# Computing divisors and common multiples of quasi-linear ordinary differential equations (jointly with F. Schwarz)

Dima Grigoriev (Lille)

CNRS

10/09/2012, Berlin

# Factoring ordinary differential equations

## Definition

Ordinary differential equation $f(y^{(k)}, \ldots, y', y, x) = 0$ is a *generalized factor* of $g(y^{(n)}, \ldots, y', y, x) = 0$ if any solution of the former is a solution of the latter.

## Factoring linear operators

A linear operator $A = \sum_{0 \le i \le k} a_i \cdot y^{(i)}$ is a (generalized) factor of a linear operator $B = \sum_{0 \le j \le n} b_j \cdot y^{(j)}$ iff $B = CA$ for a suitable linear operator $C$. Factoring of a linear operator is not unique. Thus, the problem is to produce some factoring into irreducible factors.

• Beke-Schlesinger (1894): An algorithm for factoring linear operators $A$ with coefficients $a_i \in \overline{\mathbb{Q}}(x)$ (triple-exponential complexity);

• G. (1987): An algorithm with double-exponential complexity. Conjecture: the sharp bound of complexity is exponential.;

• Tsarev (1996): An algorithm to describe the variety of all the factorizations of an operator.

# Factoring ordinary differential equations

## Definition

Ordinary differential equation $f(y^{(k)}, \dots, y', y, x) = 0$ is a *generalized factor* of $g(y^{(n)}, \dots, y', y, x) = 0$ if any solution of the former is a solution of the latter.

## Factoring linear operators

A linear operator $A = \sum_{0 \le i \le k} a_i \cdot y^{(i)}$ is a (generalized) factor of a linear operator $B = \sum_{0 \le j \le n} b_j \cdot y^{(j)}$ iff $B = CA$ for a suitable linear operator $C$.

Factoring of a linear operator is not unique. Thus, the problem is to produce some factoring into irreducible factors.

- Beke-Schlesinger (1894): An algorithm for factoring linear operators $A$ with coefficients $a_i \in \overline{\mathbb{Q}}(x)$ (triple-exponential complexity);
- G. (1987): An algorithm with double-exponential complexity. Conjecture: the sharp bound of complexity is exponential.;
- Tsarev (1996): An algorithm to describe the variety of all the factorizations of an operator.

# Factoring ordinary differential equations

## Definition

Ordinary differential equation $f(y^{(k)}, \dots, y', y, x) = 0$ is a *generalized factor* of $g(y^{(n)}, \dots, y', y, x) = 0$ if any solution of the former is a solution of the latter.

## Factoring linear operators

A linear operator $A = \sum_{0 \le i \le k} a_i \cdot y^{(i)}$ is a (generalized) factor of a linear operator $B = \sum_{0 \le j \le n} b_j \cdot y^{(j)}$ iff $B = CA$ for a suitable linear operator $C$. Factoring of a linear operator is not unique. Thus, the problem is to produce some factoring into irreducible factors.

- Beke-Schlesinger (1894): An algorithm for factoring linear operators $A$ with coefficients $a_i \in \overline{\mathbb{Q}}(x)$ (triple-exponential complexity);
- G. (1987): An algorithm with double-exponential complexity. Conjecture: the sharp bound of complexity is exponential.;
- Tsarev (1996): An algorithm to describe the variety of all the factorizations of an operator.

# Factoring ordinary differential equations

## Definition

Ordinary differential equation $f(y^{(k)}, \dots, y', y, x) = 0$ is a *generalized factor* of $g(y^{(n)}, \dots, y', y, x) = 0$ if any solution of the former is a solution of the latter.

## Factoring linear operators

A linear operator $A = \sum_{0 \le i \le k} a_i \cdot y^{(i)}$ is a (generalized) factor of a linear operator $B = \sum_{0 \le j \le n} b_j \cdot y^{(j)}$ iff $B = CA$ for a suitable linear operator $C$. Factoring of a linear operator is not unique. Thus, the problem is to produce some factoring into irreducible factors.

- Beke-Schlesinger (1894): An algorithm for factoring linear operators $A$ with coefficients $a_i \in \overline{\mathbb{Q}}(x)$ (triple-exponential complexity);
- G. (1987): An algorithm with double-exponential complexity. Conjecture: the sharp bound of complexity is exponential.;
- Tsarev (1996): An algorithm to describe the variety of all the factorizations of an operator.

# Factoring ordinary differential equations

## Definition

Ordinary differential equation $f(y^{(k)}, \dots, y', y, x) = 0$ is a *generalized factor* of $g(y^{(n)}, \dots, y', y, x) = 0$ if any solution of the former is a solution of the latter.

## Factoring linear operators

A linear operator $A = \sum_{0 \le i \le k} a_i \cdot y^{(i)}$ is a (generalized) factor of a linear operator $B = \sum_{0 \le j \le n} b_j \cdot y^{(j)}$ iff $B = CA$ for a suitable linear operator $C$. Factoring of a linear operator is not unique. Thus, the problem is to produce some factoring into irreducible factors.

• Beke-Schlesinger (1894): An algorithm for factoring linear operators $A$ with coefficients $a_i \in \overline{\mathbb{Q}}(x)$ (triple-exponential complexity);

• G. (1987): An algorithm with double-exponential complexity. Conjecture: the sharp bound of complexity is exponential.;

• Tsarev (1996): An algorithm to describe the variety of all the factorizations of an operator.

# Factoring ordinary differential equations

## Definition

Ordinary differential equation $f(y^{(k)}, \dots, y', y, x) = 0$ is a *generalized factor* of $g(y^{(n)}, \dots, y', y, x) = 0$ if any solution of the former is a solution of the latter.

## Factoring linear operators

A linear operator $A = \sum_{0 \le i \le k} a_i \cdot y^{(i)}$ is a (generalized) factor of a linear operator $B = \sum_{0 \le j \le n} b_j \cdot y^{(j)}$ iff $B = CA$ for a suitable linear operator $C$. Factoring of a linear operator is not unique. Thus, the problem is to produce some factoring into irreducible factors.

• Beke-Schlesinger (1894): An algorithm for factoring linear operators $A$ with coefficients $a_i \in \overline{\mathbb{Q}}(x)$ (triple-exponential complexity);

• G. (1987): An algorithm with double-exponential complexity.
Conjecture: the sharp bound of complexity is exponential.;

• Tsarev (1996): An algorithm to describe the variety of all the factorizations of an operator.

# Factoring ordinary differential equations

## Definition

Ordinary differential equation $f(y^{(k)}, \dots, y', y, x) = 0$ is a *generalized factor* of $g(y^{(n)}, \dots, y', y, x) = 0$ if any solution of the former is a solution of the latter.

## Factoring linear operators

A linear operator $A = \sum_{0 \le i \le k} a_i \cdot y^{(i)}$ is a (generalized) factor of a linear operator $B = \sum_{0 \le j \le n} b_j \cdot y^{(j)}$ iff $B = CA$ for a suitable linear operator $C$. Factoring of a linear operator is not unique. Thus, the problem is to produce some factoring into irreducible factors.

• Beke-Schlesinger (1894): An algorithm for factoring linear operators $A$ with coefficients $a_i \in \overline{\mathbb{Q}}(x)$ (triple-exponential complexity);

• G. (1987): An algorithm with double-exponential complexity. Conjecture: the sharp bound of complexity is exponential.;

• Tsarev (1996): An algorithm to describe the variety of all the factorizations of an operator.

# Factoring ordinary differential equations

## Definition

Ordinary differential equation $f(y^{(k)}, \dots, y', y, x) = 0$ is a *generalized factor* of $g(y^{(n)}, \dots, y', y, x) = 0$ if any solution of the former is a solution of the latter.

## Factoring linear operators

A linear operator $A = \sum_{0 \le i \le k} a_i \cdot y^{(i)}$ is a (generalized) factor of a linear operator $B = \sum_{0 \le j \le n} b_j \cdot y^{(j)}$ iff $B = CA$ for a suitable linear operator $C$. Factoring of a linear operator is not unique. Thus, the problem is to produce some factoring into irreducible factors.

• Beke-Schlesinger (1894): An algorithm for factoring linear operators $A$ with coefficients $a_i \in \overline{\mathbb{Q}}(x)$ (triple-exponential complexity);

• G. (1987): An algorithm with double-exponential complexity. Conjecture: the sharp bound of complexity is exponential.;

• Tsarev (1996): An algorithm to describe the variety of all the factorizations of an operator.

# Quasi-linear generalized factors

The algebra of differential polynomials $\mathbb{C}\{y\} := \mathbb{C}[x, y, y', y'', \ldots]$ is a module over the algebra $\mathbb{C}\{y\}[\frac{d}{dx}]$ of linear operators with coefficients in $\mathbb{C}\{y\}$. For a differential polynomial $p \in \mathbb{C}\{y\}$ and an operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ their action denote by $H * p \in \mathbb{C}\{y\}$.

**Lemma**

*A quasi-linear differential polynomial $y^{(k+1)} - f(y^{(k)}, \ldots, y', y, x)$ is a generalized factor of a differential polynomial p iff there exists a linear operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ such that $H * (y^{(k+1)} - f(y^{(k)}, \ldots, y', y, x)) = p$*

Tsarev (1999), Gao-Zhang (2008) studied another concept of decomposition of differential polynomials when in a differential polynomial $p$ (depending on several variables $y_1, \ldots, y_m$) some differential polynomials $p_1, \ldots, p_m$ are substituted. This decomposition differs from our notion of factoring. Tsarev, Gao-Zhang have designed algorithms to decompose differential polynomials.

# Quasi-linear generalized factors

The algebra of differential polynomials $\mathbb{C}\{y\} := \mathbb{C}[x, y, y', y'', \ldots]$ is a module over the algebra $\mathbb{C}\{y\}[\frac{d}{dx}]$ of linear operators with coefficients in $\mathbb{C}\{y\}$. For a differential polynomial $p \in \mathbb{C}\{y\}$ and an operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ their action denote by $H * p \in \mathbb{C}\{y\}$.

**Lemma**

*A quasi-linear differential polynomial $y^{(k+1)} - f(y^{(k)}, \ldots, y', y, x)$ is a generalized factor of a differential polynomial p iff there exists a linear operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ such that $H * (y^{(k+1)} - f(y^{(k)}, \ldots, y', y, x)) = p$*

Tsarev (1999), Gao-Zhang (2008) studied another concept of decomposition of differential polynomials when in a differential polynomial $p$ (depending on several variables $y_1, \ldots, y_m$) some differential polynomials $p_1, \ldots, p_m$ are substituted. This decomposition differs from our notion of factoring. Tsarev, Gao-Zhang have designed algorithms to decompose differential polynomials.

# Quasi-linear generalized factors

The algebra of differential polynomials $\mathbb{C}\{y\} := \mathbb{C}[x, y, y', y'', \dots]$ is a module over the algebra $\mathbb{C}\{y\}[\frac{d}{dx}]$ of linear operators with coefficients in $\mathbb{C}\{y\}$. For a differential polynomial $p \in \mathbb{C}\{y\}$ and an operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ their action denote by $H * p \in \mathbb{C}\{y\}$.

**Lemma**

*A quasi-linear differential polynomial $y^{(k+1)} - f(y^{(k)}, \dots, y', y, x)$ is a generalized factor of a differential polynomial p iff there exists a linear operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ such that $H * (y^{(k+1)} - f(y^{(k)}, \dots, y', y, x)) = p$*

Tsarev (1999), Gao-Zhang (2008) studied another concept of decomposition of differential polynomials when in a differential polynomial $p$ (depending on several variables $y_1, \dots, y_m$) some differential polynomials $p_1, \dots, p_m$ are substituted. This decomposition differs from our notion of factoring. Tsarev, Gao-Zhang have designed algorithms to decompose differential polynomials.

# Quasi-linear generalized factors

The algebra of differential polynomials $\mathbb{C}\{y\} := \mathbb{C}[x, y, y', y'', \dots]$ is a module over the algebra $\mathbb{C}\{y\}[\frac{d}{dx}]$ of linear operators with coefficients in $\mathbb{C}\{y\}$. For a differential polynomial $p \in \mathbb{C}\{y\}$ and an operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ their action denote by $H * p \in \mathbb{C}\{y\}$.

**Lemma**

*A quasi-linear differential polynomial $y^{(k+1)} - f(y^{(k)}, \dots, y', y, x)$ is a generalized factor of a differential polynomial $p$ iff there exists a linear operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ such that $H * (y^{(k+1)} - f(y^{(k)}, \dots, y', y, x)) = p$*

Tsarev (1999), Gao-Zhang (2008) studied another concept of decomposition of differential polynomials when in a differential polynomial $p$ (depending on several variables $y_1, \dots, y_m$) some differential polynomials $p_1, \dots, p_m$ are substituted. This decomposition differs from our notion of factoring. Tsarev, Gao-Zhang have designed algorithms to decompose differential polynomials.

# Quasi-linear generalized factors

The algebra of differential polynomials $\mathbb{C}\{y\} := \mathbb{C}[x, y, y', y'', \dots]$ is a module over the algebra $\mathbb{C}\{y\}[\frac{d}{dx}]$ of linear operators with coefficients in $\mathbb{C}\{y\}$. For a differential polynomial $p \in \mathbb{C}\{y\}$ and an operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ their action denote by $H * p \in \mathbb{C}\{y\}$.

**Lemma**

*A quasi-linear differential polynomial $y^{(k+1)} - f(y^{(k)}, \dots, y', y, x)$ is a generalized factor of a differential polynomial $p$ iff there exists a linear operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ such that $H * (y^{(k+1)} - f(y^{(k)}, \dots, y', y, x)) = p$*

Tsarev (1999), Gao-Zhang (2008) studied another concept of decomposition of differential polynomials when in a differential polynomial $p$ (depending on several variables $y_1, \dots, y_m$) some differential polynomials $p_1, \dots, p_m$ are substituted. This decomposition differs from our notion of factoring. Tsarev, Gao-Zhang have designed algorithms to decompose differential polynomials.

# Quasi-linear generalized factors

The algebra of differential polynomials $\mathbb{C}\{y\} := \mathbb{C}[x, y, y', y'', \dots]$ is a module over the algebra $\mathbb{C}\{y\}[\frac{d}{dx}]$ of linear operators with coefficients in $\mathbb{C}\{y\}$. For a differential polynomial $p \in \mathbb{C}\{y\}$ and an operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ their action denote by $H * p \in \mathbb{C}\{y\}$.

**Lemma**

*A quasi-linear differential polynomial $y^{(k+1)} - f(y^{(k)}, \dots, y', y, x)$ is a generalized factor of a differential polynomial $p$ iff there exists a linear operator $H \in \mathbb{C}\{y\}[\frac{d}{dx}]$ such that $H * (y^{(k+1)} - f(y^{(k)}, \dots, y', y, x)) = p$*

Tsarev (1999), Gao-Zhang (2008) studied another concept of decomposition of differential polynomials when in a differential polynomial $p$ (depending on several variables $y_1, \dots, y_m$) some differential polynomials $p_1, \dots, p_m$ are substituted. This decomposition differs from our notion of factoring. Tsarev, Gao-Zhang have designed algorithms to decompose differential polynomials.

# Factoring a quasi-linear second-order equation

## Theorem

*1) If a first-order quasi-linear differential polynomial $y' - p(y,x)$ is a generalized factor of a second-order quasi-linear differential polynomial $y'' - f(y', y, x)$ for polynomials $p(y,x) \in \overline{\mathbb{Q}}[y,x]$, $f(z,y,x) = \sum_{0 \le i \le l} f_i \cdot (y')^i \in \overline{\mathbb{Q}}[z,y,x]$, then $\deg_x(p) \le \max\{\deg_x(f), 1+\deg_x(f_0)\}$, $\deg_y(p) \le \max\{\deg_y(f), \deg_y(f_1)\}$.*

*2) An algorithm is designed which for a second-order quasi-linear differential polynomial $y'' - f(y', y, x)$ either produces some its first-order generalized divisor $y' - p(y,x)$ satisfying the bounds from 1) or certifies that it does not exist.*

The algorithm from 2) solves a system of polynomial equations in the indeterminate coefficients of polynomial $p$ resulting from the equality

$\frac{\partial p}{\partial y} \cdot p + \frac{\partial p}{\partial x} = \sum_{0 \le i \le l} f_i \cdot p^i$

which is equivalent to $y' - p(y,x)$ being a generalized factor of $y'' - f(y', y, x)$. Also from this equality one deduces 1) making use of the relation $p | (f_0 - \frac{\partial p}{\partial x})$.

# Factoring a quasi-linear second-order equation

## Theorem

*1) If a first-order quasi-linear differential polynomial $y' - p(y,x)$ is a generalized factor of a second-order quasi-linear differential polynomial $y'' - f(y',y,x)$ for polynomials $p(y,x) \in \overline{\mathbb{Q}}[y,x]$, $f(z,y,x) = \sum_{0 \le i \le l} f_i \cdot (y')^i \in \overline{\mathbb{Q}}[z,y,x]$, then $\deg_x(p) \le \max\{\deg_x(f), 1+\deg_x(f_0)\}$, $\deg_y(p) \le \max\{\deg_y(f), \deg_y(f_1)\}$.*

*2) An algorithm is designed which for a second-order quasi-linear differential polynomial $y'' - f(y',y,x)$ either produces some its first-order generalized divisor $y' - p(y,x)$ satisfying the bounds from 1) or certifies that it does not exist.*

The algorithm from 2) solves a system of polynomial equations in the indeterminate coefficients of polynomial $p$ resulting from the equality

$$\frac{\partial p}{\partial y} \cdot p + \frac{\partial p}{\partial x} = \sum_{0 \le i \le l} f_i \cdot p^i$$

which is equivalent to $y' - p(y,x)$ being a generalized factor of $y'' - f(y',y,x)$. Also from this equality one deduces 1) making use of the relation $p | (f_0 - \frac{\partial p}{\partial x})$.

# Factoring a quasi-linear second-order equation

**Theorem**

*1) If a first-order quasi-linear differential polynomial $y' - p(y,x)$ is a generalized factor of a second-order quasi-linear differential polynomial $y'' - f(y', y, x)$ for polynomials $p(y,x) \in \overline{\mathbb{Q}}[y,x]$, $f(z,y,x) = \sum_{0 \le i \le l} f_i \cdot (y')^i \in \overline{\mathbb{Q}}[z,y,x]$, then $\deg_x(p) \le \max\{\deg_x(f), 1 + \deg_x(f_0)\}$, $\deg_y(p) \le \max\{\deg_y(f), \deg_y(f_1)\}$.*
*2) An algorithm is designed which for a second-order quasi-linear differential polynomial $y'' - f(y', y, x)$ either produces some its first-order generalized divisor $y' - p(y,x)$ satisfying the bounds from 1) or certifies that it does not exist.*

The algorithm from 2) solves a system of polynomial equations in the indeterminate coefficients of polynomial $p$ resulting from the equality
$\frac{\partial p}{\partial y} \cdot p + \frac{\partial p}{\partial x} = \sum_{0 \le i \le l} f_i \cdot p^i$
which is equivalent to $y' - p(y,x)$ being a generalized factor of $y'' - f(y', y, x)$. Also from this equality one deduces 1) making use of the relation $p | (f_0 - \frac{\partial p}{\partial x})$.

# Factoring a quasi-linear second-order equation

## Theorem

*1) If a first-order quasi-linear differential polynomial $y' - p(y,x)$ is a generalized factor of a second-order quasi-linear differential polynomial $y'' - f(y',y,x)$ for polynomials $p(y,x) \in \overline{\mathbb{Q}}[y,x]$, $f(z,y,x) = \sum_{0 \le i \le l} f_i \cdot (y')^i \in \overline{\mathbb{Q}}[z,y,x]$, then $\deg_x(p) \le \max\{\deg_x(f), 1 + \deg_x(f_0)\}$, $\deg_y(p) \le \max\{\deg_y(f), \deg_y(f_1)\}$.*
*2) An algorithm is designed which for a second-order quasi-linear differential polynomial $y'' - f(y',y,x)$ either produces some its first-order generalized divisor $y' - p(y,x)$ satisfying the bounds from 1) or certifies that it does not exist.*

The algorithm from 2) solves a system of polynomial equations in the indeterminate coefficients of polynomial $p$ resulting from the equality
$\frac{\partial p}{\partial y} \cdot p + \frac{\partial p}{\partial x} = \sum_{0 \le i \le l} f_i \cdot p^i$
which is equivalent to $y' - p(y,x)$ being a generalized factor of $y'' - f(y',y,x)$. Also from this equality one deduces 1) making use of the relation $p | (f_0 - \frac{\partial p}{\partial x})$.

# Factoring a quasi-linear second-order equation

**Theorem**

*1) If a first-order quasi-linear differential polynomial $y' - p(y,x)$ is a generalized factor of a second-order quasi-linear differential polynomial $y'' - f(y', y, x)$ for polynomials $p(y,x) \in \overline{\mathbb{Q}}[y,x]$, $f(z,y,x) = \sum_{0 \le i \le l} f_i \cdot (y')^i \in \overline{\mathbb{Q}}[z,y,x]$, then $\deg_x(p) \le \max\{\deg_x(f), 1 + \deg_x(f_0)\}$, $\deg_y(p) \le \max\{\deg_y(f), \deg_y(f_1)\}$.*
*2) An algorithm is designed which for a second-order quasi-linear differential polynomial $y'' - f(y', y, x)$ either produces some its first-order generalized divisor $y' - p(y,x)$ satisfying the bounds from 1) or certifies that it does not exist.*

The algorithm from 2) solves a system of polynomial equations in the indeterminate coefficients of polynomial $p$ resulting from the equality
$\frac{\partial p}{\partial y} \cdot p + \frac{\partial p}{\partial x} = \sum_{0 \le i \le l} f_i \cdot p^i$
which is equivalent to $y' - p(y,x)$ being a generalized factor of $y'' - f(y', y, x)$. Also from this equality one deduces 1) making use of the relation $p | (f_0 - \frac{\partial p}{\partial x})$.

# Factoring a quasi-linear second-order equation

**Theorem**

*1) If a first-order quasi-linear differential polynomial $y' - p(y,x)$ is a generalized factor of a second-order quasi-linear differential polynomial $y'' - f(y',y,x)$ for polynomials $p(y,x) \in \overline{\mathbb{Q}}[y,x]$, $f(z,y,x) = \sum_{0 \le i \le l} f_i \cdot (y')^i \in \overline{\mathbb{Q}}[z,y,x]$, then $\deg_x(p) \le \max\{\deg_x(f), 1+\deg_x(f_0)\}$, $\deg_y(p) \le \max\{\deg_y(f), \deg_y(f_1)\}$.*
*2) An algorithm is designed which for a second-order quasi-linear differential polynomial $y'' - f(y',y,x)$ either produces some its first-order generalized divisor $y' - p(y,x)$ satisfying the bounds from 1) or certifies that it does not exist.*

The algorithm from 2) solves a system of polynomial equations in the indeterminate coefficients of polynomial $p$ resulting from the equality
$\frac{\partial p}{\partial y} \cdot p + \frac{\partial p}{\partial x} = \sum_{0 \le i \le l} f_i \cdot p^i$
which is equivalent to $y' - p(y,x)$ being a generalized factor of $y'' - f(y',y,x)$. Also from this equality one deduces 1) making use of the relation $p | (f_0 - \frac{\partial p}{\partial x})$.

It would be interesting to extend the factoring algorithm from the second to an arbitrary order and from quasi-linear to arbitrary equations (perhaps, also from ordinary to *partial* differential equations).

**Example**

Consider the equation

$$E \equiv y'' + (x+3y)y' + y^3 + xy^2 = 0.$$

According to the above Theorem 1) $\deg_x E \leq 1$ and $\deg_y E \leq 3$. Applying Theorem 2) two factors are obtained and the representations

$$E \equiv (y'+y^2)' + (y+x)(y'+y^2), \quad E = (y'+y^2+xy-1)' + y(y'+y^2+xy-1)$$

follow. They yield the two one-parameter solutions

$$y = \frac{1}{x+C}, \quad y = \frac{1}{x} + \frac{1}{x^2}\frac{\exp\left(-\frac{1}{2}x^2\right)}{\int \exp\left(-\frac{1}{2}x^2\right)\frac{dx}{x^2} + C}$$

respectively.

It would be interesting to extend the factoring algorithm from the second to an arbitrary order and from quasi-linear to arbitrary equations (perhaps, also from ordinary to *partial* differential equations).

**Example**

Consider the equation

$$E \equiv y'' + (x + 3y)y' + y^3 + xy^2 = 0.$$

According to the above Theorem 1) $\deg_x E \leq 1$ and $\deg_y E \leq 3$.
Applying Theorem 2) two factors are obtained and the representations

$$E \equiv (y'+y^2)'+(y+x)(y'+y^2), \quad E = (y'+y^2+xy-1)'+y(y'+y^2+xy-1)$$

follow. They yield the two one-parameter solutions

$$y = \frac{1}{x + C}, \quad y = \frac{1}{x} + \frac{1}{x^2} \frac{\exp\left(-\frac{1}{2}x^2\right)}{\int \exp\left(-\frac{1}{2}x^2\right)\frac{dx}{x^2} + C}$$

respectively.

It would be interesting to extend the factoring algorithm from the second to an arbitrary order and from quasi-linear to arbitrary equations (perhaps, also from ordinary to *partial* differential equations).

**Example**

Consider the equation

$$E \equiv y'' + (x + 3y)y' + y^3 + xy^2 = 0.$$

According to the above Theorem 1) $\deg_x E \leq 1$ and $\deg_y E \leq 3$. Applying Theorem 2) two factors are obtained and the representations

$$E \equiv (y' + y^2)' + (y + x)(y' + y^2), \quad E = (y' + y^2 + xy - 1)' + y(y' + y^2 + xy - 1)$$

follow. They yield the two one-parameter solutions

$$y = \frac{1}{x + C}, \quad y = \frac{1}{x} + \frac{1}{x^2} \frac{\exp\left(-\frac{1}{2}x^2\right)}{\int \exp\left(-\frac{1}{2}x^2\right)\frac{dx}{x^2} + C}$$

respectively.

It would be interesting to extend the factoring algorithm from the second to an arbitrary order and from quasi-linear to arbitrary equations (perhaps, also from ordinary to *partial* differential equations).

**Example**

Consider the equation

$$E \equiv y'' + (x + 3y)y' + y^3 + xy^2 = 0.$$

According to the above Theorem 1) $\deg_x E \leq 1$ and $\deg_y E \leq 3$. Applying Theorem 2) two factors are obtained and the representations

$$E \equiv (y'+y^2)'+(y+x)(y'+y^2), \quad E = (y'+y^2+xy-1)'+y(y'+y^2+xy-1)$$

follow. They yield the two one-parameter solutions

$$y = \frac{1}{x+C}, \quad y = \frac{1}{x} + \frac{1}{x^2}\frac{\exp\left(-\frac{1}{2}x^2\right)}{\int \exp\left(-\frac{1}{2}x^2\right)\frac{dx}{x^2} + C}$$

respectively.

# Quasi-linear common multiples

We say that a differential polynomial $f \in \mathbb{C}\{y\}$ is a *common multiple* of differential polynomials $f_1,\ f_2$ if solutions of $f = 0$ contain solutions of both $f_1 = 0$ and $f_2 = 0$.

**Lemma**

*For polynomials $p, q \in \mathbb{Q}[y, x]$ there exists a quasi-linear common multiple of the order $n + 1$ of a pair of first-order equations $y' = p(y, x),\ y' = q(y, x)$ iff n-th derivative $(p - q)^{(n)} \in \langle p - q, (p - q)^{(1)}, \dots, (p - q)^{(n-1)} \rangle$ belongs to the ideal (in the algebra $\mathbb{Q}[y, x]$) generated by first $n - 1$ derivatives of $p - q$.*

*More explicitly, if the latter relation holds, i. e. $(p - q)^{(n)} = \sum_{0 \le i < n} r_i \cdot (p - q)^{(i)}$ for some polynomials $r_i \in \mathbb{Q}[y, x]$, $0 \le i < n$ then for polynomial $s_n(z_n, \dots, z_1, y, x) := \sum_{0 \le i < n} r_i \cdot (z_{i+1} - p^{(i)}) + p^{(n)} = \sum_{0 \le i < n} r_i \cdot (z_{i+1} - q^{(i)}) + q^{(n)}$ equation $y^{(n+1)} = s_n(y^{(n)}, \dots, y', y, x)$ is a required quasi-linear common multiple.*

# Quasi-linear common multiples

We say that a differential polynomial $f \in \mathbb{C}\{y\}$ is a *common multiple* of differential polynomials $f_1,\ f_2$ if solutions of $f = 0$ contain solutions of both $f_1 = 0$ and $f_2 = 0$.

## Lemma

*For polynomials $p, q \in \mathbb{Q}[y, x]$ there exists a quasi-linear common multiple of the order $n + 1$ of a pair of first-order equations $y' = p(y, x),\ y' = q(y, x)$ iff n-th derivative $(p - q)^{(n)} \in \langle p - q,\ (p - q)^{(1)}, \dots, (p - q)^{(n-1)} \rangle$ belongs to the ideal (in the algebra $\mathbb{Q}[y, x]$) generated by first $n - 1$ derivatives of $p - q$.*

*More explicitly, if the latter relation holds, i. e. $(p - q)^{(n)} = \sum_{0 \le i < n} r_i \cdot (p - q)^{(i)}$ for some polynomials $r_i \in \mathbb{Q}[y, x],\ 0 \le i < n$ then for polynomial $s_n(z_n, \dots, z_1, y, x) := \sum_{0 \le i < n} r_i \cdot (z_{i+1} - p^{(i)}) + p^{(n)} = \sum_{0 \le i < n} r_i \cdot (z_{i+1} - q^{(i)}) + q^{(n)}$ equation $y^{(n+1)} = s_n(y^{(n)}, \dots, y', y, x)$ is a required quasi-linear common multiple.*

# Quasi-linear common multiples

We say that a differential polynomial $f \in \mathbb{C}\{y\}$ is a *common multiple* of differential polynomials $f_1,\ f_2$ if solutions of $f = 0$ contain solutions of both $f_1 = 0$ and $f_2 = 0$.

**Lemma**

*For polynomials $p, q \in \mathbb{Q}[y, x]$ there exists a quasi-linear common multiple of the order $n + 1$ of a pair of first-order equations $y' = p(y, x),\ y' = q(y, x)$ iff n-th derivative $(p - q)^{(n)} \in \langle p - q,\ (p - q)^{(1)}, \ldots, (p - q)^{(n-1)}\rangle$ belongs to the ideal (in the algebra $\mathbb{Q}[y, x]$) generated by first $n - 1$ derivatives of $p - q$.*

*More explicitly, if the latter relation holds, i. e. $(p - q)^{(n)} = \sum_{0 \le i < n} r_i \cdot (p - q)^{(i)}$ for some polynomials $r_i \in \mathbb{Q}[y, x],\ 0 \le i < n$ then for polynomial $s_n(z_n, \ldots, z_1, y, x) := \sum_{0 \le i < n} r_i \cdot (z_{i+1} - p^{(i)}) + p^{(n)} = \sum_{0 \le i < n} r_i \cdot (z_{i+1} - q^{(i)}) + q^{(n)}$ equation $y^{(n+1)} = s_n(y^{(n)}, \ldots, y', y, x)$ is a required quasi-linear common multiple.*

# Quasi-linear common multiples

We say that a differential polynomial $f \in \mathbb{C}\{y\}$ is a *common multiple* of differential polynomials $f_1, f_2$ if solutions of $f = 0$ contain solutions of both $f_1 = 0$ and $f_2 = 0$.

## Lemma

*For polynomials $p, q \in \mathbb{Q}[y, x]$ there exists a quasi-linear common multiple of the order $n + 1$ of a pair of first-order equations $y' = p(y, x)$, $y' = q(y, x)$ iff n-th derivative $(p - q)^{(n)} \in \langle p - q, (p - q)^{(1)}, \dots, (p - q)^{(n-1)} \rangle$ belongs to the ideal (in the algebra $\mathbb{Q}[y, x]$) generated by first $n - 1$ derivatives of $p - q$.*

*More explicitly, if the latter relation holds, i. e. $(p - q)^{(n)} = \sum_{0 \le i < n} r_i \cdot (p - q)^{(i)}$ for some polynomials $r_i \in \mathbb{Q}[y, x]$, $0 \le i < n$ then for polynomial $s_n(z_n, \dots, z_1, y, x) := \sum_{0 \le i < n} r_i \cdot (z_{i+1} - p^{(i)}) + p^{(n)} = \sum_{0 \le i < n} r_i \cdot (z_{i+1} - q^{(i)}) + q^{(n)}$ equation $y^{(n+1)} = s_n(y^{(n)}, \dots, y', y, x)$ is a required quasi-linear common multiple.*

# Quasi-linear common multiples

We say that a differential polynomial $f \in \mathbb{C}\{y\}$ is a *common multiple* of differential polynomials $f_1,\ f_2$ if solutions of $f = 0$ contain solutions of both $f_1 = 0$ and $f_2 = 0$.

**Lemma**

*For polynomials $p, q \in \mathbb{Q}[y, x]$ there exists a quasi-linear common multiple of the order $n + 1$ of a pair of first-order equations $y' = p(y, x),\ y' = q(y, x)$ iff n-th derivative $(p - q)^{(n)} \in \langle p - q,\ (p - q)^{(1)}, \dots, (p - q)^{(n-1)}\rangle$ belongs to the ideal (in the algebra $\mathbb{Q}[y, x]$) generated by first $n - 1$ derivatives of $p - q$.*

*More explicitly, if the latter relation holds, i. e. $(p - q)^{(n)} = \sum_{0 \le i < n} r_i \cdot (p - q)^{(i)}$ for some polynomials $r_i \in \mathbb{Q}[y, x]$, $0 \le i < n$ then for polynomial $s_n(z_n, \dots, z_1, y, x) := \sum_{0 \le i < n} r_i \cdot (z_{i+1} - p^{(i)}) + p^{(n)} = \sum_{0 \le i < n} r_i \cdot (z_{i+1} - q^{(i)}) + q^{(n)}$ equation $y^{(n+1)} = s_n(y^{(n)}, \dots, y', y, x)$ is a required quasi-linear common multiple.*

# Existence of a quasi-linear common multiple

One can directly extend the lemma to a quasi-linear common multiple of a pair of quasi-linear equations of an arbitrary order.

Employing Hilbert's Idealbasissatz we obtain

**Corollary**

*Any pair of ordinary quasi-linear differential equations has a quasi-linear common multiple.*

To formulate the complexity bound of an algorithm computing a quasi-linear common multiple we need to recall Grzegorczyk's classes of primitive-recursive function

# Existence of a quasi-linear common multiple

One can directly extend the lemma to a quasi-linear common multiple of a pair of quasi-linear equations of an arbitrary order.

Employing Hilbert's Idealbasissatz we obtain

**Corollary**

*Any pair of ordinary quasi-linear differential equations has a quasi-linear common multiple.*

To formulate the complexity bound of an algorithm computing a quasi-linear common multiple we need to recall Grzegorczyk's classes of primitive-recursive function

# Existence of a quasi-linear common multiple

One can directly extend the lemma to a quasi-linear common multiple of a pair of quasi-linear equations of an arbitrary order.

Employing Hilbert's Idealbasissatz we obtain

**Corollary**

*Any pair of ordinary quasi-linear differential equations has a quasi-linear common multiple.*

To formulate the complexity bound of an algorithm computing a quasi-linear common multiple we need to recall Grzegorczyk's classes of primitive-recursive function

## Grzegorczyk's classes of primitive-recursive functions

$\mathcal{E}^l$ consist of functions $\mathbb{Z}^s \to \mathbb{Z}^t$.

For the base of recursion

class $\mathcal{E}^0$ contains

- constant functions $x_k \to c$,
- shifts $x_k \to x_k + c$,
- projections $(x_1, \dots, x_n) \to x_k$;

class $\mathcal{E}^1$ contains linear functions $x_k \to c \cdot x_k$ and $(x_{k_1}, x_{k_2}) \to x_{k_1} + x_{k_2}$;

class $\mathcal{E}^2$ contains all the polynomials with integer coefficients.

## Grzegorczyk's classes of primitive-recursive functions

$\mathcal{E}^l$ consist of functions $\mathbb{Z}^s \to \mathbb{Z}^t$.

For the base of recursion

class $\mathcal{E}^0$ contains

• constant functions $x_k \to c$,

• shifts $x_k \to x_k + c$,

• projections $(x_1, \ldots, x_n) \to x_k$;

class $\mathcal{E}^1$ contains linear functions $x_k \to c \cdot x_k$ and $(x_{k_1}, x_{k_2}) \to x_{k_1} + x_{k_2}$;

class $\mathcal{E}^2$ contains all the polynomials with integer coefficients.

## Grzegorczyk's classes of primitive-recursive functions

$\mathcal{E}^l$ consist of functions $\mathbb{Z}^s \to \mathbb{Z}^t$.

For the base of recursion

class $\mathcal{E}^0$ contains

- constant functions $x_k \to c$,
- shifts $x_k \to x_k + c$,
- projections $(x_1, \dots, x_n) \to x_k$;

class $\mathcal{E}^1$ contains linear functions $x_k \to c \cdot x_k$ and $(x_{k_1}, x_{k_2}) \to x_{k_1} + x_{k_2}$;

class $\mathcal{E}^2$ contains all the polynomials with integer coefficients.

## Grzegorczyk's classes of primitive-recursive functions

$\mathcal{E}^l$ consist of functions $\mathbb{Z}^s \to \mathbb{Z}^t$.

For the base of recursion

class $\mathcal{E}^0$ contains
- constant functions $x_k \to c$,
- shifts $x_k \to x_k + c$,
- projections $(x_1, \dots, x_n) \to x_k$;

class $\mathcal{E}^1$ contains linear functions $x_k \to c \cdot x_k$ and $(x_{k_1}, x_{k_2}) \to x_{k_1} + x_{k_2}$;

class $\mathcal{E}^2$ contains all the polynomials with integer coefficients.

## Grzegorczyk's classes of primitive-recursive functions

$\mathcal{E}^l$ consist of functions $\mathbb{Z}^s \to \mathbb{Z}^t$.

For the base of recursion

class $\mathcal{E}^0$ contains
• constant functions $x_k \to c$,
• shifts $x_k \to x_k + c$,
• projections $(x_1, \dots, x_n) \to x_k$;

class $\mathcal{E}^1$ contains linear functions $x_k \to c \cdot x_k$ and $(x_{k_1}, x_{k_2}) \to x_{k_1} + x_{k_2}$;

class $\mathcal{E}^2$ contains all the polynomials with integer coefficients.

## Grzegorczyk's classes of primitive-recursive functions

$\mathcal{E}^l$ consist of functions $\mathbb{Z}^s \to \mathbb{Z}^t$.

For the base of recursion

class $\mathcal{E}^0$ contains
- constant functions $x_k \to c$,
- shifts $x_k \to x_k + c$,
- projections $(x_1, \dots, x_n) \to x_k$;

class $\mathcal{E}^1$ contains linear functions $x_k \to c \cdot x_k$ and $(x_{k_1}, x_{k_2}) \to x_{k_1} + x_{k_2}$;

class $\mathcal{E}^2$ contains all the polynomials with integer coefficients.

## Primitive and limited primitive recursion

Let $l \geq 2$. For the inductive step of the definition, assume that functions $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by the primitive recursion,

$$F(x_1, \ldots, x_n, 0) = G(x_1, \ldots, x_n), \tag{1}$$

$$F(x_1, \ldots, x_n, y+1) = H(x_1, \ldots, x_n, y, F(x_1, \ldots, x_n, y)), \tag{2}$$

belongs to $\mathcal{E}^{l+1}$.

To complete the definition of $\mathcal{E}^l$, $l \geq 0$, take the closure with respect to composition and the following *limited primitive recursion*:

> Let $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z), Q(x_1, \ldots, x_n, y) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by (1),(2) also belongs to $\mathcal{E}^l$, *provided that* $F(x_1, \ldots, x_n, y) \leq Q(x_1, \ldots, x_n, y)$.

Clearly, $\mathcal{E}^{l+1} \supset \mathcal{E}^l$.

Observe that $\mathcal{E}^3$ contains all towers of exponential functions.

Union $\cup_{l<\infty} \mathcal{E}^l$ coincides with the set of all primitive-recursive functions.

## Primitive and limited primitive recursion

Let $l \geq 2$. For the inductive step of the definition, assume that functions $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by the primitive recursion,

$$F(x_1, \ldots, x_n, 0) = G(x_1, \ldots, x_n), \quad (1)$$

$$F(x_1, \ldots, x_n, y + 1) = H(x_1, \ldots, x_n, y, F(x_1, \ldots, x_n, y)), \quad (2)$$

belongs to $\mathcal{E}^{l+1}$.

To complete the definition of $\mathcal{E}^l$, $l \geq 0$, take the closure with respect to composition and the following *limited primitive recursion*:

> Let $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z), Q(x_1, \ldots, x_n, y) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by (1),(2) also belongs to $\mathcal{E}^l$, *provided that* $F(x_1, \ldots, x_n, y) \leq Q(x_1, \ldots, x_n, y)$.

Clearly, $\mathcal{E}^{l+1} \supset \mathcal{E}^l$.

Observe that $\mathcal{E}^3$ contains all towers of exponential functions.

Union $\cup_{l<\infty} \mathcal{E}^l$ coincides with the set of all primitive-recursive functions.

## Primitive and limited primitive recursion

Let $l \geq 2$. For the inductive step of the definition, assume that functions $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by the primitive recursion,

$$F(x_1, \ldots, x_n, 0) = G(x_1, \ldots, x_n), \tag{1}$$

$$F(x_1, \ldots, x_n, y + 1) = H(x_1, \ldots, x_n, y, F(x_1, \ldots, x_n, y)), \tag{2}$$

belongs to $\mathcal{E}^{l+1}$.

To complete the definition of $\mathcal{E}^l$, $l \geq 0$, take the closure with respect to composition and the following *limited primitive recursion*:

> Let $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z), Q(x_1, \ldots, x_n, y) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by (1),(2) also belongs to $\mathcal{E}^l$, *provided that* $F(x_1, \ldots, x_n, y) \leq Q(x_1, \ldots, x_n, y)$.

Clearly, $\mathcal{E}^{l+1} \supset \mathcal{E}^l$.

Observe that $\mathcal{E}^3$ contains all towers of exponential functions.

Union $\cup_{l<\infty}\mathcal{E}^l$ coincides with the set of all primitive-recursive functions.

## Primitive and limited primitive recursion

Let $l \geq 2$. For the inductive step of the definition, assume that functions $G(x_1, \dots, x_n), H(x_1, \dots, x_n, y, z) \in \mathcal{E}^l$. Then the function $F(x_1, \dots, x_n, y)$ defined by the primitive recursion,

$$F(x_1, \dots, x_n, 0) = G(x_1, \dots, x_n), \tag{1}$$

$$F(x_1, \dots, x_n, y + 1) = H(x_1, \dots, x_n, y, F(x_1, \dots, x_n, y)), \tag{2}$$

belongs to $\mathcal{E}^{l+1}$.

To complete the definition of $\mathcal{E}^l$, $l \geq 0$, take the closure with respect to composition and the following *limited primitive recursion*:

> Let $G(x_1, \dots, x_n), H(x_1, \dots, x_n, y, z), Q(x_1, \dots, x_n, y) \in \mathcal{E}^l$. Then the function $F(x_1, \dots, x_n, y)$ defined by (1),(2) also belongs to $\mathcal{E}^l$, *provided that* $F(x_1, \dots, x_n, y) \leq Q(x_1, \dots, x_n, y)$.

Clearly, $\mathcal{E}^{l+1} \supset \mathcal{E}^l$.

Observe that $\mathcal{E}^3$ contains all towers of exponential functions.

Union $\cup_{l<\infty} \mathcal{E}^l$ coincides with the set of all primitive-recursive functions.

# Primitive and limited primitive recursion

Let $l \geq 2$. For the inductive step of the definition, assume that functions $G(x_1, \dots, x_n), H(x_1, \dots, x_n, y, z) \in \mathcal{E}^l$. Then the function $F(x_1, \dots, x_n, y)$ defined by the primitive recursion,

$$F(x_1, \dots, x_n, 0) = G(x_1, \dots, x_n), \tag{1}$$

$$F(x_1, \dots, x_n, y + 1) = H(x_1, \dots, x_n, y, F(x_1, \dots, x_n, y)), \tag{2}$$

belongs to $\mathcal{E}^{l+1}$.

To complete the definition of $\mathcal{E}^l$, $l \geq 0$, take the closure with respect to composition and the following *limited primitive recursion*:

> Let $G(x_1, \dots, x_n), H(x_1, \dots, x_n, y, z), Q(x_1, \dots, x_n, y) \in \mathcal{E}^l$. Then the function $F(x_1, \dots, x_n, y)$ defined by (1),(2) also belongs to $\mathcal{E}^l$, *provided that* $F(x_1, \dots, x_n, y) \leq Q(x_1, \dots, x_n, y)$.

Clearly, $\mathcal{E}^{l+1} \supset \mathcal{E}^l$.

Observe that $\mathcal{E}^3$ contains all towers of exponential functions.

Union $\cup_{l<\infty}\mathcal{E}^l$ coincides with the set of all primitive-recursive functions.

## Primitive and limited primitive recursion

Let $l \geq 2$. For the inductive step of the definition, assume that functions $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by the primitive recursion,

$$F(x_1, \ldots, x_n, 0) = G(x_1, \ldots, x_n), \tag{1}$$

$$F(x_1, \ldots, x_n, y+1) = H(x_1, \ldots, x_n, y, F(x_1, \ldots, x_n, y)), \tag{2}$$

belongs to $\mathcal{E}^{l+1}$.

To complete the definition of $\mathcal{E}^l$, $l \geq 0$, take the closure with respect to composition and the following *limited primitive recursion*:

> Let $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z), Q(x_1, \ldots, x_n, y) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by (1),(2) also belongs to $\mathcal{E}^l$, *provided that* $F(x_1, \ldots, x_n, y) \leq Q(x_1, \ldots, x_n, y)$.

Clearly, $\mathcal{E}^{l+1} \supset \mathcal{E}^l$.

Observe that $\mathcal{E}^3$ contains all towers of exponential functions.

Union $\cup_{l<\infty}\mathcal{E}^l$ coincides with the set of all primitive-recursive functions.

## Primitive and limited primitive recursion

Let $l \geq 2$. For the inductive step of the definition, assume that functions $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by the primitive recursion,

$$F(x_1, \ldots, x_n, 0) = G(x_1, \ldots, x_n), \tag{1}$$

$$F(x_1, \ldots, x_n, y+1) = H(x_1, \ldots, x_n, y, F(x_1, \ldots, x_n, y)), \tag{2}$$

belongs to $\mathcal{E}^{l+1}$.

To complete the definition of $\mathcal{E}^l$, $l \geq 0$, take the closure with respect to composition and the following *limited primitive recursion*:

> Let $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z), Q(x_1, \ldots, x_n, y) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by (1),(2) also belongs to $\mathcal{E}^l$, *provided that* $F(x_1, \ldots, x_n, y) \leq Q(x_1, \ldots, x_n, y)$.

Clearly, $\mathcal{E}^{l+1} \supset \mathcal{E}^l$.

Observe that $\mathcal{E}^3$ contains all towers of exponential functions.

Union $\cup_{l<\infty} \mathcal{E}^l$ coincides with the set of all primitive-recursive functions.

## Primitive and limited primitive recursion

Let $l \geq 2$. For the inductive step of the definition, assume that functions $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by the primitive recursion,

$$F(x_1, \ldots, x_n, 0) = G(x_1, \ldots, x_n), \tag{1}$$

$$F(x_1, \ldots, x_n, y+1) = H(x_1, \ldots, x_n, y, F(x_1, \ldots, x_n, y)), \tag{2}$$

belongs to $\mathcal{E}^{l+1}$.

To complete the definition of $\mathcal{E}^l$, $l \geq 0$, take the closure with respect to composition and the following *limited primitive recursion*:

> Let $G(x_1, \ldots, x_n), H(x_1, \ldots, x_n, y, z), Q(x_1, \ldots, x_n, y) \in \mathcal{E}^l$. Then the function $F(x_1, \ldots, x_n, y)$ defined by (1),(2) also belongs to $\mathcal{E}^l$, *provided that* $F(x_1, \ldots, x_n, y) \leq Q(x_1, \ldots, x_n, y)$.

Clearly, $\mathcal{E}^{l+1} \supset \mathcal{E}^l$.

Observe that $\mathcal{E}^3$ contains all towers of exponential functions.

Union $\cup_{l<\infty} \mathcal{E}^l$ coincides with the set of all primitive-recursive functions.

# Complexity of computing a quasi-linear common multiple

From the explicit bound on the Idealbasissatz (due to Seidenberg) we obtain the following complexity bound

**Corollary**

*Any pair of ordinary quasi-linear differential equations*
$y^{(k)} = p_k(y^{(k-1)}, \dots, y, x), \quad y^{(k)} = q_k(y^{(k-1)}, \dots, y, x)$
*of order $k$ with polynomials of degrees $\deg(p_k), \deg(q_k) \le d$ has a quasi-linear common multiple of order $g(d)$, where $g$ belongs to the class $\mathcal{E}^{k+2}$ of Grzegorczyk's hierarchy.*
*This provides also a complexity bound of the similar order of magnitude of the algorithm which looks for a quasi-linear common multiple by trying consecutively increasing orders $n$ of a candidate and solving the membership problem to an ideal generated by first $n$ derivatives (using Lemma above), say, with the help of Gröbner basis.*

In particular, in case of first-order equations ($k = 1$) function $g(d)$ grows exponentially.

# Complexity of computing a quasi-linear common multiple

From the explicit bound on the Idealbasissatz (due to Seidenberg) we obtain the following complexity bound

**Corollary**

*Any pair of ordinary quasi-linear differential equations*
$y^{(k)} = p_k(y^{(k-1)}, \dots, y, x), \quad y^{(k)} = q_k(y^{(k-1)}, \dots, y, x)$
*of order $k$ with polynomials of degrees $\deg(p_k), \deg(q_k) \leq d$ has a quasi-linear common multiple of order $g(d)$, where $g$ belongs to the class $\mathcal{E}^{k+2}$ of Grzegorczyk's hierarchy.*

*This provides also a complexity bound of the similar order of magnitude of the algorithm which looks for a quasi-linear common multiple by trying consecutively increasing orders $n$ of a candidate and solving the membership problem to an ideal generated by first $n$ derivatives (using Lemma above), say, with the help of Gröbner basis.*

In particular, in case of first-order equations ($k = 1$) function $g(d)$ grows exponentially.

# Complexity of computing a quasi-linear common multiple

From the explicit bound on the Idealbasissatz (due to Seidenberg) we obtain the following complexity bound

## Corollary

*Any pair of ordinary quasi-linear differential equations*
$y^{(k)} = p_k(y^{(k-1)}, \ldots, y, x), \quad y^{(k)} = q_k(y^{(k-1)}, \ldots, y, x)$
*of order $k$ with polynomials of degrees $\deg(p_k), \deg(q_k) \leq d$ has a quasi-linear common multiple of order $g(d)$, where $g$ belongs to the class $\mathcal{E}^{k+2}$ of Grzegorczyk's hierarchy.*
*This provides also a complexity bound of the similar order of magnitude of the algorithm which looks for a quasi-linear common multiple by trying consecutively increasing orders $n$ of a candidate and solving the membership problem to an ideal generated by first $n$ derivatives (using Lemma above), say, with the help of Gröbner basis.*

In particular, in case of first-order equations ($k = 1$) function $g(d)$ grows exponentially.

# Complexity of computing a quasi-linear common multiple

From the explicit bound on the Idealbasissatz (due to Seidenberg) we obtain the following complexity bound

## Corollary

*Any pair of ordinary quasi-linear differential equations*
$y^{(k)} = p_k(y^{(k-1)}, \dots, y, x), \quad y^{(k)} = q_k(y^{(k-1)}, \dots, y, x)$
*of order* $k$ *with polynomials of degrees* $\deg(p_k), \deg(q_k) \leq d$ *has a quasi-linear common multiple of order* $g(d)$*, where* $g$ *belongs to the class* $\mathcal{E}^{k+2}$ *of Grzegorczyk's hierarchy.*
*This provides also a complexity bound of the similar order of magnitude of the algorithm which looks for a quasi-linear common multiple by trying consecutively increasing orders* $n$ *of a candidate and solving the membership problem to an ideal generated by first* $n$ *derivatives (using Lemma above), say, with the help of Gröbner basis.*

In particular, in case of first-order equations ($k = 1$) function $g(d)$ grows exponentially.

## Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' + y = 0$. A common multiple of order 2 does not exist; however, our algorithm yields the following common multiple of order 3 involving a parameter $C$:
$E_3 \equiv y''' + (C-4)yy'' + (C+1)y'' + (2C-2)y'^2 + (2C+2)yy' + Cy' + Cy^2$.
For $C = 0$ it simplifies to $E_0 \equiv y''' + 4yy'' + y'' - 2y'^2 + 2yy' = 0$. Our factorization algorithm yields factors $y' + y^2$, $y' + y$ and $y'$ of $E_0$.

## Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' = 0$ with solutions $y = \frac{1}{x+C}$ and $y = C$ respectively. The common multiple algorithm for $E_1$ and $E_2$ yields $y'' + 2yy' = 0$. Its general solution is $y = C_1 \tan(C_2 - C_1 x)$.

## Remark

*The general solution of the second-order equation in the preceding example may also be written as $y = C_1 \tanh(C_2 + C_1 x)$. From the latter representation the constant solution may be obtained by taking the limit $C_2 \to \infty$.*

**Example**

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' + y = 0$. A common multiple of order 2 does not exist; however, our algorithm yields the following common multiple of order 3 involving a parameter $C$:
$E_3 \equiv y''' + (C-4)yy'' + (C+1)y'' + (2C-2)y'^2 + (2C+2)yy' + Cy' + Cy^2$.
For $C = 0$ it simplifies to $E_0 \equiv y''' + 4yy'' + y'' - 2y'^2 + 2yy' = 0$. Our factorization algorithm yields factors $y' + y^2$, $y' + y$ and $y'$ of $E_0$.

**Example**

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' = 0$ with solutions $y = \frac{1}{x+C}$ and $y = C$ respectively. The common multiple algorithm for $E_1$ and $E_2$ yields $y'' + 2yy' = 0$. Its general solution is $y = C_1 \tan(C_2 - C_1 x)$.

**Remark**

*The general solution of the second-order equation in the preceding example may also be written as $y = C_1 \tanh(C_2 + C_1 x)$. From the latter representation the constant solution may be obtained by taking the limit $C_2 \to \infty$.*

**Example**

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' + y = 0$. A common multiple of order 2 does not exist; however, our algorithm yields the following common multiple of order 3 involving a parameter $C$:
$E_3 \equiv y''' + (C-4)yy'' + (C+1)y'' + (2C-2)y'^2 + (2C+2)yy' + Cy' + Cy^2$.
For $C = 0$ it simplifies to $E_0 \equiv y''' + 4yy'' + y'' - 2y'^2 + 2yy' = 0$. Our factorization algorithm yields factors $y' + y^2$, $y' + y$ and $y'$ of $E_0$.

**Example**

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' = 0$ with solutions $y = \frac{1}{x+C}$ and $y = C$ respectively. The common multiple algorithm for $E_1$ and $E_2$ yields $y'' + 2yy' = 0$. Its general solution is $y = C_1 \tan(C_2 - C_1 x)$.

**Remark**

*The general solution of the second-order equation in the preceding example may also be written as $y = C_1 \tanh(C_2 + C_1 x)$. From the latter representation the constant solution may be obtained by taking the limit $C_2 \to \infty$.*

### Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' + y = 0$. A common multiple of order 2 does not exist; however, our algorithm yields the following common multiple of order 3 involving a parameter $C$:
$E_3 \equiv y''' + (C-4)yy'' + (C+1)y'' + (2C-2)y'^2 + (2C+2)yy' + Cy' + Cy^2$.
For $C = 0$ it simplifies to $E_0 \equiv y''' + 4yy'' + y'' - 2y'^2 + 2yy' = 0$. Our factorization algorithm yields factors $y' + y^2$, $y' + y$ and $y'$ of $E_0$.

### Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' = 0$ with solutions $y = \frac{1}{x+C}$ and $y = C$ respectively. The common multiple algorithm for $E_1$ and $E_2$ yields $y'' + 2yy' = 0$. Its general solution is $y = C_1 \tan(C_2 - C_1 x)$.

### Remark

*The general solution of the second-order equation in the preceding example may also be written as $y = C_1 \tanh(C_2 + C_1 x)$. From the latter representation the constant solution may be obtained by taking the limit $C_2 \to \infty$.*

## Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' + y = 0$. A common multiple of order 2 does not exist; however, our algorithm yields the following common multiple of order 3 involving a parameter $C$:
$E_3 \equiv y''' + (C-4)yy'' + (C+1)y'' + (2C-2)y'^2 + (2C+2)yy' + Cy' + Cy^2$.
For $C = 0$ it simplifies to $E_0 \equiv y''' + 4yy'' + y'' - 2y'^2 + 2yy' = 0$. Our factorization algorithm yields factors $y' + y^2$, $y' + y$ and $y'$ of $E_0$.

## Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' = 0$ with solutions $y = \frac{1}{x+C}$ and $y = C$ respectively. The common multiple algorithm for $E_1$ and $E_2$ yields $y'' + 2yy' = 0$. Its general solution is $y = C_1 \tan(C_2 - C_1 x)$.

## Remark

*The general solution of the second-order equation in the preceding example may also be written as $y = C_1 \tanh(C_2 + C_1 x)$. From the latter representation the constant solution may be obtained by taking the limit $C_2 \to \infty$.*

## Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' + y = 0$. A common multiple of order 2 does not exist; however, our algorithm yields the following common multiple of order 3 involving a parameter $C$:
$E_3 \equiv y''' + (C-4)yy'' + (C+1)y'' + (2C-2)y'^2 + (2C+2)yy' + Cy' + Cy^2$.
For $C = 0$ it simplifies to $E_0 \equiv y''' + 4yy'' + y'' - 2y'^2 + 2yy' = 0$. Our factorization algorithm yields factors $y' + y^2$, $y' + y$ and $y'$ of $E_0$.

## Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' = 0$ with solutions $y = \frac{1}{x+C}$ and $y = C$ respectively. The common multiple algorithm for $E_1$ and $E_2$ yields $y'' + 2yy' = 0$. Its general solution is $y = C_1 \tan(C_2 - C_1 x)$.

## Remark

*The general solution of the second-order equation in the preceding example may also be written as* $y = C_1 \tanh(C_2 + C_1 x)$. *From the latter representation the constant solution may be obtained by taking the limit* $C_2 \to \infty$.

### Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' + y = 0$. A common multiple of order 2 does not exist; however, our algorithm yields the following common multiple of order 3 involving a parameter $C$:
$E_3 \equiv y''' + (C-4)yy'' + (C+1)y'' + (2C-2)y'^2 + (2C+2)yy' + Cy' + Cy^2$.
For $C = 0$ it simplifies to $E_0 \equiv y''' + 4yy'' + y'' - 2y'^2 + 2yy' = 0$. Our factorization algorithm yields factors $y' + y^2$, $y' + y$ and $y'$ of $E_0$.

### Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' = 0$ with solutions $y = \frac{1}{x+C}$ and $y = C$ respectively. The common multiple algorithm for $E_1$ and $E_2$ yields $y'' + 2yy' = 0$. Its general solution is $y = C_1 \tan(C_2 - C_1 x)$.

### Remark

*The general solution of the second-order equation in the preceding example may also be written as $y = C_1 \tanh(C_2 + C_1 x)$. From the latter representation the constant solution may be obtained by taking the limit $C_2 \to \infty$.*

## Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' + y = 0$. A common multiple of order 2 does not exist; however, our algorithm yields the following common multiple of order 3 involving a parameter $C$:
$E_3 \equiv y''' + (C-4)yy'' + (C+1)y'' + (2C-2)y'^2 + (2C+2)yy' + Cy' + Cy^2$.
For $C = 0$ it simplifies to $E_0 \equiv y''' + 4yy'' + y'' - 2y'^2 + 2yy' = 0$. Our factorization algorithm yields factors $y' + y^2$, $y' + y$ and $y'$ of $E_0$.

## Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' = 0$ with solutions $y = \frac{1}{x+C}$ and $y = C$ respectively. The common multiple algorithm for $E_1$ and $E_2$ yields $y'' + 2yy' = 0$. Its general solution is $y = C_1 \tan(C_2 - C_1 x)$.

## Remark

*The general solution of the second-order equation in the preceding example may also be written as* $y = C_1 \tanh(C_2 + C_1 x)$. *From the latter representation the constant solution may be obtained by taking the limit* $C_2 \to \infty$.

## Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' + y = 0$. A common multiple of order 2 does not exist; however, our algorithm yields the following common multiple of order 3 involving a parameter $C$:
$E_3 \equiv y''' + (C-4)yy'' + (C+1)y'' + (2C-2)y'^2 + (2C+2)yy' + Cy' + Cy^2$.
For $C = 0$ it simplifies to $E_0 \equiv y''' + 4yy'' + y'' - 2y'^2 + 2yy' = 0$. Our factorization algorithm yields factors $y' + y^2$, $y' + y$ and $y'$ of $E_0$.

## Example

Let $E_1 \equiv y' + y^2 = 0$ and $E_2 \equiv y' = 0$ with solutions $y = \frac{1}{x+C}$ and $y = C$ respectively. The common multiple algorithm for $E_1$ and $E_2$ yields $y'' + 2yy' = 0$. Its general solution is $y = C_1 \tan(C_2 - C_1 x)$.

## Remark

*The general solution of the second-order equation in the preceding example may also be written as $y = C_1 \tanh(C_2 + C_1 x)$. From the latter representation the constant solution may be obtained by taking the limit $C_2 \to \infty$.*