3.58e-17] + [0.17328679513998633 +/- 2.68e-18]*I
sage: dop.numerical_solution(ini=[0,1], path=[0,1+I,2*I,I-1,0,2+I])
sage:
```

Automatically bound truncation errors [van der Hoeven 1999; M. 2018]

Interval arithmetic for error propagation and rounding errors

Analytic Continuation of Solutions


```
sage: (Dx - 1).numerical_solution(ini=[1], path=[0,1])
[2.7182818284590452 +/- 3.57e-17]
sage: dop = (x^2 + 1)*Dx^2 + 2*x*Dx
sage: dop.numerical_transition_matrix([0, 1])
[    [1.0000000000000000 +/- 2e-21] [0.78539816339744831 +/- 3.89e-19]]
[                    [+/- 1.02e-21]  [0.50000000000000000 +/- 2.9e-21]]
sage: dop.numerical_solution(ini=[0,1], path=[0,1])
[0.78539816339744831 +/- 5.40e-19]
sage: dop.numerical_solution(ini=[0,1], path=[0,2+I])
[1.1780972450961725 +/- 3.58e-17] + [0.17328679513998633 +/- 2.68e-18]*I
sage: dop.numerical_solution(ini=[0,1], path=[0,1+I,2*I,I-1,0,2+I])
[4.3196898986859657 +/- 3.33e-18] + [0.17328679513998633 +/- 3.00e-18]*I
sage:
```

Automatically bound truncation errors [van der Hoeven 1999; M. 2018]

Interval arithmetic for error propagation and rounding errors

High Precision


```
sage: (Dx - 1).numerical_solution(ini=[1], path=[0,1], eps=1e-1000)
sage:
```

'Standard' numerical methods like RK4 need $\Omega(2^n)$ steps for n digits.

High precision requires order-adaptive methods, large steps.

High Precision

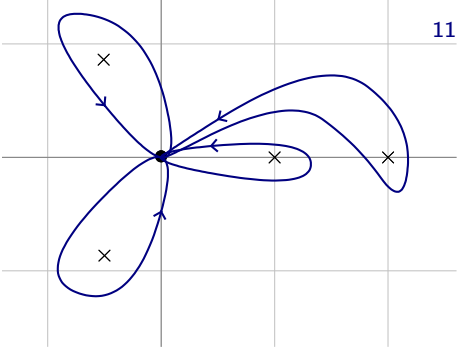

```
sage: (Dx - 1).numerical_solution(ini=[1], path=[0,1], eps=1e-1000)
[2.71828182845904523536028747135266249775724709369995957496696762772407663035354759457138217852516642742746639193200305992181741359662904357290033429526059563073813232862794349076323382988075319525101901157383418793070215408914993488416750924476146066808226480016847741185374234544243710753907774499206955170276183860626133138458300075204493382656029760673711320070932870912744374704723069697720931014169283681902551510865746377211125238978442505695369677078544996996794686445490598793163688923009879312773617821542499922957635148220826989519366803318252886939849646510582093923982948879332036250944311730123819706841614039701983767932068328237646480429531180232878250981945581530175671736133206981125099618188159304169035159888851934580727386673858942287922849989208680582574927961048419844436346324496848756023362482704197862320900216099023530436994184914631409343173814364054625315209618369088870701676839642437814059271456354906130310720851038375051011574770417189861068739696552126715468895703503540 +/- 5.53e-1002]
sage:
```

'Standard' numerical methods like RK4 need $\Omega(2^n)$ steps for n digits.

High precision requires order-adaptive methods, large steps.

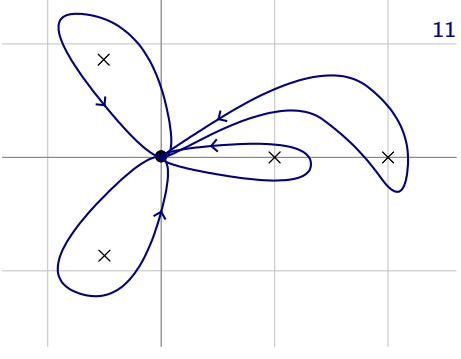
Monodromy Matrices

```
sage: from ore_algebra.analytic.monodromy \
          import monodromy_matrices
sage: dop = (x^3-1)*(x-2)*Dx^2 + x*Dx + 5
sage: mon = monodromy_matrices(dop, 0)
sage: len(mon)
sage: mon[0].change_ring(ComplexField(40))
sage: mon[0][0,0]
sage: mon[1]
sage:
```

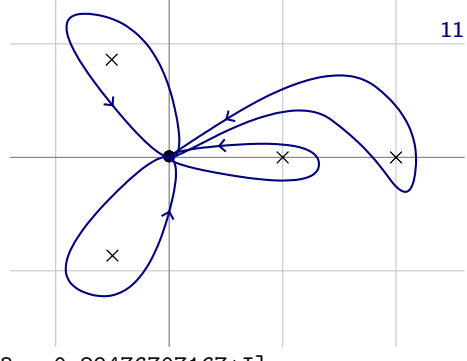


Monodromy Matrices

```
sage: from ore_algebra.analytic.monodromy \
          import monodromy_matrices
sage: dop = (x^3-1)*(x-2)*Dx^2 + x*Dx + 5
sage: mon = monodromy_matrices(dop, 0)
sage: len(mon)
4
sage: mon[0].change_ring(ComplexField(40))
sage: mon[0][0,0]
sage: mon[1]
sage:
```

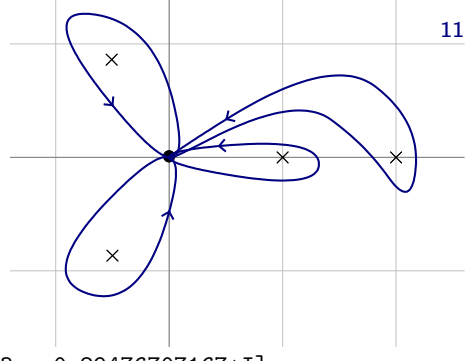


Monodromy Matrices



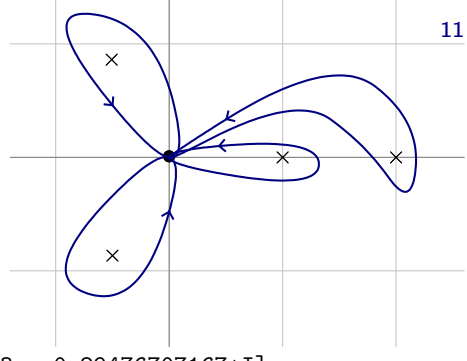
```
sage: from ore_algebra.analytic.monodromy \
          import monodromy_matrices
sage: dop = (x^3-1)*(x-2)*Dx^2 + x*Dx + 5
sage: mon = monodromy_matrices(dop, 0)
sage: len(mon)
4
sage: mon[0].change_ring(ComplexField(40))
[ -0.55564547505 + 0.89815233379*I   0.51055154453 - 0.29476707167*I]
[-0.16955120869 + 0.097890435978*I   1.0556454751 - 0.032126930001*I]
sage: mon[0][0,0]
sage: mon[1]
sage:
```

Monodromy Matrices



```
sage: from ore_algebra.analytic.monodromy \
          import monodromy_matrices
sage: dop = (x^3-1)*(x-2)*Dx^2 + x*Dx + 5
sage: mon = monodromy_matrices(dop, 0)
sage: len(mon)
4
sage: mon[0].change_ring(ComplexField(40))
[ -0.55564547505 + 0.89815233379*I   0.51055154453 - 0.29476707167*I]
[-0.16955120869 + 0.097890435978*I   1.0556454751 - 0.032126930001*I]
sage: mon[0][0,0]
[-0.555645475053369 +/- 2.73e-16] + [0.898152333785686 +/- 1.82e-16]*I
sage: mon[1]
sage:
```

Monodromy Matrices



```
sage: from ore_algebra.analytic.monodromy \
          import monodromy_matrices
sage: dop = (x^3-1)*(x-2)*Dx^2 + x*Dx + 5
sage: mon = monodromy_matrices(dop, 0)
sage: len(mon)
4
sage: mon[0].change_ring(ComplexField(40))
[ -0.55564547505 + 0.89815233379*I   0.51055154453 - 0.29476707167*I]
[-0.16955120869 + 0.097890435978*I   1.0556454751 - 0.032126930001*I]
sage: mon[0][0,0]
[-0.555645475053369 +/- 2.73e-16] + [0.898152333785686 +/- 1.82e-16]*I
sage: mon[1]
[ [119.00208196561 +/- 1.54e-12] + [199.08061912444 +/- 1.31e-12]*I   [-
138.556085
17603 +/- 1.28e-12] + [84.30928189891 +/- 1.27e-12]*I]
[[-287.40309314516 +/- 5.56e-12] + [166.80270946776 +/- 1.70e-12]*I [-
118.22460289
957 +/- 3.82e-12] + [-200.05554703662 +/- 2.18e-12]*I]
sage:
```

Regular Singular Points

$$L = a_r(x)\, D^r + \cdots + a_0(x)$$

Recall: if $a_r(\xi) \neq 0$, full basis of **analytic** solutions around ξ

Regular Singular Points

$$L = a_r(x)\, D^r + \cdots + a_0(x)$$

Recall: if $a_r(\xi) \neq 0$, full basis of **analytic** solutions around ξ

Definition. A (finite) **regular singular point** of L is a point $\xi \in \mathbb{C}$ where $a_r(\xi) = 0$ but there is a full basis of solutions of the form

$$(x-\xi)^{\lambda}\big(f_0(x) + f_1(x)\log(x-\xi) + \cdots + f_K(x)\log(x-\xi)^K\big).$$

$\lambda \in \mathbb{C}$; $K \in \mathbb{N}$; $f_0, f_1, \ldots, f_K$: convergent power series

Again, we **fix** such a local basis at any regular singular ξ.

Regular Singular Points

$$L = a_r(x)\, D^r + \cdots + a_0(x)$$

Recall: if $a_r(\xi) \neq 0$, full basis of **analytic** solutions around ξ

Definition. A (finite) **regular singular point** of L is a point $\xi \in \mathbb{C}$ where $a_r(\xi) = 0$ but there is a full basis of solutions of the form

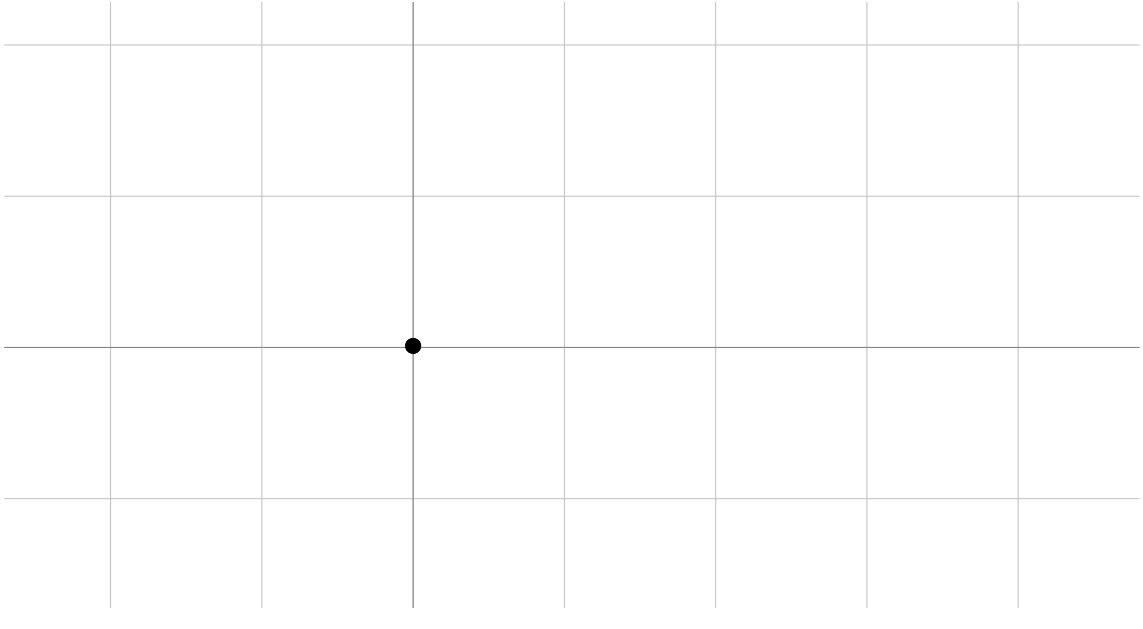
$$(x-\xi)^{\lambda}\big(f_0(x) + f_1(x)\log(x-\xi) + \cdots + f_K(x)\log(x-\xi)^K\big).$$

$\lambda \in \mathbb{C}$; $K \in \mathbb{N}$; $f_0, f_1, \ldots, f_K$: convergent power series

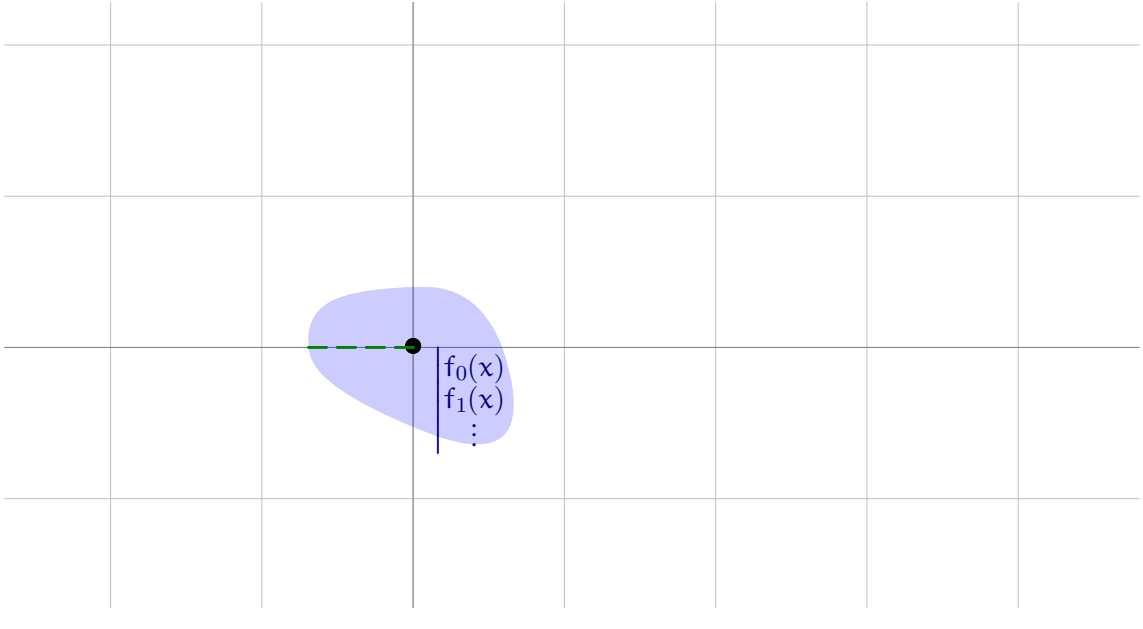
Again, we **fix** such a local basis at any regular singular ξ.

An equation is **Fuchsian** if all its singular points in $\mathbb{P}^1(\mathbb{C})$ are regular.

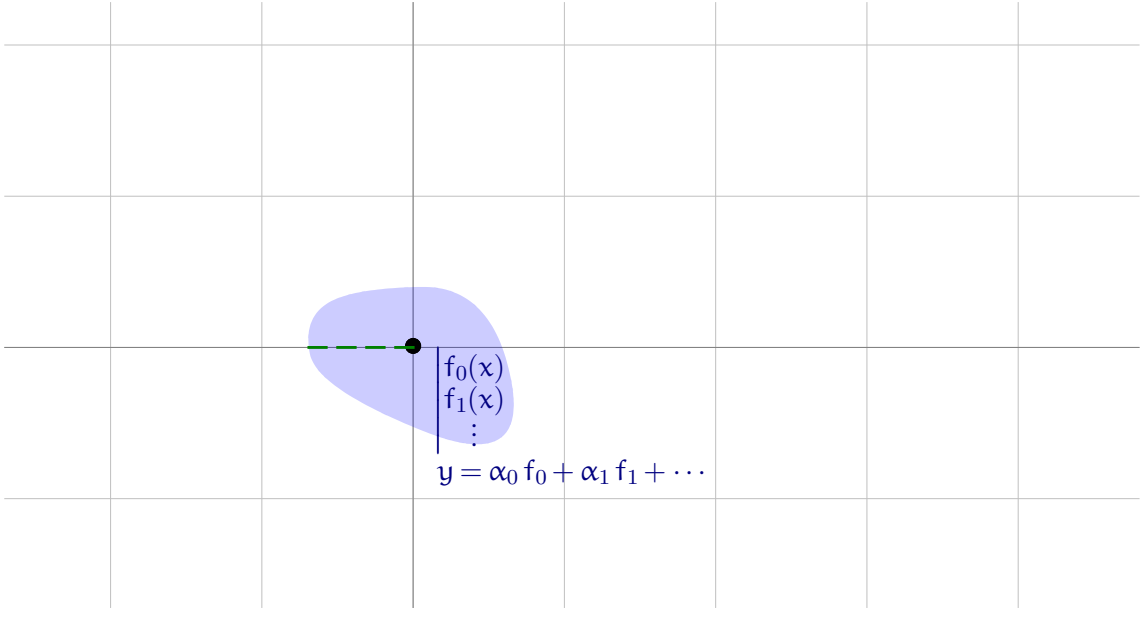
Singular Connection Matrices



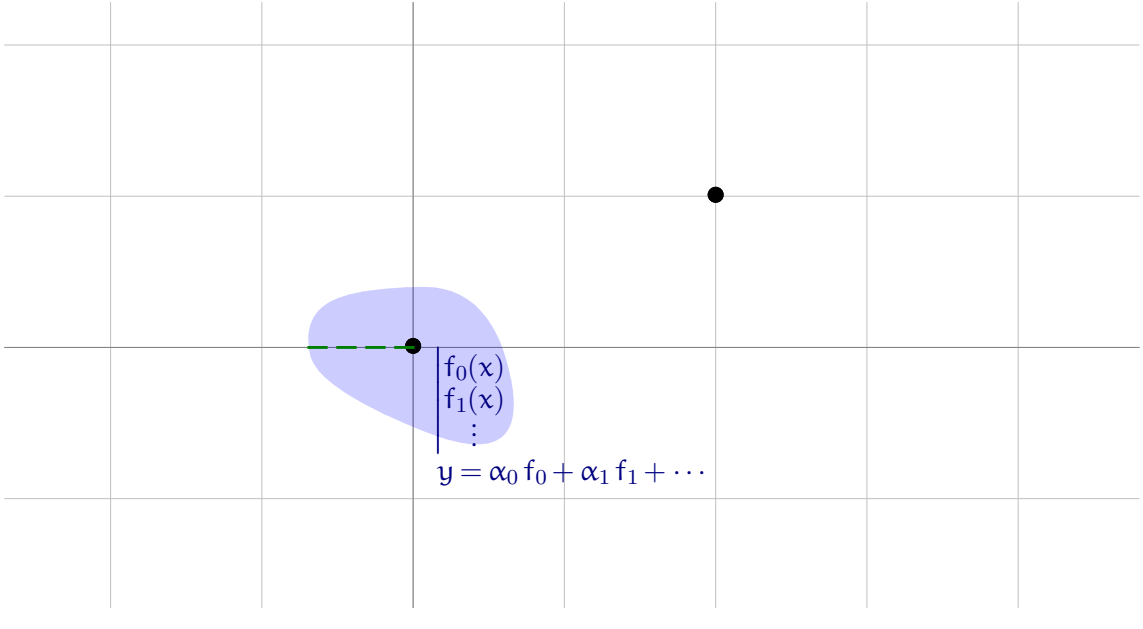
Singular Connection Matrices



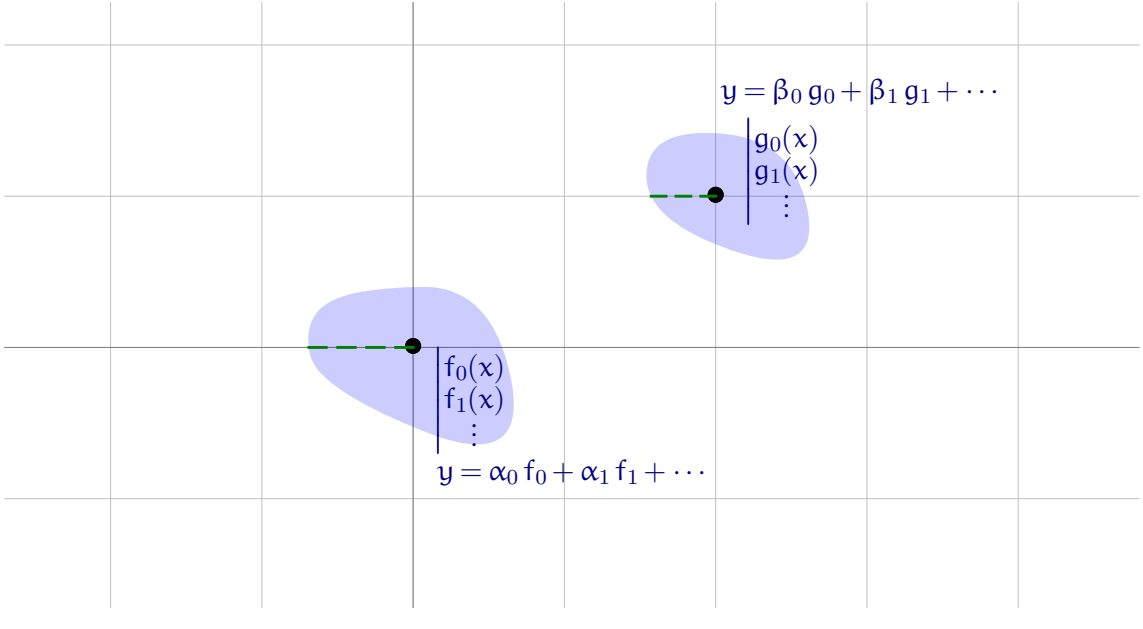
Singular Connection Matrices



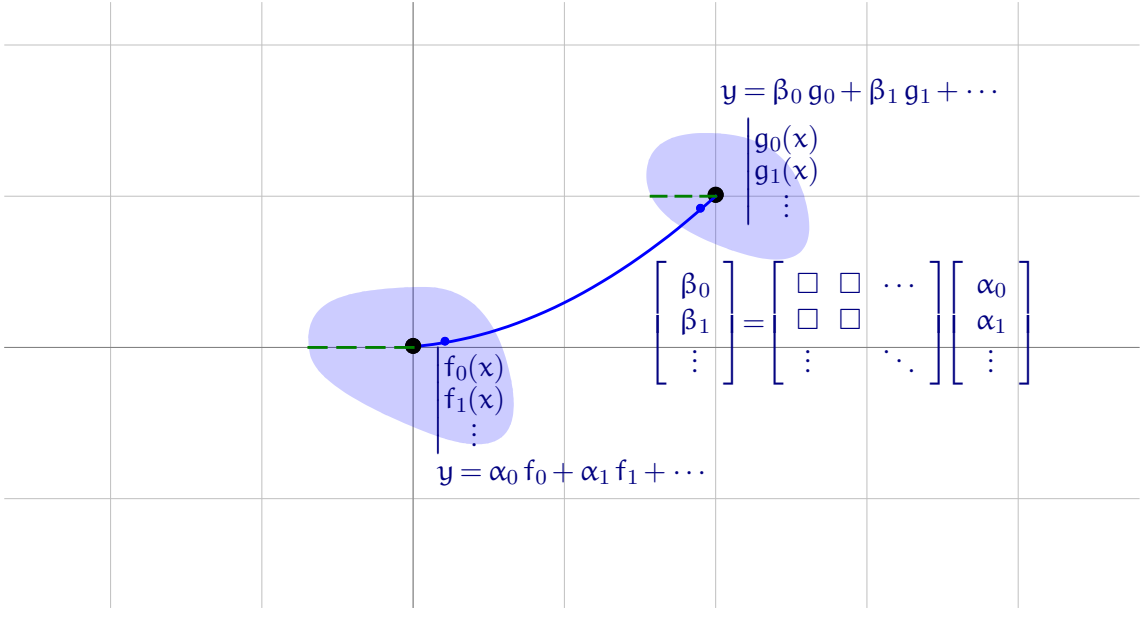
Singular Connection Matrices



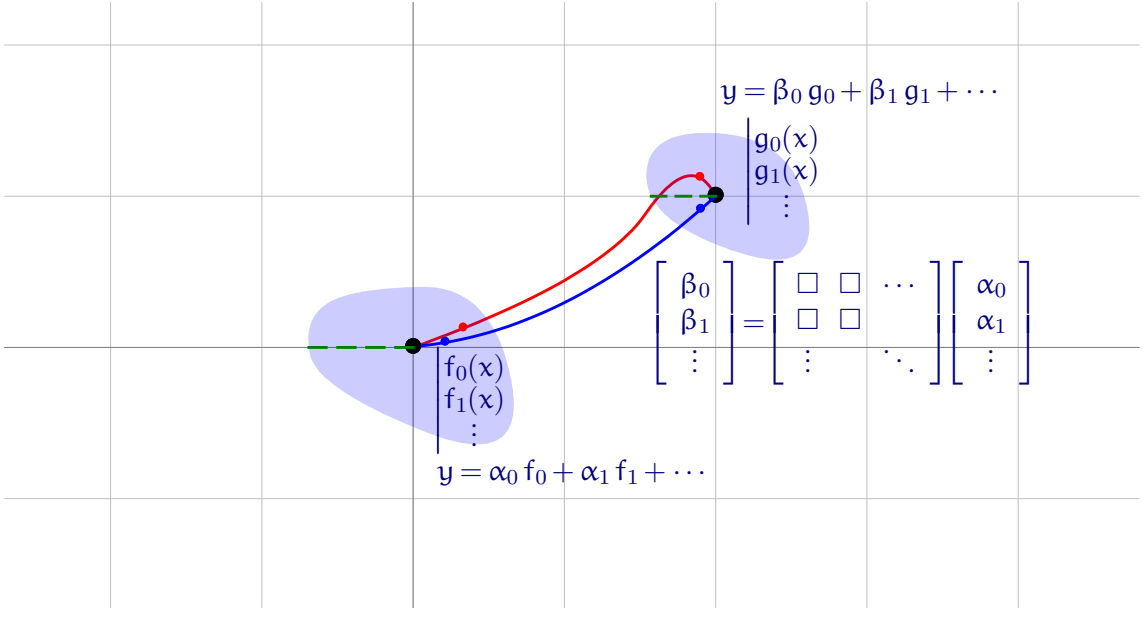
Singular Connection Matrices



Singular Connection Matrices



Singular Connection Matrices



Singular Connection Matrices


```
sage: dop = 2*x^2*(x-1)*Dx^2 + 3*x*(x-1)*Dx + x + 1
sage: dop.local_basis_expansions(0)
sage: dop.local_basis_expansions(1)
sage: mat = dop.numerical_transition_matrix([0,1])
sage: [list(row) for row in mat.rows()]
sage:
```

Singular Connection Matrices


```
sage: dop = 2*x^2*(x-1)*Dx^2 + 3*x*(x-1)*Dx + x + 1

sage: dop.local_basis_expansions(0)

[x^(-1) - 2*1 - x - 4/9*x^2 - 11/45*x^3,
 x^(1/2) + 2/5*x^(3/2) + 1/5*x^(5/2) + 16/135*x^(7/2) + 116/1485*x^(9/2)]

sage: dop.local_basis_expansions(1)

sage: mat = dop.numerical_transition_matrix([0,1])

sage: [list(row) for row in mat.rows()]

sage:
```

Singular Connection Matrices


```
sage: dop = 2*x^2*(x-1)*Dx^2 + 3*x*(x-1)*Dx + x + 1

sage: dop.local_basis_expansions(0)

[x^(-1) - 2*1 - x - 4/9*x^2 - 11/45*x^3,
 x^(1/2) + 2/5*x^(3/2) + 1/5*x^(5/2) + 16/135*x^(7/2) + 116/1485*x^(9/2)]

sage: dop.local_basis_expansions(1)

[1 - (x - 1)*log(x - 1) + 5/4*(x - 1)^2*log(x - 1) - 3/8*(x - 1)^2 - 4/3*(x - 1)^3
*log(x - 1) + 67/144*(x - 1)^3 + 395/288*(x - 1)^4*log(x - 1) - 215/432*(x - 1)^4,
 (x - 1) - 5/4*(x - 1)^2 + 4/3*(x - 1)^3 - 395/288*(x - 1)^4]

sage: mat = dop.numerical_transition_matrix([0,1])

sage: [list(row) for row in mat.rows()]

sage:
```

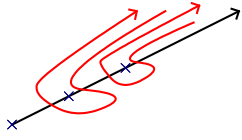
Singular Connection Matrices


```
sage: dop = 2*x^2*(x-1)*Dx^2 + 3*x*(x-1)*Dx + x + 1

sage: dop.local_basis_expansions(0)

[x^(-1) - 2*1 - x - 4/9*x^2 - 11/45*x^3,
 x^(1/2) + 2/5*x^(3/2) + 1/5*x^(5/2) + 16/135*x^(7/2) + 116/1485*x^(9/2)]

sage: dop.local_basis_expansions(1)

[1 - (x - 1)*log(x - 1) + 5/4*(x - 1)^2*log(x - 1) - 3/8*(x - 1)^2 - 4/3*(x - 1)^3
*log(x - 1) + 67/144*(x - 1)^3 + 395/288*(x - 1)^4*log(x - 1) - 215/432*(x - 1)^4,
 (x - 1) - 5/4*(x - 1)^2 + 4/3*(x - 1)^3 - 395/288*(x - 1)^4]

sage: mat = dop.numerical_transition_matrix([0,1])

sage: [list(row) for row in mat.rows()]

[[[-3.5066299264057947 +/- 3.63e-17] + [+/- 2.25e-20]*I,
  [2.1726930052195791 +/- 4.40e-17] + [+/- 1.32e-20]*I],
 [[0.61309036792451532 +/- 2.18e-18] + [-11.016402815654562 +/- 6.06e-17]*I,
  [-0.80762932313757961 +/- 1.47e-19] + [6.8257163837037599 +/- 4.22e-17]*I]]

sage:
```

Stokes Matrices

[Loday-Richaud, M., Remy, 2026]

For irregular singularities of pure level 1, reduces to regular singular connection problems in the Borel plane.

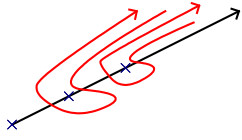


```
sage: from ore_algebra.analytic.stokes import stokes_dict
sage: dop = 2*(x^2*Dx)^3 + (x-5)*(x^2*Dx)^2 + (2*x+2)*(x^2*Dx) - 2*x
sage: dop.generalized_series_solutions()
sage: stokes = stokes_dict(dop)
sage: stokes.keys()  # 1 = exp(i*0), -1 = exp(i*pi) stand for the directions 0, pi
sage: stokes[1].change_ring(ComplexField(32))
sage: list(stokes[1].row(2))
sage:
```

Stokes Matrices

[Loday-Richaud, M., Remy, 2026]

For irregular singularities of pure level 1, reduces to regular singular connection problems in the Borel plane.

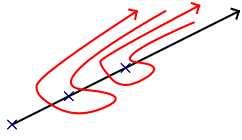


```
sage: from ore_algebra.analytic.stokes import stokes_dict
sage: dop = 2*(x^2*Dx)^3 + (x-5)*(x^2*Dx)^2 + (2*x+2)*(x^2*Dx) - 2*x
sage: dop.generalized_series_solutions()
[exp(-1/2*x^(-1))*x^(-1/2)*(1 - 2/3*x - 1/3*x^2 - 4/9*x^3 - 25/27*x^4 + O(x^5)),
 exp(-2*x^(-1))*x^(-1)*(1 + x + O(x^5)),
 x*(1 + 4*x + 45/2*x^2 + 333/2*x^3 + 3087/2*x^4 + O(x^5))]
sage: stokes = stokes_dict(dop)
sage: stokes.keys()  # 1 = exp(i*0), -1 = exp(i*pi) stand for the directions 0, pi
sage: stokes[1].change_ring(ComplexField(32))
sage: list(stokes[1].row(2))
sage:
```

Stokes Matrices

[Loday-Richaud, M., Remy, 2026]

For irregular singularities of pure level 1, reduces to regular singular connection problems in the Borel plane.

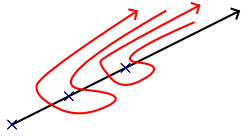


```
sage: from ore_algebra.analytic.stokes import stokes_dict
sage: dop = 2*(x^2*Dx)^3 + (x-5)*(x^2*Dx)^2 + (2*x+2)*(x^2*Dx) - 2*x
sage: dop.generalized_series_solutions()
[exp(-1/2*x^(-1))*x^(-1/2)*(1 - 2/3*x - 1/3*x^2 - 4/9*x^3 - 25/27*x^4 + O(x^5)),
 exp(-2*x^(-1))*x^(-1)*(1 + x + O(x^5)),
 x*(1 + 4*x + 45/2*x^2 + 333/2*x^3 + 3087/2*x^4 + O(x^5))]
sage: stokes = stokes_dict(dop)
sage: stokes.keys()  # 1 = exp(i*0), -1 = exp(i*pi) stand for the directions 0, pi
dict_keys([1, -1])
sage: stokes[1].change_ring(ComplexField(32))
sage: list(stokes[1].row(2))
sage:
```

Stokes Matrices

[Loday-Richaud, M., Remy, 2026]

For irregular singularities of pure level 1, reduces to regular singular connection problems in the Borel plane.

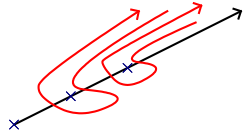


```
sage: from ore_algebra.analytic.stokes import stokes_dict
sage: dop = 2*(x^2*Dx)^3 + (x-5)*(x^2*Dx)^2 + (2*x+2)*(x^2*Dx) - 2*x
sage: dop.generalized_series_solutions()
[exp(-1/2*x^(-1))*x^(-1/2)*(1 - 2/3*x - 1/3*x^2 - 4/9*x^3 - 25/27*x^4 + O(x^5)),
 exp(-2*x^(-1))*x^(-1)*(1 + x + O(x^5)),
 x*(1 + 4*x + 45/2*x^2 + 333/2*x^3 + 3087/2*x^4 + O(x^5))]
sage: stokes = stokes_dict(dop)
sage: stokes.keys()  # 1 = exp(i*0), -1 = exp(i*pi) stand for the directions 0, pi
dict_keys([1, -1])
sage: stokes[1].change_ring(ComplexField(32))
[                   1.00000000                    0.000000000 0.000000000]
[-2.32580238e-23 - 4.45622804*I                    1.00000000 0.000000000]
[ 4.83679830 - 1.23209063e-23*I 5.52974675e-24 + 2.17080376*I  1.00000000]
sage: list(stokes[1].row(2))
sage:
```

Stokes Matrices

[Loday-Richaud, M., Remy, 2026]

For irregular singularities of pure level 1, reduces to regular singular connection problems in the Borel plane.

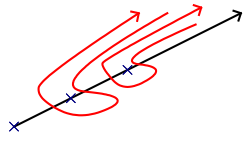


```
sage: from ore_algebra.analytic.stokes import stokes_dict
sage: dop = 2*(x^2*Dx)^3 + (x-5)*(x^2*Dx)^2 + (2*x+2)*(x^2*Dx) - 2*x
sage: dop.generalized_series_solutions()
[exp(-1/2*x^(-1))*x^(-1/2)*(1 - 2/3*x - 1/3*x^2 - 4/9*x^3 - 25/27*x^4 + O(x^5)),
 exp(-2*x^(-1))*x^(-1)*(1 + x + O(x^5)),
 x*(1 + 4*x + 45/2*x^2 + 333/2*x^3 + 3087/2*x^4 + O(x^5))]
sage: stokes = stokes_dict(dop)
sage: stokes.keys()  # 1 = exp(i*0), -1 = exp(i*pi) stand for the directions 0, pi
dict_keys([1, -1])
sage: stokes[1].change_ring(ComplexField(32))
[                   1.00000000                    0.000000000 0.000000000]
[-2.32580238e-23 - 4.45622804*I                     1.00000000 0.000000000]
[ 4.83679830 - 1.23209063e-23*I 5.52974675e-24 + 2.17080376*I  1.00000000]
sage: list(stokes[1].row(2))
[[4.8367983046245809349 +/- 3.46e-20] + [+/- 2.84e-23]*I,
 [+/- 1.28e-23] + [2.1708037636748029781 +/- 3.50e-21]*I,
 1.0000000000000000000000]
sage:
```

Stokes Matrices

[Loday-Richaud, M., Remy, 2026]

For irregular singularities of pure level 1, reduces to regular singular connection problems in the Borel plane.



```
sage: from ore_algebra.analytic.stokes import stokes_dict
sage: dop = 2*(x^2*Dx)^3 + (x-5)*(x^2*Dx)^2 + (2*x+2)*(x^2*Dx) - 2*x
sage: dop.generalized_series_solutions()
[exp(-1/2*x^(-1))*x^(-1/2)*(1 - 2/3*x - 1/3*x^2 - 4/9*x^3 - 25/27*x^4 + O(x^5)),
 exp(-2*x^(-1))*x^(-1)*(1 + x + O(x^5)),
 x*(1 + 4*x + 45/2*x^2 + 333/2*x^3 + 3087/2*x^4 + O(x^5))]
sage: stokes = stokes_dict(dop)
sage: stokes.keys()  # 1 = exp(i*0), -1 = exp(i*pi) stand for the directions 0, pi
dict_keys([1, -1])
sage: stokes[1].change_ring(ComplexField(32))
[                  1.00000000                   0.000000000 0.000000000]
[-2.32580238e-23 - 4.45622804*I                   1.00000000 0.000000000]
[ 4.83679830 - 1.23209063e-23*I 5.52974675e-24 + 2.17080376*I  1.00000000]
sage: list(stokes[1].row(2))
[[4.8367983046245809349 +/- 3.46e-20] + [+/- 2.84e-23]*I,
 [+/- 1.28e-23] + [2.1708037636748029781 +/- 3.50e-21]*I,
 1.0000000000000000000000]
sage:
```

Also some experimental support for Borel summation.

Summary So Far

Input.

- a differential operator $L = a_r(x)\,D^r + \cdots + a_0(x)$ with $a_i \in \mathbb{C}[x]$
- a path in $\mathbb{C} \setminus \{\text{singular points}\}$ but possibly joining regular singular points
- a working precision ε

Output.

connection matrix	entries are intervals containing the mathematical value intervals $\to$ points as $\varepsilon \to 0$

Stokes matrices (pure level 1)
Borel summation (experimental)
…

SageMath package available
GNU LGPL

Application: From Numeric to Exact Solutions

Degree of an Algebraic Solution

[cf. Lairez 2021, Yurkevich 2023]

Yang-Zagier Numbers [Zagier 2018]

$$(\tilde{c}_n)_{n\geq 0} = \left(1, -\frac{161}{288}, \frac{26605753}{24883200}, \ldots\right) \qquad \tilde{c}_n = 1800^n\, c_n$$

$$f(x) = \sum_{n=0}^{\infty} \tilde{c}_n x^n$$

$$\begin{aligned} &1800\,x\,(7x-62)\,(x^2+50x+20) && f''(x)\\ &+720\,(42x^3+173x^2-14230x-620) && f'(x)\\ &+6048x^2-139453x-249550 && f(x) \;=\; 0 \end{aligned}$$

Degree of an Algebraic Solution

[cf. Lairez 2021, Yurkevich 2023]

Yang-Zagier Numbers [Zagier 2018]

$$(\tilde{c}_n)_{n\geq 0} = \left(1, -\frac{161}{288}, \frac{26605753}{24883200}, \ldots\right) \qquad \tilde{c}_n = 1800^n\, c_n$$

$$f(x) = \sum_{n=0}^{\infty} \tilde{c}_n x^n$$

$$\begin{aligned} &1800\,x\,(7x-62)\,(x^2+50x+20)\;f''(x) \\ &+720\,(42x^3+173x^2-14230x-620)\;f'(x) \\ &+6048x^2-139453x-249550\;f(x) \;=\; 0 \end{aligned}$$

Theorem. The series $f(x)$ is algebraic over $\mathbb{Q}(x)$. [Yurkevich 2022]

Degree of an Algebraic Solution

[cf. Lairez 2021, Yurkevich 2023]

Yang-Zagier Numbers [Zagier 2018]

$$(\tilde{c}_n)_{n\geq 0} = \left(1, -\frac{161}{288}, \frac{26605753}{24883200}, \ldots\right) \qquad \tilde{c}_n = 1800^n\, c_n$$

$$f(x) = \sum_{n=0}^{\infty} \tilde{c}_n x^n$$

$$\begin{aligned} &1800\,x\,(7x-62)\,(x^2+50x+20) && f''(x)\\ &+720\,(42x^3+173x^2-14230x-620) && f'(x)\\ &+6048\,x^2-139453x-249550 && f(x) \;=\; 0 \end{aligned}$$

Theorem. The series $f(x)$ is algebraic over $\mathbb{Q}(x)$. [Yurkevich 2022]

```
sage: dop = (1800*x*(7*x-62)*(x^2 + 50*x + 20)*Dx^2
             + 720*(42*x^3 + 173*x^2 - 14230*x - 620)*Dx
             + 6048*x^2 - 139453*x - 249550)

sage: dop.local_basis_expansions(0, 3)

sage: ini = vector(CBF, [1, 0])
```

Monodromy

Facts.

- If $P(x, f(x)) = 0$ for some $P \in \mathbb{C}[x, y]$, then also $P(x, M \cdot f(x)) = 0$.
- The coefficients of $\prod_{M \in \mathrm{Mon}} (y - M \cdot f(x))$ are analytic on $\mathbb{P}^1(\mathbb{C}) \setminus \{\text{sing}\}$ with Puiseux expansions at the singularities, so they are in $\mathbb{C}(x)$.

```
sage: mon = monodromy_matrices(dop, 0, 1e-10)
sage: mon[0]
sage: len(mon)
sage:
```

Monodromy

Facts.

- If $P(x, f(x)) = 0$ for some $P \in \mathbb{C}[x, y]$, then also $P(x, M \cdot f(x)) = 0$.
- The coefficients of $\prod_{M \in \mathrm{Mon}} (y - M \cdot f(x))$ are analytic on $\mathbb{P}^1(\mathbb{C}) \setminus \{\text{sing}\}$ with Puiseux expansions at the singularities, so they are in $\mathbb{C}(x)$.

```
sage: mon = monodromy_matrices(dop, 0, 1e-10)

sage: mon[0]

[1.0000000000                                                              0]
[           0 [0.3090169944 +/- 5.64e-11] + [-0.9510565163 +/- 1.53e-11]*I]

sage: len(mon)

sage:
```

Monodromy

Facts.

- If $P(x, f(x)) = 0$ for some $P \in \mathbb{C}[x, y]$, then also $P(x, M \cdot f(x)) = 0$.
- The coefficients of $\prod_{M \in \mathrm{Mon}} (y - M \cdot f(x))$ are analytic on $\mathbb{P}^1(\mathbb{C}) \setminus \{\text{sing}\}$ with Puiseux expansions at the singularities, so they are in $\mathbb{C}(x)$.

```
sage: mon = monodromy_matrices(dop, 0, 1e-10)

sage: mon[0]

[1.0000000000                                                              0]
[           0 [0.3090169944 +/- 5.64e-11] + [-0.9510565163 +/- 1.53e-11]*I]

sage: len(mon)

3

sage:
```

Monodromy


```
sage: def neq(v, w):
          for a, b in zip(v, w):
              if not a.overlaps(b):
                  return True
          return False

sage: orbit = []; new = [ini]
      while new:
          orbit.extend(new)
          act = [mat*branch for branch in new for mat in mon]
          new = [f for i, f in enumerate(act) if all(neq(f, g) for g in act[:i])
                                              and all(neq(f, g) for g in orbit)]

sage: orbit[17]

sage: len(orbit)

sage:
```

Monodromy


```
sage: def neq(v, w):
          for a, b in zip(v, w):
              if not a.overlaps(b):
                  return True
          return False

sage: orbit = []; new = [ini]
      while new:
          orbit.extend(new)
          act = [mat*branch for branch in new for mat in mon]
          new = [f for i, f in enumerate(act) if all(neq(f, g) for g in act[:i])
                                              and all(neq(f, g) for g in orbit)]

sage: orbit[17]

([-0.309016994 +/- 7.96e-10] + [-0.951056516 +/- 6.99e-10]*I, [+/- 3.37e-10] + [+/
- 3.40e-10]*I)

sage: len(orbit)

sage:
```

Monodromy


```
sage: def neq(v, w):
          for a, b in zip(v, w):
              if not a.overlaps(b):
                  return True
          return False

sage: orbit = []; new = [ini]
      while new:
          orbit.extend(new)
          act = [mat*branch for branch in new for mat in mon]
          new = [f for i, f in enumerate(act) if all(neq(f, g) for g in act[:i])
                                              and all(neq(f, g) for g in orbit)]

sage: orbit[17]

([-0.309016994 +/- 7.96e-10] + [-0.951056516 +/- 6.99e-10]*I, [+/- 3.37e-10] + [+/
- 3.40e-10]*I)

sage: len(orbit)

120

sage:
```

Monodromy


```
sage: def neq(v, w):
          for a, b in zip(v, w):
              if not a.overlaps(b):
                  return True
          return False

sage: orbit = []; new = [ini]
      while new:
          orbit.extend(new)
          act = [mat*branch for branch in new for mat in mon]
          new = [f for i, f in enumerate(act) if all(neq(f, g) for g in act[:i])
                                              and all(neq(f, g) for g in orbit)]

sage: orbit[17]

([-0.309016994 +/- 7.96e-10] + [-0.951056516 +/- 6.99e-10]*I, [+/- 3.37e-10] + [+/
- 3.40e-10]*I)

sage: len(orbit)

120

sage:
```

Remark. This gives a rigorous **lower bound** on the degree.

Heuristic Computation of the Galois Group


```
sage: def img(mat, i):
          f = mat*orbit[i]
          for j, g in enumerate(orbit):
              if not neq(f, g):
                  return j

sage: PermutationGroup([[1 + img(mat, i) for i in range(len(orbit))]
                        for mat in mon])

sage:
```

Heuristic Computation of the Galois Group


```
sage: def img(mat, i):
          f = mat*orbit[i]
          for j, g in enumerate(orbit):
              if not neq(f, g):
                  return j

sage: PermutationGroup([[1 + img(mat, i) for i in range(len(orbit))]
                        for mat in mon])
Permutation Group with generators [(2,4,9,21,38)(3,7,16,34,65)(5,12,26,23,47)(8,19
,39,71,106)(10,15,32,44,66)(11,17,36,33,43)(13,28,55,91,102)(14,24,22,45,79)(20,41
,75,52,88)(25,51,72,86,48)(27,37,68,103,30)(29,58,94,116,114)(35,64,76,70,100)(40,
73,84,78,56)(42,77,109,115,112)(46,82,53,60,85)(49,61,98,81,101)(50,62,67,99,80)(5
4,90,107,74,96)(57,69,105,92,83)(59,93,87,108,119)(104,117,113,111,118), (1,2,5,13
,29)(3,6,14,30,59)(4,10,23,48,39)(7,17,37,69,95)(8,15,31,60,96)(9,22,46,83,109)(11
,24,49,68,72)(16,21,43,78,88)(18,32,62,90,112)(19,40,74,94,115)(20,33,61,97,105)(2
5,50,85,104,107)(26,53,89,58,71)(28,56,47,81,82)(34,66,101,118,116)(35,44,45,80,73
)(38,63,98,91,75)(42,64,99,117,120)(51,87,77,65,100)(52,86,55,92,114)(57,93,84,103
,111)(70,79,110,113,108), (1,3,8,20,42)(2,6,15,33,64)(4,11,25,52,65)(5,14,31,61,99
)(7,18,38,70,71)(10,24,50,86,100)(12,27,54,41,76)(13,30,60,97,117)(16,35,56,93,115
)(17,32,63,79,26)(19,21,44,47,84)(23,49,85,55,51)(28,57,94,88,73)(29,59,96,105,120
)(36,67,102,119,106)(37,62,98,110,53)(39,72,107,114,77)(40,43,45,81,103)(48,68,104
,92,87)(58,95,112,75,108)(69,90,91,113,89)(74,78,80,82,111)]

sage:
```

Heuristic Computation of the Galois Group


```
sage: def img(mat, i):
          f = mat*orbit[i]
          for j, g in enumerate(orbit):
              if not neq(f, g):
                  return j

sage: PermutationGroup([[1 + img(mat, i) for i in range(len(orbit))]
                        for mat in mon])
Permutation Group with generators [(2,4,9,21,38)(3,7,16,34,65)(5,12,26,23,47)(8,19
,39,71,106)(10,15,32,44,66)(11,17,36,33,43)(13,28,55,91,102)(14,24,22,45,79)(20,41
,75,52,88)(25,51,72,86,48)(27,37,68,103,30)(29,58,94,116,114)(35,64,76,70,100)(40,
73,84,78,56)(42,77,109,115,112)(46,82,53,60,85)(49,61,98,81,101)(50,62,67,99,80)(5
4,90,107,74,96)(57,69,105,92,83)(59,93,87,108,119)(104,117,113,111,118), (1,2,5,13
,29)(3,6,14,30,59)(4,10,23,48,39)(7,17,37,69,95)(8,15,31,60,96)(9,22,46,83,109)(11
,24,49,68,72)(16,21,43,78,88)(18,32,62,90,112)(19,40,74,94,115)(20,33,61,97,105)(2
5,50,85,104,107)(26,53,89,58,71)(28,56,47,81,82)(34,66,101,118,116)(35,44,45,80,73
)(38,63,98,91,75)(42,64,99,117,120)(51,87,77,65,100)(52,86,55,92,114)(57,93,84,103
,111)(70,79,110,113,108), (1,3,8,20,42)(2,6,15,33,64)(4,11,25,52,65)(5,14,31,61,99
)(7,18,38,70,71)(10,24,50,86,100)(12,27,54,41,76)(13,30,60,97,117)(16,35,56,93,115
)(17,32,63,79,26)(19,21,44,47,84)(23,49,85,55,51)(28,57,94,88,73)(29,59,96,105,120
)(36,67,102,119,106)(37,62,98,110,53)(39,72,107,114,77)(40,43,45,81,103)(48,68,104
,92,87)(58,95,112,75,108)(69,90,91,113,89)(74,78,80,82,111)]
sage:
```

$C_5 \times \mathrm{SL}_2(5)$

Minimal Annihilators

Problem. What is the differential operator of **miminal order** annihilating a given f?

Linear algebra / Hermite–Padé approximants: (code by M. Kauers)

```
sage: from ore_algebra import guess
sage: dop = -x^2*Dx^4 + (x^2 - 7*x)*Dx^3 + (15*x - 9)*Dx^2 + 24*Dx
sage: sol = dop.power_series_solutions(20)  # 40
sage: guess(list(sol[0]), DOP)
sage:
```

Algebraic state of the art: generalized Fuchs relation + linear programming
[Bostan–Rivoal–Salvy 2023]

motivated by the computation of algebraic values of E-functions [Adamczewski–Rivoal 2018]

Minimal Annihilators

Problem. What is the differential operator of **miminal order** annihilating a given f?

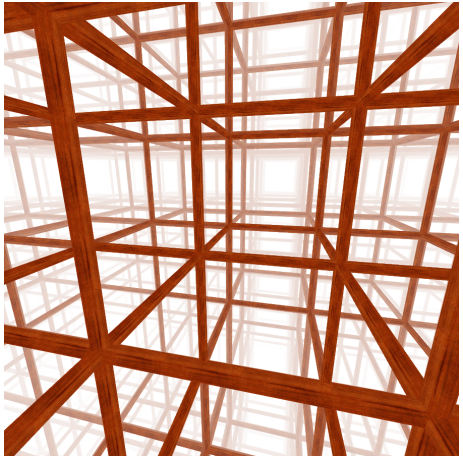
Linear algebra / Hermite–Padé approximants: (code by M. Kauers)

```
sage: from ore_algebra import guess
sage: dop = -x^2*Dx^4 + (x^2 - 7*x)*Dx^3 + (15*x - 9)*Dx^2 + 24*Dx
sage: sol = dop.power_series_solutions(20)  # 40
sage: guess(list(sol[0]), DOP)
-x^3*Dx^3 + (x^3 - 4*x^2)*Dx^2 + (12*x^2 - x)*Dx + 1
sage:
```

Algebraic state of the art: generalized Fuchs relation + linear programming
[Bostan–Rivoal–Salvy 2023]

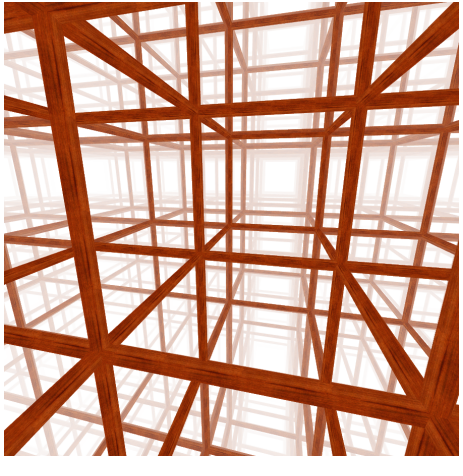
motivated by the computation of algebraic values of E-functions [Adamczewski–Rivoal 2018]

Pólya walks



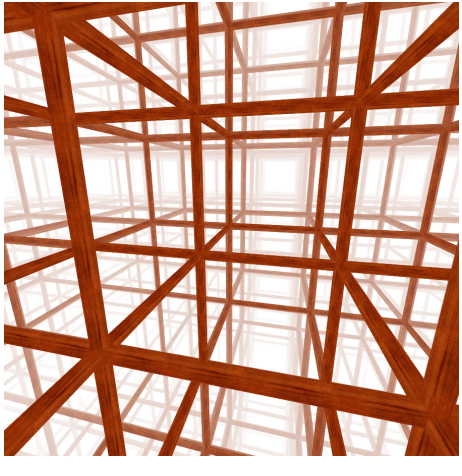
```
sage: from ore_algebra.examples import polya
sage: polya.dop[3]
sage: polya.dop[3].local_basis_expansions(0, 3)
sage: ini = vector([0, 0, 1])
sage:
```

Pólya walks



```
sage: from ore_algebra.examples import polya
sage: polya.dop[3]
(144*z^6 - 40*z^4 + z^2)*Dz^3 + (1296*z^5 -
240*z^3 + 3*z)*Dz^2 + (2592*z^4 - 288*
z^2 + 1)*Dz + 864*z^3 - 48*z
sage: polya.dop[3].local_basis_expansions(0, 3)
sage: ini = vector([0, 0, 1])
sage:
```

Pólya walks



```
sage: from ore_algebra.examples import polya
sage: polya.dop[3]
(144*z^6 - 40*z^4 + z^2)*Dz^3 + (1296*z^5 -
240*z^3 + 3*z)*Dz^2 + (2592*z^4 - 288*
z^2 + 1)*Dz + 864*z^3 - 48*z
sage: polya.dop[3].local_basis_expansions(0, 3)
[1/2*log(z)^2 + 3*z^2*log(z)^2 + 7*z^2*log(z),
 log(z) + 6*z^2*log(z) + 7*z^2,
 1 + 6*z^2]
sage: ini = vector([0, 0, 1])
sage:
```

Minimal Annihilators and Monodromy

As before: if $L(f) = 0$ with $L \in \mathbb{C}(x)\langle D_x \rangle$, then $L(M \cdot f) = 0$.

```
sage: mon = monodromy_matrices(polya.dop[3], 0, 1e-10)
sage: ini
sage: mon[1]*ini
sage: mon[2]*ini
sage: matrix([ini, mon[1]*ini, mon[2]*ini]).det()
sage:
```

- Interval arithmetic **underestimates** the dimension of a span → rigorous proof
- Increase the precision until the linear independence check succeeds
- This method is complete for Fuchsian operators

Minimal Annihilators and Monodromy

As before: if $L(f) = 0$ with $L \in \mathbb{C}(x)\langle D_x \rangle$, then $L(M \cdot f) = 0$.

```
sage: mon = monodromy_matrices(polya.dop[3], 0, 1e-10)
sage: ini
(0, 0, 1)
sage: mon[1]*ini
sage: mon[2]*ini
sage: matrix([ini, mon[1]*ini, mon[2]*ini]).det()
sage:
```

- Interval arithmetic **underestimates** the dimension of a span → rigorous proof
- Increase the precision until the linear independence check succeeds
- This method is complete for Fuchsian operators

Minimal Annihilators and Monodromy

As before: if $L(f) = 0$ with $L \in \mathbb{C}(x)\langle D_x \rangle$, then $L(M \cdot f) = 0$.

```
sage: mon = monodromy_matrices(polya.dop[3], 0, 1e-10)
sage: ini
(0, 0, 1)
sage: mon[1]*ini
([1.215854204 +/- 4.22e-10], [+/- 1.66e-10], [+/- 1.06e-10])
sage: mon[2]*ini
sage: matrix([ini, mon[1]*ini, mon[2]*ini]).det()
sage:
```

- Interval arithmetic **underestimates** the dimension of a span → rigorous proof
- Increase the precision until the linear independence check succeeds
- This method is complete for Fuchsian operators

Minimal Annihilators and Monodromy

As before: if $L(f) = 0$ with $L \in \mathbb{C}(x)\langle D_x \rangle$, then $L(M \cdot f) = 0$.

```
sage: mon = monodromy_matrices(polya.dop[3], 0, 1e-10)
sage: ini
(0, 0, 1)
sage: mon[1]*ini
([1.215854204 +/- 4.22e-10], [+/- 1.66e-10], [+/- 1.06e-10])
sage: mon[2]*ini
([3.6475626 +/- 2.41e-8] + [+/- 1.57e-8]*I, [+/- 1.93e-8] + [-3.8197186 +/- 4.04e-
8]*I, [-2.0000000 +/- 1.09e-8] + [+/- 1.30e-8]*I)
sage: matrix([ini, mon[1]*ini, mon[2]*ini]).det()
sage:
```

- Interval arithmetic **underestimates** the dimension of a span → rigorous proof
- Increase the precision until the linear independence check succeeds
- This method is complete for Fuchsian operators

Minimal Annihilators and Monodromy

As before: if $L(f) = 0$ with $L \in \mathbb{C}(x)\langle D_x \rangle$, then $L(M \cdot f) = 0$.

```
sage: mon = monodromy_matrices(polya.dop[3], 0, 1e-10)
sage: ini
(0, 0, 1)
sage: mon[1]*ini
([1.215854204 +/- 4.22e-10], [+/- 1.66e-10], [+/- 1.06e-10])
sage: mon[2]*ini
([3.6475626 +/- 2.41e-8] + [+/- 1.57e-8]*I, [+/- 1.93e-8] + [-3.8197186 +/- 4.04e-
8]*I, [-2.0000000 +/- 1.09e-8] + [+/- 1.30e-8]*I)
sage: matrix([ini, mon[1]*ini, mon[2]*ini]).det()
[+/- 2.40e-8] + [-4.6442210 +/- 5.01e-8]*I
sage:
```

- Interval arithmetic **underestimates** the dimension of a span → rigorous proof
- Increase the precision until the linear independence check succeeds
- This method is complete for Fuchsian operators

The Schlesinger–Ramis Density Theorem

gruppe enthalten, d. h. wir haben den Satz:

Wenn die Differentialgleichung (A) der Fuchs'schen Classe angehört, so ist ihre Transformationsgruppe die engste algebraische Gruppe linearer homogener Transformationen, die die Gruppe der Differentialgleichung als Untergruppe in sich schliesst.

Dass dieser Satz für Differentialgleichungen, die nicht der Fuchs-

[Schlesinger 1897]

Theorem. [Schlesinger, 1885]

The differential Galois group of a Fuchsian differential operator

$$L = a_r(x)\, D_x^r + \cdots + a_1(x)\, D_x + a_0(x)$$

is the algebraic group generated by the monodromy matrices

(viewed as elements of $GL_r(\mathbb{C})$ acting on local solutions at a common base point x_0).

The Schlesinger–Ramis Density Theorem

gruppe enthalten, d. h. wir haben den Satz:

Wenn die Differentialgleichung (A) der Fuchs'schen Classe angehört, so ist ihre Transformationsgruppe die engste algebraische Gruppe linearer homogener Transformationen, die die Gruppe der Differentialgleichung als Untergruppe in sich schliesst.

Dass dieser Satz für Differentialgleichungen, die nicht der Fuchs-

[Schlesinger 1897]

Theorem. [Ramis, 1985]

The differential Galois group of a ~~Fuchsian~~ differential operator

$$L = a_r(x)\, D_x^r + \cdots + a_1(x)\, D_x + a_0(x)$$

is the algebraic group generated by the
(i) monodromy matrices, (ii) exponential matrices, (iii) Stokes matrices

(viewed as elements of $GL_r(\mathbb{C})$ acting on local solutions at a common base point x_0).

Factoring Differential Operators

Problem. Given $L \in \bar{\mathbb{Q}}[x]\langle D_x \rangle$, write $L = P \cdot Q$ or prove that no such factorization exists.

- Complete algebraic algorithms: 19th century [Fabry, Markov, Bendixson, Beke,…]
- Practical state of the art: van Hoeij 1997

Factoring Differential Operators

Problem. Given $L \in \bar{\mathbb{Q}}[x]\langle D_x \rangle$, write $L = P \cdot Q$ or prove that no such factorization exists.

A subspace $V \subseteq \mathrm{Sol}(L)$ is the solution space of a right-hand factor

$\Leftrightarrow$ it is invariant under the action of the differential Galois group

For Fuchsian L,

$\Leftrightarrow$ it is invariant under monodromy

- Complete algebraic algorithms: 19th century [Fabry, Markov, Bendixson, Beke,…]
- Practical state of the art: van Hoeij 1997

Factoring Differential Operators

Problem. Given $L \in \bar{\mathbb{Q}}[x]\langle D_x \rangle$, write $L = P \cdot Q$ or prove that no such factorization exists.

A subspace $V \subseteq \mathrm{Sol}(L)$ is the solution space of a right-hand factor

$\Leftrightarrow$ it is invariant under the action of the differential Galois group

For Fuchsian L,

$\Leftrightarrow$ it is invariant under monodromy

"Finding" a factor $\Leftrightarrow$ "finding" an invariant subspace
$\approx$ writing the monodromy mat. in simult. block triangular form

- Complete algebraic algorithms: 19th century [Fabry, Markov, Bendixson, Beke,…]
- Practical state of the art: van Hoeij 1997

Eine lineare Differentialgleichung (A) mit in x rationalen Coefficienten ist dann und nur dann reductibel in dem Sinne, dass sie mit einer linearen Differentialgleichung von niedrigerer Ordnung mit ebenfalls rationalen Coefficienten Lösungen gemein hat, wenn ihre Transformationsgruppe G reductibel ist.

Wenn die vorgelegte Differentialgleichung der Fuchsschen Classe angehört, so ist für ihre Reductibilität in dem jetzt festgehaltenen Sinne schon die Reductibilität der Monodromiegruppe nothwendig und hinreichend.

[Schlesinger 1897]

- A differential equation with rational coefficients is reducible in the sense that is has solutions in common with another one iff its differential Galois group is reducible
- In the Fuchsian case, already the reducibility of the monodromy group is a necessary and sufficient condition

Van der Hoeven's Factoring Algorithm

[van der Hoeven 2007]

Algorithm (sketch). *Input*: $L \in \bar{\mathbb{Q}}[x]\langle D_x \rangle$ *Output*: a nontrivial factor or *Irreducible*

Compute generators of the differential Galois group

Search for a nontrivial invariant subspace V

If the search proves that none exists:

Return *Irreducible*

Otherwise:

Construct $\tilde{Q} \in \mathbb{C}[[x]]\langle D_x \rangle$ such that $\mathrm{Sol}(\tilde{Q}) \approx V$

Reconstruct $Q \in \bar{\mathbb{Q}}[x]\langle D_x \rangle$ with $Q \approx \tilde{Q}$ (Padé + LLL)

If Q divides L, return Q

Increase the precision and Padé order, try again

Van der Hoeven's Factoring Algorithm

[van der Hoeven 2007]

Algorithm (sketch). *Input*: $L \in \bar{\mathbb{Q}}[x]\langle D_x \rangle$ *Output*: a nontrivial factor or *Irreducible*

Compute generators of the differential Galois group

Search for a nontrivial invariant subspace V

If the search proves that none exists: ok at high prec.

Return *Irreducible*

Otherwise:

Construct $\tilde{Q} \in \mathbb{C}[[x]]\langle D_x \rangle$ such that $\mathrm{Sol}(\tilde{Q}) \approx V$

Reconstruct $Q \in \bar{\mathbb{Q}}[x]\langle D_x \rangle$ with $Q \approx \tilde{Q}$ (Padé + LLL)

If Q divides L, return Q

Increase the precision and Padé order, try again

Van der Hoeven's Factoring Algorithm

[van der Hoeven 2007]

Algorithm (sketch). *Input*: $L \in \bar{\mathbb{Q}}[x]\langle D_x\rangle$ *Output*: a nontrivial factor or *Irreducible*

Compute generators of the differential Galois group

Search for a nontrivial invariant subspace V

If the search proves that none exists: ok at high prec.

Return *Irreducible*

Otherwise:

Construct $\tilde{Q} \in \mathbb{C}[[x]]\langle D_x\rangle$ such that $\mathrm{Sol}(\tilde{Q}) \approx V$

Reconstruct $Q \in \bar{\mathbb{Q}}[x]\langle D_x\rangle$ with $Q \approx \tilde{Q}$ (Padé + LLL) almost ok at high prec.

If Q divides L, return Q

Increase the precision and Padé order, try again

Subtlety [Goyer]: When the variety of right-hand factors has $\dim > 0$, find an algebraic point

[→ van der Hoeven 2022]

Factorization Using Norton's Test

[Chyzak, Goyer, M. 2022]

Theorem (special case of Norton's criterion). [Parker 1984, Holt–Rees 1994]

Let $\mathcal{A} \subseteq \mathbb{C}^{r\times r}$ be a matrix algebra.

Suppose $M \in \mathcal{A}$ has a simple eigenvalue λ with $M u = \lambda u$
$v M = \lambda v$.

If both $A \cdot u = \mathbb{C}^{r\times 1}$ and $v \cdot A = \mathbb{C}^{1\times r}$, then the A-module $\mathbb{C}^{r\times 1}$ is irreducible.

Factorization Using Norton's Test

[Chyzak, Goyer, M. 2022]

Theorem (special case of Norton's criterion). [Parker 1984, Holt–Rees 1994]

Let $\mathcal{A} \subseteq \mathbb{C}^{r\times r}$ be a matrix algebra.

Suppose $M \in \mathcal{A}$ has a simple eigenvalue λ with $M\,u = \lambda\,u$
$v\,M = \lambda\,v$.

If both $A \cdot u = \mathbb{C}^{r\times 1}$ and $v \cdot A = \mathbb{C}^{1\times r}$, then the A-module $\mathbb{C}^{r\times 1}$ is irreducible.

- Take $\mathcal{A} = \mathbb{C}[M_1, \ldots, M_k]$, with $M_1, \ldots, M_k \in \mathrm{Mon}$; pick $M \in \mathcal{A}$ at random
- Existence of a simple λ can be certified numerically
- Irreducibility: $A \cdot u = \mathbb{C}^{r\times 1}$ and $v \cdot A = \mathbb{C}^{1\times r}$ can be certified numerically

  two monodromy matrices are often enough
  - Failure of $A \cdot u = \mathbb{C}^{r\times 1}$ $\Rightarrow$ invariant subspace $\Rightarrow$ right-hand factor
  - Failure of $v \cdot A = \mathbb{C}^{1\times r}$ $\Rightarrow$ invariant subspace of the adjoint $\Rightarrow$ left-hand factor

Link with van Hoeij's Algorithm

exponents
↓
Local basis: $(x-\xi)^{\lambda}\left(f_0(x)+f_1(x)\log(x-\xi)+\cdots+f_K(x)\log(x-\xi)^K\right)$

Link with van Hoeij's Algorithm

$$\begin{array}{ccc} \text{exponents} & \rightsquigarrow & \textbf{exponent classes } \lambda+\mathbb{Z} \\ \downarrow & & \\ \lambda & & \end{array}$$

Local basis: $(x-\xi)^{\lambda}\left(f_0(x)+f_1(x)\log(x-\xi)+\cdots+f_K(x)\log(x-\xi)^K\right)$

If $L=P\,Q$ then $\quad \operatorname{expos}(L)/\mathbb{Z} \;=\; \operatorname{expos}(P)/\mathbb{Z} \;\sqcup\; \operatorname{expos}(Q)/\mathbb{Z}$ (as multisets)

Link with van Hoeij's Algorithm

exponents $\rightsquigarrow$ **exponent classes** $\lambda+\mathbb{Z}$

$\downarrow$

Local basis: $(x-\xi)^{\lambda}\left(f_0(x)+f_1(x)\log(x-\xi)+\cdots+f_K(x)\log(x-\xi)^K\right)$

If $L=P\,Q$ then $\quad \mathrm{expos}(L)/\mathbb{Z} \;=\; \mathrm{expos}(P)/\mathbb{Z} \;\sqcup\; \mathrm{expos}(Q)/\mathbb{Z}$ (as multisets)

Main criterion (Fuchsian case).

If $\mathrm{mult}(\lambda+\mathbb{Z})=1$ as an exponent class of L,

a) $\lambda+\mathbb{Z}\in\mathrm{Expo}(\text{proper right factor})$

b) or $\lambda+\mathbb{Z}\in\mathrm{Expo}(\text{proper left factor})$

c) or L is irreducible

Link with van Hoeij's Algorithm

exponents $\rightsquigarrow$ **exponent classes** $\lambda+\mathbb{Z}$

$\downarrow$

Local basis: $(x-\xi)^{\lambda}\left(f_0(x)+f_1(x)\log(x-\xi)+\cdots+f_K(x)\log(x-\xi)^K\right)$

If $L=P\,Q$ then $\quad \mathrm{expos}(L)/\mathbb{Z} \;=\; \mathrm{expos}(P)/\mathbb{Z} \;\sqcup\; \mathrm{expos}(Q)/\mathbb{Z}$ (as multisets)

Main criterion (Fuchsian case).

If $\mathrm{mult}(\lambda+\mathbb{Z})=1$ as an exponent class of L,

a) $\lambda+\mathbb{Z}\in\mathrm{Expo}$(proper right factor) guess the factor by Hermite-Padé approx

b) or $\lambda+\mathbb{Z}\in\mathrm{Expo}$(proper left factor)

c) or L is irreducible

Link with van Hoeij's Algorithm

exponents $\rightsquigarrow$ **exponent classes** $\lambda+\mathbb{Z}$

$\downarrow$

Local basis: $(x-\xi)^{\lambda}\left(f_0(x)+f_1(x)\log(x-\xi)+\cdots+f_K(x)\log(x-\xi)^K\right)$

If $L=P\,Q$ then $\quad \mathrm{expos}(L)/\mathbb{Z} \;=\; \mathrm{expos}(P)/\mathbb{Z} \;\sqcup\; \mathrm{expos}(Q)/\mathbb{Z}$ (as multisets)

Main criterion (Fuchsian case).

If $\mathrm{mult}(\lambda+\mathbb{Z})=1$ as an exponent class of L,

a) $\lambda+\mathbb{Z}\in\mathrm{Expo}(\text{proper right factor})$ — guess the factor by Hermite-Padé approx

b) or $\lambda+\mathbb{Z}\in\mathrm{Expo}(\text{proper left factor})$ — idem (adjoint)

c) or L is irreducible

Link with van Hoeij's Algorithm

exponents $\rightsquigarrow$ **exponent classes** $\lambda+\mathbb{Z}$

$\downarrow$

Local basis: $(x-\xi)^{\lambda}\left(f_0(x)+f_1(x)\log(x-\xi)+\cdots+f_K(x)\log(x-\xi)^K\right)$

If $L=P\,Q$ then $\quad \mathrm{expos}(L)/\mathbb{Z} \;=\; \mathrm{expos}(P)/\mathbb{Z} \;\sqcup\; \mathrm{expos}(Q)/\mathbb{Z}$ (as multisets)

Main criterion (Fuchsian case).

If $\mathrm{mult}(\lambda+\mathbb{Z})=1$ as an exponent class of L,

a) $\lambda+\mathbb{Z}\in\mathrm{Expo}$(proper right factor) — guess the factor by Hermite-Padé approx

b) or $\lambda+\mathbb{Z}\in\mathrm{Expo}$(proper left factor) — idem (adjoint)

c) or L is irreducible — exclude a) and b) using degree bounds

Link with van Hoeij's Algorithm

exponents $\rightsquigarrow$ **exponent classes** $\lambda+\mathbb{Z}$

$\downarrow$

Local basis: $(x-\xi)^{\lambda}\left(f_0(x)+f_1(x)\log(x-\xi)+\cdots+f_K(x)\log(x-\xi)^K\right)$

If $L=P\,Q$ then $\quad \mathrm{expos}(L)/\mathbb{Z} \;=\; \mathrm{expos}(P)/\mathbb{Z} \;\sqcup\; \mathrm{expos}(Q)/\mathbb{Z}$ (as multisets)

Main criterion (Fuchsian case).

If $\mathrm{mult}(\lambda+\mathbb{Z})=1$ as an exponent class of L,

a) $\lambda+\mathbb{Z}\in\mathrm{Expo}(\text{proper right factor})$ — guess the factor by Hermite-Padé approx

b) or $\lambda+\mathbb{Z}\in\mathrm{Expo}(\text{proper left factor})$ — idem (adjoint)

c) or L is irreducible — exclude a) and b) using degree bounds

Eigenvalues of the **local** monodromy at ξ
$=\exp(\text{exponent classes})$

($\Rightarrow$ the numeric Norton test "interpolates"
between van Hoeij's and van der Hoeven's method)

Link with van Hoeij's Algorithm

exponents $\rightsquigarrow$ **exponent classes** $\lambda+\mathbb{Z}$

$\downarrow$

Local basis: $(x-\xi)^{\lambda}\left(f_0(x)+f_1(x)\log(x-\xi)+\cdots+f_K(x)\log(x-\xi)^K\right)$

If $L=P\,Q$ then $\quad \mathrm{expos}(L)/\mathbb{Z} \;=\; \mathrm{expos}(P)/\mathbb{Z} \;\sqcup\; \mathrm{expos}(Q)/\mathbb{Z}$ (as multisets)

Main criterion (Fuchsian case).

If $\mathrm{mult}(\lambda+\mathbb{Z})=1$ as an exponent class of L,

a) $\lambda+\mathbb{Z}\in\mathrm{Expo}$(proper right factor) — guess the factor by Hermite-Padé approx

b) or $\lambda+\mathbb{Z}\in\mathrm{Expo}$(proper left factor) — idem (adjoint)

c) or L is irreducible — exclude a) and b) using degree bounds

Eigenvalues of the **local** monodromy at ξ
$=\exp$(exponent classes)

($\Rightarrow$ the numeric Norton test "interpolates"
between van Hoeij's and van der Hoeven's method)

Numeric variant:

- can use other elements of $\mathbb{C}[G]$
- can test c) directly

Factoring: Implementation

[Goyer 2025]

```
sage: ((4*x^2 + 6*x + 2)*Dx^2 + (4*x + 3)*Dx - 1).factor()
sage: from ore_algebra.examples import polya
sage: polya.dop[9]  # cf. Beukers & Vlasenko 2021
sage: polya.dop[9].is_provably_irreducible()  # Maple DFactor: > 1 min 40 s
sage:
```

Factoring: Implementation

[Goyer 2025]

```
sage: ((4*x^2 + 6*x + 2)*Dx^2 + (4*x + 3)*Dx - 1).factor()
[(-2*x - 2)*Dx - 1, (-2*x - 1)*Dx + 1]
sage: from ore_algebra.examples import polya
sage: polya.dop[9]  # cf. Beukers & Vlasenko 2021
sage: polya.dop[9].is_provably_irreducible()  # Maple DFactor: > 1 min 40 s
sage:
```

Factoring: Implementation

[Goyer 2025]

```
sage: ((4*x^2 + 6*x + 2)*Dx^2 + (4*x + 3)*Dx - 1).factor()
[(-2*x - 2)*Dx - 1, (-2*x - 1)*Dx + 1]
sage: from ore_algebra.examples import polya
sage: polya.dop[9]  # cf. Beukers & Vlasenko 2021
(-914457600*z^18 + 270648576*z^16 - 11059840*z^14 + 140448*z^12 - 660*z^10 + z^8)*
Dz^9 + (-74071065600*z^17 + 19486697472*z^15 - 696769920*z^13 + 7584192*z^11 - 297
00*z^9 + 36*z^7)*Dz^8 + (-2370274099200*z^16 + 550176012288*z^14 - 17038986624*z^1
2 + 156601632*z^10 - 498696*z^8 + 462*z^6)*Dz^7 + (-387144769536000*z^15 + 78616610
79552*z^13 - 208388936448*z^11 + 1587838560*z^9 - 3984288*z^7 + 2646*z^5)*Dz^6 + (
-348430292582400*z^14 + 61302652637184*z^12 - 1371240707328*z^10 + 8466726048*z^8
 - 16052916*z^6 + 6951*z^4)*Dz^5 + (-1742151462912000*z^13 + 262594912849920*z^11 -
  4872727756800*z^9 + 23679448320*z^7 - 32006700*z^5 + 7770*z^3)*Dz^4 + (-464573723
4432000*z^12 + 592052526858240*z^10 - 8923673318400*z^8 + 32838948864*z^6 - 290544
76*z^4 + 3025*z^2)*Dz^3 + (-5973090729984000*z^11 + 633557545205760*z^9 - 75524219
59680*z^7 + 19955195136*z^5 - 10089816*z^3 + 255*z)*Dz^2 + (-2986545364992000*z^10
 + 258683438039040*z^8 - 2355318466560*z^6 + 4133152512*z^4 - 935592*z^2 + 1)*Dz -
 331838373888000*z^9 + 22924354682880*z^7 - 152023080960*z^5 + 156432384*z^3 - 921
6*z
sage: polya.dop[9].is_provably_irreducible()  # Maple DFactor: > 1 min 40 s
sage:
```

Factoring: Implementation

[Goyer 2025]

```
sage: ((4*x^2 + 6*x + 2)*Dx^2 + (4*x + 3)*Dx - 1).factor()
[(-2*x - 2)*Dx - 1, (-2*x - 1)*Dx + 1]
sage: from ore_algebra.examples import polya
sage: polya.dop[9]  # cf. Beukers & Vlasenko 2021
(-914457600*z^18 + 270648576*z^16 - 11059840*z^14 + 140448*z^12 - 660*z^10 + z^8)*
Dz^9 + (-74071065600*z^17 + 19486697472*z^15 - 696769920*z^13 + 7584192*z^11 - 297
00*z^9 + 36*z^7)*Dz^8 + (-2370274099200*z^16 + 550176012288*z^14 - 17038986624*z^1
2 + 156601632*z^10 - 498696*z^8 + 462*z^6)*Dz^7 + (-38714476953600*z^15 + 78616610
79552*z^13 - 208388936448*z^11 + 1587838560*z^9 - 3984288*z^7 + 2646*z^5)*Dz^6 + (
-348430292582400*z^14 + 61302652637184*z^12 - 1371240707328*z^10 + 8466726048*z^8
 - 16052916*z^6 + 6951*z^4)*Dz^5 + (-1742151462912000*z^13 + 262594912849920*z^11 -
  4872727756800*z^9 + 23679448320*z^7 - 32006700*z^5 + 7770*z^3)*Dz^4 + (-464573723
4432000*z^12 + 592052526858240*z^10 - 8923673318400*z^8 + 32838948864*z^6 - 290544
76*z^4 + 3025*z^2)*Dz^3 + (-5973090729984000*z^11 + 633557545205760*z^9 - 75524219
59680*z^7 + 19955195136*z^5 - 10089816*z^3 + 255*z)*Dz^2 + (-2986545364992000*z^10
 + 258683438039040*z^8 - 2355318466560*z^6 + 4133152512*z^4 - 935592*z^2 + 1)*Dz -
 331838373888000*z^9 + 229243546828 80*z^7 - 152023080960*z^5 + 156432384*z^3 - 921
6*z
sage: polya.dop[9].is_provably_irreducible()  # Maple DFactor: > 1 min 40 s
True
sage:
```

Other Problems

rational solution	fixed point
hyperexponential solution	invariant line (weight vector)
minimal annihilator	orbit
right-hand factor	invariant subspace (submodule)
complete factorization	Jordan-Hölder filtration
Loewy decomposition	socle filtration
eigenring	centralizer
absolute factor	virtually invariant subspace
…	…

Other Problems

rational solution	fixed point
hyperexponential solution	invariant line (weight vector)
minimal annihilator	orbit
right-hand factor	invariant subspace (submodule)
complete factorization	Jordan-Hölder filtration
Loewy decomposition	socle filtration
eigenring	centralizer
absolute factor	virtually invariant subspace
…	…

1. design **numerical algorithms** for the right column

   → linear algebra / classical computational group theory / …

Other Problems

rational solution	fixed point
hyperexponential solution	invariant line (weight vector)
minimal annihilator	orbit
right-hand factor	invariant subspace (submodule)
complete factorization	Jordan-Hölder filtration
Loewy decomposition	socle filtration
eigenring	centralizer
absolute factor	virtually invariant subspace
…	…

1. design **numerical algorithms** for the right column
   → linear algebra / classical computational group theory / …
2. what can be certified from **rigorous approximations**?
   → ≈ expressed as "open" conditions on the monodromy/Stokes matrices

Other Problems

rational solution	fixed point
hyperexponential solution	invariant line (weight vector)
minimal annihilator	orbit
right-hand factor	invariant subspace (submodule)
complete factorization	Jordan-Hölder filtration
Loewy decomposition	socle filtration
eigenring	centralizer
absolute factor	virtually invariant subspace
…	…

1. design **numerical algorithms** for the right column
   → linear algebra / classical computational group theory / …
2. what can be certified from **rigorous approximations**?
   → ≈ expressed as "open" conditions on the monodromy/Stokes matrices
3. what can be efficiently checked from **"guessed" reconstructions**?

Other Problems

rational solution	fixed point
hyperexponential solution	invariant line (weight vector)
minimal annihilator	orbit
right-hand factor	invariant subspace (submodule)
complete factorization	Jordan-Hölder filtration
Loewy decomposition	socle filtration
eigenring	centralizer
absolute factor	virtually invariant subspace
…	…

1. design **numerical algorithms** for the right column — useful on their own
   → linear algebra / classical computational group theory / …
2. what can be certified from **rigorous approximations**?
   → ≈ expressed as "open" conditions on the monodromy/Stokes matrices
3. what can be efficiently checked from **"guessed" reconstructions**?

Outlook

- A rigorous high-precision ODE solver
  with full support for regular singular points

  Comments, bugreports, examples… welcome
- ”Elementary” analytic Galois-theoretic characterizations are (quite) practical!

- Complete, even more efficient implementation of factorization
- Loewy decomposition [Goyer 2025]
- Algebraic, Liouvillian solutions

  More practical version of the Llorente–Mozo-Fernández algorithm?
- ”Implicit” proofs of algebraicity (à la Barkatou–Cluzeau–Di Vizio–Weil), transcendence