# Разработка и исследование алгоритма приведения операторных матриц по строкам

Гулак Максим Андреевич, 625 группа
Научный руководитель: Панфёров Антон Александрович

24 апреля 2026 г.

# Операторные матрицы

Системы линейных дифференциальных уравнений могут быть представлены в матричном виде, например:

$$\begin{cases} y_1'' + xy_2'' + (2x^2+1)y_3 = 0 \\ y_1''' + 2y_2'' + (x^2+x)y_3 + xy_1 = 0 \\ y_1' + xy_2' + y_3 = 0 \end{cases}$$

$$\Updownarrow$$

$$Ly = 0,$$

где $L = \begin{pmatrix} D^2 & xD^2 & 2x^2+1 \\ D^3+x & 2D^2 & x^2+x \\ D & xD & 1 \end{pmatrix} \in K[D]^{3\times 3}$, $y = (y_1, \ldots, y_3)^T$,

$D = \dfrac{d}{dx}$ — оператор дифференцирования.

# Фронтальная матрица

$$L = \begin{pmatrix} D^2 & xD^2 & 2x^2+1 \\ D^3+x & 2D^2 & x^2+x \\ D & xD & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} D^3 + \begin{pmatrix} 1 & x & 0 \\ 0 & 2 & x \\ 0 & 0 & 0 \end{pmatrix} D^2 + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & x & 0 \end{pmatrix} D + \begin{pmatrix} 0 & 0 & 2x^2+1 \\ 1 & 0 & x^2+x \\ 0 & 0 & 1 \end{pmatrix}$$

# Фронтальная матрица

$$L = \begin{pmatrix} D^2 & xD^2 & 2x^2+1 \\ D^3+x & 2D^2 & x^2+x \\ D & xD & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 0 & 0 & 0 \\ \boxed{1 & 0 & 0} \\ 0 & 0 & 0 \end{pmatrix} D^3 + \begin{pmatrix} \boxed{1 & x & 0} \\ 0 & 2 & x \\ 0 & 0 & 0 \end{pmatrix} D^2 + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \boxed{1 & x & 0} \end{pmatrix} D + \begin{pmatrix} 0 & 0 & 2x^2+1 \\ 1 & 0 & x^2+x \\ 0 & 0 & 1 \end{pmatrix}$$

$$\hat{L} = \begin{pmatrix} 1 & x & 0 \\ 1 & 0 & 0 \\ 1 & x & 0 \end{pmatrix} \text{ — } \textit{фронтальная матрица}$$

# Приведённая по строкам форма

## Определение

Операторная матрица находится в приведённой по строкам форме, если ненулевые строки её фронтальной матрицы линейно независимы над $K$.

Применение:

- проверка полноты ранга системы, нахождение ранга;
- проверка унимодулярности матрицы;
- использование в других алгоритмах.

Для приведения по строкам существует алгоритм Row-Reduction (RR).

Для полиномиальных матриц существует алгоритм WeakPopovForm (WPF), осуществляющий преобразование к слабой форме Попова.

# Постановка задачи

1. Изучить алгоритмы RR и WPF;
2. Разработать алгоритм QRR приведения операторных матриц на основе алгоритма WPF;
3. Получить теоретические оценки сложности алгоритмов RR и QRR;
4. Реализовать алгоритмы RR и QRR в системе SageMath;
5. Получить экспериментальные оценки времени работы алгоритмов.

# Алгоритм Row-Reduction

**Обозначения:**

$L_{i,*}$ – $i$-я строка матрицы $L$.

$\operatorname{ord} L$ – порядок $L$.

$\delta_i = \operatorname{ord} L_{i,*}$ – порядок $i$-й строки $L$.

$\vec{\delta} = (\delta_1, \ldots, \delta_m)$ – вектор порядков строк.

$|\vec{\delta}| = \sum_{i=1}^{m} \delta_i$ – сумма вектора порядков строк.

## Пример

$$L = \begin{pmatrix} D^2 & xD^2 & 2x^2+1 \\ D^3+x & 2D^2 & x^2+x \\ D & xD & 1 \end{pmatrix}, \operatorname{ord} L = 3, \quad \vec{\delta} = (2,3,1)$$

# Алгоритм Row-Reduction (Beckermann et al.)

**Вход:** матрица дифференциальных операторов $L$ с вектором порядков строк $\vec{\delta} = (\delta_1, ..., \delta_m)$.
**Выход:** приведённая по строкам матрица $L'$ и матрица $U$ такая, что $L' = UL$.
**Инициализация:** $L^{'} := L$, $U := E_m$, $\vec{\delta}^{'} := \vec{\delta}$, $\hat{L}^{'}$ — фронтальная матрица.

Пока ненулевые строки $\hat{L}^{'}$ линейно зависимы:

1. вычислить ненулевой вектор $\vec{v} = (v_1, ..., v_m)$, такой, что $\vec{v}\hat{L}^{'} = 0$;
2. выбрать целое число $t$ такое, что $v_t \neq 0, \delta_t^{'} = \max(\delta_i^{'}, v_i \neq 0)$;
3. заменить $L_{t,*}^{'}$ на $\sum_{i=1}^{m} v_i D^{\delta_t^{'} - \delta_i^{'}} L_{i,*}^{'}$;
4. заменить $U_{t,*}$ на $\sum_{i=1}^{m} v_i D^{\delta_t^{'} - \delta_i^{'}} U_{i,*}^{'}$;
5. обновить $\hat{L}^{'}$ и $\vec{\delta}^{'}$.

# Оценка сложности RR

## Утверждение (Barkatou M. A. et al.)

Сложность алгоритма Row-Reduction по числу операций в $K$ допускает оценку

$$O\big(m^2(|\vec{\delta}| + m)(m + 2\ell + |\vec{\delta}|)\big).$$

Действия:

1. Вычисление вектора $\vec{v}$ $\leqslant m^3$ операций
2. выбор индекса
3. $L'_{t,*} \to \sum_{i=1}^{m} v_i D^{\delta'_t - \delta'_i} L'_{i,*}$ $\leqslant m^2\ell$ операций
4. $U_{t,*} \to \sum_{i=1}^{m} v_i D^{\delta'_t - \delta'_i} U'_{i,*}$ $\leqslant m^2(2\ell + |\vec{\delta}|)$ операций

Цикл нужно повторить в худшем случае $m + |\vec{\delta}| - 1$ раз.

# Оценка сложности RR

Случай $K = \mathbb{Q}[x]$:

## Утверждение

Сложность алгоритма Row-Reduction по числу операций в $\mathbb{Q}$ допускает оценку

$$O\big(m^2 d^2 (|\vec{\delta}| + m)(md + 2\ell + |\vec{\delta}|)\big),$$

где $d$ — максимальная степень многочленов из $K$.

Действия:

1. Вычисление вектора $\vec{v}$ $\leqslant m^3 d^3$ операций
2. выбор индекса
3. $L'_{t,*} \to \sum_{i=1}^{m} v_i D^{\delta'_t - \delta'_i} L'_{i,*}$ $\leqslant m^2 d^2 \ell$ операций
4. $U_{t,*} \to \sum_{i=1}^{m} v_i D^{\delta'_t - \delta'_i} U'_{i,*}$ $\leqslant m^2 d^2 (2\ell + |\vec{\delta}|)$ операций

Цикл нужно повторить в худшем случае $m + |\vec{\delta}| - 1$ раз.

# Слабая форма Попова операторных матриц

## Определение

Опорный индекс $I_j(L)$ $j$-й строки операторной матрицы $L$ — наименьший номер столбца с наибольшим порядком оператора в этой строке.

## Пример

$$L = \begin{pmatrix} D^2 & xD^2 & 2x^2+1 \\ D^2+x & 2D^3 & x^2+x \\ D & xD & 1 \end{pmatrix} \quad \begin{aligned} I_1(L) &= 1 \\ I_2(L) &= 2 \\ I_3(L) &= 1 \end{aligned}$$

# Слабая форма Попова операторных матриц

**Определение**

Операторная матрица $L$ находится в слабой форме Попова, если все ненулевые опорные индексы $L$ различны.

**Утверждение**

Пусть операторная матрица находится в слабой форме Попова. Тогда она является приведённой по строкам.

Для полиномиальных матриц существует алгоритм WeakPopovForm (WPF), описанный в статье *On lattice reduction for polynomial matrices.* (Mulders T., Storjohann A., 2003)

# Алгоритм QRR

**Вход:** матрица дифференциальных операторов $L$ с вектором порядков строк $\vec{\delta} = (\delta_1, ..., \delta_m)$.
**Выход:** приведённая по строкам матрица $L'$ и матрица $U \in K[\partial]^{m \times m}$, такая, что $L' = UL$
**Инициализация:** $L' = L,\ U' = E_m,\ \vec{\delta}' = \vec{\delta}$.

Пока есть строки с одинаковыми ненулевыми опорными индексами:

1. найти такие $i, j$, что $I_i(L') = I_j(L')$ и $\vec{\delta}'_i \geqslant \vec{\delta}'_j$;
2. заменить $L'_{i,*}$ на $\hat{L'}_{j,I_i(L')} L'_{i,*} - \hat{L'}_{i,I_i(L')} D^{\vec{\delta}'_i - \vec{\delta}'_j} L'_{j,*}$;
3. заменить $U'_{i,*}$ на $\hat{L'}_{j,I_i(L')} U'_{i,*} - \hat{L'}_{i,I_i(L')} D^{\vec{\delta}'_i - \vec{\delta}'_j} U'_{j,*}$;
4. обновить $I_i(L')$ и $\vec{\delta}'$.

# Оценка сложности QRR

## Предложение

Сложность алгоритма QRR по числу операций в $K$ допускает оценку

$$O\left(m\ell\left(|\vec{\delta}| + \frac{m(m-1)}{2}\right)(2\ell + |\vec{\delta}|)\right).$$

Действия:

1. Поиск строк $\leqslant |\vec{\delta}| + \frac{m(m-1)}{2}$ раз
2. $L'_{i,*} \to \hat{L'}_{j,I_i(L')} L'_{i,*} - \hat{L'}_{i,I_i(L')} D^{\delta'_i - \delta'_j} L'_{j,*}$ $\leqslant m\ell^2$ операций
3. $U'_{i,*} \to \hat{L'}_{j,I_i(L')} U'_{i,*} - \hat{L'}_{i,I_i(L')} D^{\delta'_i - \delta'_j} U'_{j,*}$ $\leqslant m\ell(\ell + |\vec{\delta}|)$ операций

# Оценка сложности QRR

Случай $K = \mathbb{Q}[x]$:

**Предложение**

Сложность алгоритма QRR по числу операций в $\mathbb{Q}$ допускает оценку

$$O\left(md^2\ell\left(|\vec{\delta}| + \frac{m(m-1)}{2}\right)(2\ell + |\vec{\delta}|)\right),$$

где $d$ — максимальная степень многочленов из $K$.

Действия:

1. Поиск строк $\leqslant |\vec{\delta}| + \frac{m(m-1)}{2}$ раз
2. $L'_{i,*} \to \hat{L'}_{j,I_i(L')} L'_{i,*} - \hat{L'}_{i,I_i(L')} D^{\delta'_i - \delta'_j} L'_{j,*}$ $\leqslant m\ell^2 d^2$ операций
3. $U'_{i,*} \to \hat{L'}_{j,I_i(L')} U'_{i,*} - \hat{L'}_{i,I_i(L')} D^{\delta'_i - \delta'_j} U'_{j,*}$ $\leqslant m\ell d^2(\ell + |\vec{\delta}|)$ операций

# Точность оценок

## Утверждение

Полученные оценки являются точными.

Пусть

$$L = \begin{pmatrix} 2D^2 + 7D - 3 & 2D^2 + 5D - 2 \\ D^2 + 4D + \frac{1}{2} & D^2 + 3D + \frac{1}{2} \end{pmatrix}.$$

Тогда для $L$ достигается максимальное количество операций.

В ходе алгоритма порядки элементов изменяются таким образом, что требуется 5 итераций цикла:

$$\begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} \to \begin{pmatrix} 2 & 2 \\ 1 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \to \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \to \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \to \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix}$$

# Моделирование операторов

Кольцо многочленов Оре — кольцо $K[Z, \sigma, \delta]$ многочленов от $Z$ с умножением, заданным формулой

$$Za = \sigma(a)Z + \delta a, \forall a \in K,$$

где $K$ — кольцо или поле, $\sigma : K \to K, \delta : K \to K$

$$\delta(a + b) = \delta a + \delta b,$$

$$\delta(ab) = \sigma(a)\delta(b) + \delta(a)b, \forall a, b \in K.$$

Если $K = \mathbb{Q}(x), \sigma = 1, \delta = \frac{d}{dx}$, то $\mathbb{Q}(x)[D, \frac{d}{dx}]$ моделирует кольцо дифференциальных операторов.

# Многочлены Ope в SageMath

Для использования многочленов Ope был использован встроенный модуль в SageMath для работы с некоммутативными многочленами.

Пример инициализации кольца многочленов Ope и вычисления в нём:

[3]:
```
R.<x> = QQ[] # кольцо многочленов от x
der = R.derivation()
A.<D> = R['D',der]
```

[4]:
```
A
```

[4]:
```
Ore Polynomial Ring in D over Univariate Polynomial Ring in x over Rational Field twisted by d/dx
```

[5]:
```
D*(3*x^2+x)
```

[5]:
```
(3*x^2 + x)*D + 6*x + 1
```

# Реализация алгоритма RR

Реализован в виде функции `row_red(L, trans=True)`.
Дополнительно используются функции:

- `frMat(M)` для вычисления фронтальной матрицы;
- `rowRanks(M)` для вычисления вектора порядков строк ;
- `null_space_vector(M)` для вычисления вектора линейной зависимости.

[10]:
```
L = matrix([[D^2, x*D^2, 2*x^2+D],
            [D^3+x,2*D^2, x^2+x],
            [D, x*D, 1]])
pretty_print(row_red(L))
```

$$\left(\begin{pmatrix} 0 & D & -2x^2 \\ -x & 0 & \left(2x^3+1\right)D^2+8x^2D-x^2+3x \\ D & xD & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & D \\ xD^2 & -1 & -xD^3+D^2 \\ 0 & 0 & 1 \end{pmatrix}\right)$$

# Реализация алгоритма QRR

Реализован в виде функции `quick_row_red(L, trans=True)`.

- для каждой строки вычисляется опорный индекс;
- индексы хранятся в словаре `pivots`: ключ — номер столбца, значение — список пар (номер строки, опорная степень).
- если для одного столбца найдено несколько строк, выполняется трансформация только для одной из них;
- после преобразования пересчитывается только изменившаяся строка.

[16]:
```
L = matrix([[D^2, x*D^2, 2*x^2+D],
            [D^3+x,2*D^2, x^2+x],
            [D, x*D, 1]])
pretty_print(quick_row_red(L))
```

$$\left(\begin{pmatrix} 0 & -D & 2x^2 \\ -x & 0 & \left(2x^3+1\right)D^2+8x^2D-x^2+3x \\ D & xD & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & -D \\ xD^2 & -1 & -xD^3+D^2 \\ 0 & 0 & 1 \end{pmatrix}\right)$$

# Система проведения экспериментов

Необходима для унификации процесса проведения экспериментов по замеру времени работы произвольного количества алгоритмов с разными параметрами.
Реализована в виде класса `TestFramework`.

**Основные методы**

- `set_test_params`
- `generate_matrix`
- `run_single_test`
- `run_experiment`
- `save_results / load_results`
- `plot_results`

```
[113]: pretty_print(test.generate_matrix(A, 2, 3, 2))
```

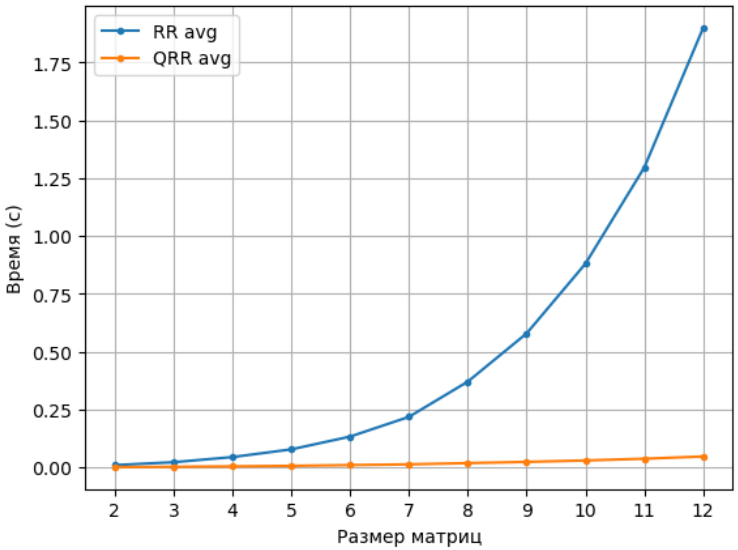
$$\begin{pmatrix} D^3+5D^2+6D+1 & (-3x^2-x+5)\,D^3+(-15x^2-23x+21)\,D^2+(-18x^2-66x-1)\,D-3x^2-37x-32 \\ D^3+6D^2+10D+4 & (-3x^2-x+5)\,D^3+(-18x^2-24x+26)\,D^2+(-30x^2-82x+16)\,D-12x^2-64x-29 \end{pmatrix}$$

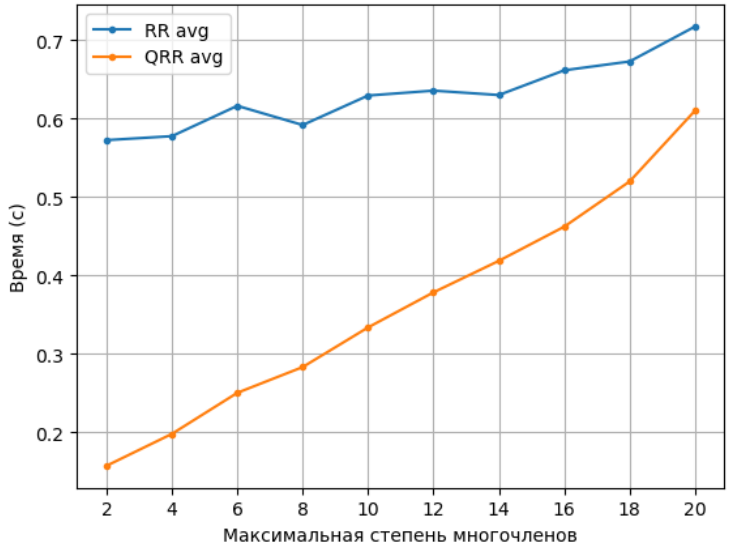
# Генерация матриц

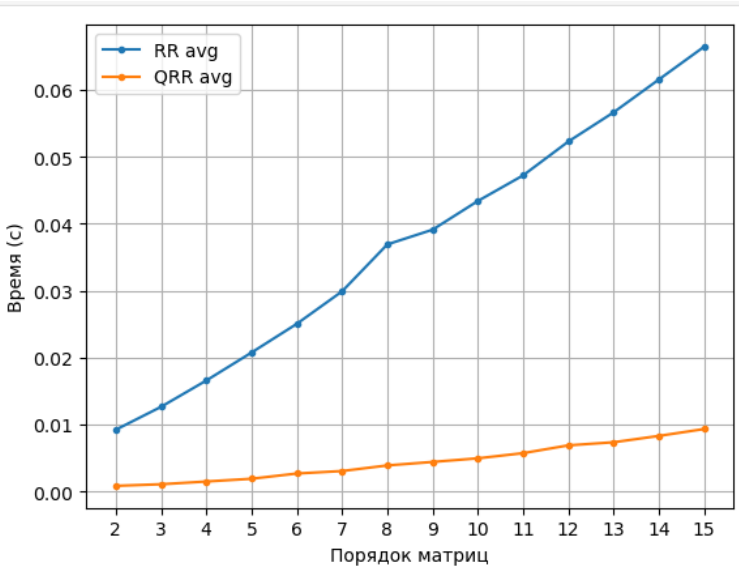
Идея генерации:

- строить матрицы, близкие к худшему случаю для QRR;
- не генерировать их напрямую, а восстанавливать в обратном порядке;
- начинать с базовой матрицы порядка 0 и рекуррентно получать матрицу нужного порядка.

# Результаты тестирования




Вычисления проводились в SageMath 10.5, Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 16 GB ОЗУ.

# Заключение

1. были изучены алгоритмы RR и WPF;
2. на основе алгоритма WPF был разработан новый алгоритм QuickRowReduction;
3. получены оценки сложности алгоритма QRR, а существующая оценка RR была уточнена;
4. алгоритмы RR и QRR были реализованы в системе компьютерной алгебры SageMath
5. разработана программная система, обеспечивающая генерацию тестовых матриц, запуск серий экспериментов, сбор статистики и визуализацию результатов;
6. проведено экспериментальное сравнение алгоритмов QRR и RR по времени работы.

# Литература


1. Beckermann B., Cheng H. and Labahn G. Fraction-free row reduction of matrices Ore polynomials. *Journal of Symbolic Computation*, 41(5), 513–543, 2006.
2. Barkatou M. A., El Bacha C., Labahn G., and Pflügel E. On simultaneous row and column reduction of higher-order linear differential systems. *Journal of Symbolic Computation*, 49(1), 45–64, 2013.
3. Mulders T., Storjohann A.: On Lattice Reduction for Polynomial Matrices. *Journal of Symbolic Computation* 37(4), 485–510, 2004.
4. Algorithms for computer algebra / Geddes K. O., Czapor S. R., Labahn G. – Boston: Kluwer Academic Publishers, 1992. 585 p.
5. Ore O. Theory of non-commutative polynomials. *Annals of Mathematics*, 34, 480–508, 1933.
6. Computational Mathematics with SageMath / Zimmermann P, Casamayou A., Cohen N. et al. – Philadelphia: SIAM, 2018. 464 p.