Перемножение матриц: практические (точные) методы

Группа 1: тройной цикл, алгоритм Штрассена, вариант Штрассена–Винограда

Аннотация

Мы рассматриваем три практически применимых и *точных* алгоритма умножения квадратных матриц: классический тройной цикл с трудоёмкостью $\Theta(n^3)$, алгоритм Штрассена (1969) с $\Theta(n^{\log_2 7}) \approx \Theta(n^{2{,}8074})$ и его усовершенствованный вариант Штрассена–Винограда (1971), уменьшающий константу числа сложений. Для каждого алгоритма даётся краткая историческая справка, формулы, небольшой пример, код на Python, обсуждение численных и практических аспектов, а также сопоставление теоретической и эмпирической производительности.

1 Введение

Умножение матриц — центральная операция численных вычислений: от решения систем линейных уравнений и методов наименьших квадратов до компьютерной графики и глубокого обучения. Для двух матриц $A \in \mathbb{R}^{n\times n}$ и $B \in \mathbb{R}^{n\times n}$ результат $C = AB$ определяется поэлементно

$$c_{ij} = \sum_{r=1}^{n} a_{ir}\, b_{rj}, \qquad i, j = 1, \ldots, n. \tag{1}$$

Наивная реализация даёт n^3 умножений и $n^2(n-1) = n^3 - n^2$ сложений, то есть трудоёмкость порядка $\Theta(n^3)$. В настоящей секции собраны *практические* алгоритмы, использующие точную арифметику: классический тройной цикл, метод Штрассена и его вариант Штрассена–Винограда. В отличие от приближённых тензорных схем, эти методы корректны для обычной точной арифметики над полями и кольцами; при работе с числами с плавающей запятой обсуждаются вопросы накопления ошибок.

2 Классический алгоритм (тройной цикл)

История и идея

Это прямое следование определению, известное со времён становления линейной алгебры. Его сильные стороны — простота, численная устойчивость и отличная локальность данных, что позволяет эффективно оптимизировать реализацию (блочное умножение, векторизация, параллелизм).

Сложность и операции

Точная оценка: n^3 умножений и $n^3 - n^2$ сложений. Асимптотика: $\Theta(n^3)$.

Пример 3×3

Возьмём:

$$A = \begin{pmatrix} 2 & -1 & 3 \\ 0 & 5 & 1 \\ 4 & 2 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 4 & -2 \\ 3 & -1 & 0 \\ 2 & 5 & 1 \end{pmatrix}.$$

Посчитаем два элемента явно (скалярные произведения строки на столбец):

$$c_{12} = (2, -1, 3) \cdot (4, -1, 5) = 2 \cdot 4 + (-1) \cdot (-1) + 3 \cdot 5 = 8 + 1 + 15 = 24,$$
$$c_{33} = (4, 2, 1) \cdot (-2, 0, 1) = 4 \cdot (-2) + 2 \cdot 0 + 1 \cdot 1 = -8 + 0 + 1 = -7.$$

Полный результат (опуская промежуточные вычисления для остальных семи элементов):

$$C = AB = \begin{pmatrix} 5 & 24 & -1 \\ 17 & 0 & 1 \\ 12 & 19 & -7 \end{pmatrix}.$$

Реализация (Python)

```
def matmul_classic(A, B):
    n = len(A)
    C = [[0]*n for _ in range(n)]
    for i in range(n):
        for k in range(n):
            aik = A[i][k]
            for j in range(n):
                C[i][j] += aik * B[k][j]
    return C
```

Данный «ijk/ikj»-вариант использует кэш-дружелюбный порядок (`aik` вытягивается вне внутреннего цикла).

Численные и практические аспекты

Метод устойчив при обычных вычислениях с плавающей запятой; ошибки округления растут умеренно. На малых n и при высокооптимизированных BLAS-реализациях тройной цикл (в блочной, векторизованной форме) часто оказывается быстрее более изощрённых алгоритмов за счёт меньшей константы и отсутствия накладных расходов.

3 Алгоритм Штрассена (1969)

История и идея

Ф. Штрассен в 1969 г. показал, что можно умножать матрицы быстрее $\Theta(n^3)$. Идея: рекурсивное разбиение на блоки 2×2 и вычисление произведения этих блоков *семью* умножениями вместо восьми за счёт дополнительных сложений/вычитаний. При рекурсивном применении получается сложность $T(n) = 7T(n/2) + O(n^2)$, то есть

$$T(n) = \Theta\big(n^{\log_2 7}\big) \approx \Theta(n^{2{,}8074}). \tag{2}$$

Формулы для блока 2×2

Пусть $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, $B = \begin{pmatrix} e & f \\ g & h \end{pmatrix}$. Введём семь промежуточных величин:

$$\begin{aligned} &M_1 = (a+d)(e+h), && M_2 = (c+d)e, && M_3 = a(f-h), \\ &M_4 = d(g-e), && M_5 = (a+b)h, && M_6 = (c-a)(e+f), \\ &M_7 = (b-d)(g+h). \end{aligned}$$

Тогда элементы $C = AB$ выражаются как:

$$\begin{aligned} &C_{11} = M_1 + M_4 - M_5 + M_7, && C_{12} = M_3 + M_5, \\ &C_{21} = M_2 + M_4, && C_{22} = M_1 - M_2 + M_3 + M_6. \end{aligned}$$

Короткий числовой пример 2×2

Возьмём $A = \left(\begin{smallmatrix} 1 & 3 \\ 7 & 5 \end{smallmatrix}\right)$, $B = \left(\begin{smallmatrix} 6 & 8 \\ 4 & 2 \end{smallmatrix}\right)$. Подсчёт даёт $M_1 = 48$, $M_2 = 72$, $M_3 = 6$, $M_4 = -10$, $M_5 = 8$, $M_6 = 84$, $M_7 = -12$, а затем

$$C = \begin{pmatrix} 18 & 14 \\ 62 & 66 \end{pmatrix},$$

что совпадает с обычным перемножением.

Как это работает на матрице 3×3: разбиение на блоки и "склейка"

Чтобы применить схему к нечётному размеру, дописываем нулевые строку и столбец и получаем матрицы $A', B' \in \mathbb{R}^{4\times4}$:

$$A' = \begin{pmatrix} 2 & -1 & 3 & 0 \\ 0 & 5 & 1 & 0 \\ 4 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad B' = \begin{pmatrix} 1 & 4 & -2 & 0 \\ 3 & -1 & 0 & 0 \\ 2 & 5 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Разобьём на блоки 2×2:

$$A' = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} \begin{smallmatrix} 2 & -1 \\ 0 & 5 \end{smallmatrix} & \begin{smallmatrix} 3 & 0 \\ 1 & 0 \end{smallmatrix} \\ \begin{smallmatrix} 4 & 2 \\ 0 & 0 \end{smallmatrix} & \begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix} \end{pmatrix}, \quad B' = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} \begin{smallmatrix} 1 & 4 \\ 3 & -1 \end{smallmatrix} & \begin{smallmatrix} -2 & 0 \\ 0 & 0 \end{smallmatrix} \\ \begin{smallmatrix} 2 & 5 \\ 0 & 0 \end{smallmatrix} & \begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix} \end{pmatrix}.$$

Далее вычисляем семь произведений для этих блоков (все блоки размера 2×2):

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22}) = \begin{pmatrix} 3 & 13 \\ 15 & -5 \end{pmatrix}, \qquad M_2 = (A_{21} + A_{22})B_{11} = \begin{pmatrix} 11 & 18 \\ 0 & 0 \end{pmatrix},$$

$$M_3 = A_{11}(B_{12} - B_{22}) = \begin{pmatrix} -6 & 0 \\ 0 & 0 \end{pmatrix}, \qquad M_4 = A_{22}(B_{21} - B_{11}) = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix},$$

$$M_5 = (A_{11} + A_{12})B_{22} = \begin{pmatrix} 5 & 0 \\ 1 & 0 \end{pmatrix}, \qquad M_6 = (A_{21} - A_{11})(B_{11} + B_{12}) = \begin{pmatrix} 7 & 5 \\ -15 & 5 \end{pmatrix},$$

$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22}) = \begin{pmatrix} 6 & 10 \\ 3 & 5 \end{pmatrix}.$$

Комбинируем их в четыре блоки результата:

$$C_{11} = M_1 + M_4 - M_5 + M_7 = \begin{pmatrix} 5 & 24 \\ 17 & 0 \end{pmatrix}, \qquad C_{12} = M_3 + M_5 = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix},$$

$$C_{21} = M_2 + M_4 = \begin{pmatrix} 12 & 19 \\ 0 & 0 \end{pmatrix}, \qquad C_{22} = M_1 - M_2 + M_3 + M_6 = \begin{pmatrix} -7 & 0 \\ 0 & 0 \end{pmatrix}.$$

И, наконец, "склеиваем" блоки в $C' = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$ и отбрасываем последнюю строку и столбец (искусственно добавленные нули):

$$C' = \begin{pmatrix} 5 & 24 & -1 & 0 \\ 17 & 0 & 1 & 0 \\ 12 & 19 & -7 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \Longrightarrow \boxed{C = C'_{1:3,\,1:3} = \begin{pmatrix} 5 & 24 & -1 \\ 17 & 0 & 1 \\ 12 & 19 & -7 \end{pmatrix}}.$$

Такое построение показывает, как блоки вычисляются *параллельно* и затем объединяются в полный результат.

Реализация (Python, компактная)

```python
def matmul_strassen(A, B, cutoff=64):
    n = len(A)
    if n <= cutoff:  # classic triple loop
        C = [[0]*n for _ in range(n)]
        for i in range(n):
            for k in range(n):
                aik = A[i][k]
                for j in range(n):
                    C[i][j] += aik * B[k][j]
        return C
    if n & (n-1):  # pad to next power of two
        m = 1 << (n-1).bit_length()
        Ap = [row + [0]*(m-n) for row in A] + [[0]*m for _ in range(m-n)]
        Bp = [row + [0]*(m-n) for row in B] + [[0]*m for _ in range(m-n)]
        Cp = matmul_strassen(Ap, Bp, cutoff)
        return [row[:n] for row in Cp[:n]]
    m = n // 2
    a11 = [r[:m] for r in A[:m]]; a12 = [r[m:] for r in A[:m]]
    a21 = [r[:m] for r in A[m:]]; a22 = [r[m:] for r in A[m:]]
    b11 = [r[:m] for r in B[:m]]; b12 = [r[m:] for r in B[:m]]
    b21 = [r[:m] for r in B[m:]]; b22 = [r[m:] for r in B[m:]]
    add = lambda X, Y: [[X[i][j] + Y[i][j] for j in range(m)] for i in range(m)]
    sub = lambda X, Y: [[X[i][j] - Y[i][j] for j in range(m)] for i in range(m)]
    M1 = matmul_strassen(add(a11, a22), add(b11, b22), cutoff)
    M2 = matmul_strassen(add(a21, a22), b11, cutoff)
    M3 = matmul_strassen(a11, sub(b12, b22), cutoff)
    M4 = matmul_strassen(a22, sub(b21, b11), cutoff)
    M5 = matmul_strassen(add(a11, a12), b22, cutoff)
    M6 = matmul_strassen(sub(a21, a11), add(b11, b12), cutoff)
    M7 = matmul_strassen(sub(a12, a22), add(b21, b22), cutoff)
    c11 = sub(add(add(M1, M4), M7), M5)
    c12 = add(M3, M5)
    c21 = add(M2, M4)
    c22 = add(sub(add(M1, M3), M2), M6)
    return [c11[i] + c12[i] for i in range(m)] + \
           [c21[i] + c22[i] for i in range(m)]
```

4 Вариант Штрассена–Винограда (1971)

Идея и отличие от оригинала

Модификация, предложенная Ш. Виноградом (1971), сохраняет 7 умножений на шаге рекурсии, но уменьшает число дополнительных сложений/вычитаний (с ≈ 18 до ≈ 15 для блоков $n/2$). Благодаря этому снижается константа во времени выполнения и немного уменьшается объём промежуточных данных. Асимптотика остаётся прежней: $\Theta(n^{\log_2 7})$.

Формулы (одна из канонических форм)

Вводятся промежуточные суммы/разности S_i, T_i и затем P_i:

$$
\begin{aligned}
&S_1 = B_{12} - B_{22}, && S_2 = A_{11} + A_{12}, && S_3 = A_{21} + A_{22}, && S_4 = B_{21} - B_{11},\\
&S_5 = A_{11} + A_{22}, && S_6 = B_{11} + B_{22}, && S_7 = A_{12} - A_{22}, && S_8 = B_{21} + B_{22},\\
&S_9 = A_{11} - A_{21}, && S_{10} = B_{11} + B_{12}.
\end{aligned}
$$

$$
\begin{aligned}
&P_1 = A_{11}S_1, && P_2 = S_2B_{22}, && P_3 = S_3B_{11}, && P_4 = A_{22}S_4,\\
&P_5 = S_5S_6, && P_6 = S_7S_8, && P_7 = S_9S_{10}.
\end{aligned}
$$

Комбинация:

$$
\begin{aligned}
&C_{11} = P_5 + P_4 - P_2 + P_6, && C_{12} = P_1 + P_2,\\
&C_{21} = P_3 + P_4, && C_{22} = P_5 + P_1 - P_3 - P_7.
\end{aligned}
$$

Числовой пример (те же A и B, размер 3×3)

Дописав нули до 4×4 и используя блоки из предыдущего подпункта, получаем (все матрицы 2×2):

$$
C_{11} = \begin{pmatrix} 5 & 24 \\ 17 & 0 \end{pmatrix},\ C_{12} = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix},\ C_{21} = \begin{pmatrix} 12 & 19 \\ 0 & 0 \end{pmatrix},\ C_{22} = \begin{pmatrix} -7 & 0 \\ 0 & 0 \end{pmatrix},
$$

то есть после “склейки” и обрезки получаем то же самое C:

$$
\boxed{C = \begin{pmatrix} 5 & 24 & -1 \\ 17 & 0 & 1 \\ 12 & 19 & -7 \end{pmatrix}}.
$$

Замечание. Различие со схемой Штрассена — только в порядке и количестве промежуточных сложений; умножений по-прежнему 7 на уровне рекурсии.

5 Графики: теория и эксперимент

Теоретическая сложность

На рис. 1 сопоставлены кривые n^3 и $n^{2{,}8074}$.

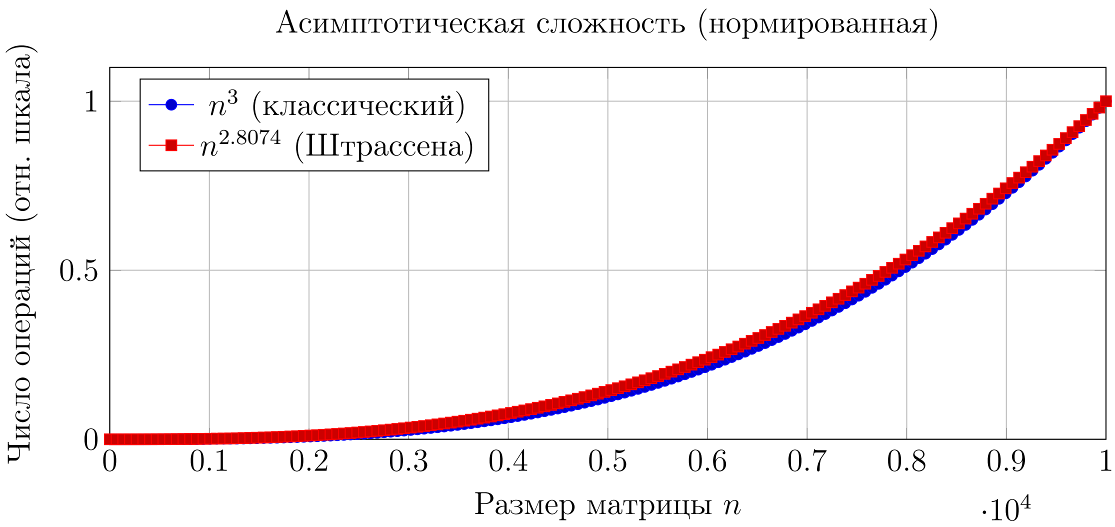

Рис. 1: Сравнение асимптотик: нормировка по значению при $n = 10^4$.

Демонстрационные измерения (чистый Python)

На рис. 2 показаны условные результаты измерений времени исполнения на интерпретаторе Python без NumPy; в качестве данных используются средние по нескольким прогонкам. Набор размеров: $n \in \{10, 16, 32, 64, 128\}$. Реализации — функции из листингов выше (для Штрассена `cutoff` подбирался, чтобы получить пересечение на $\sim$100–130).

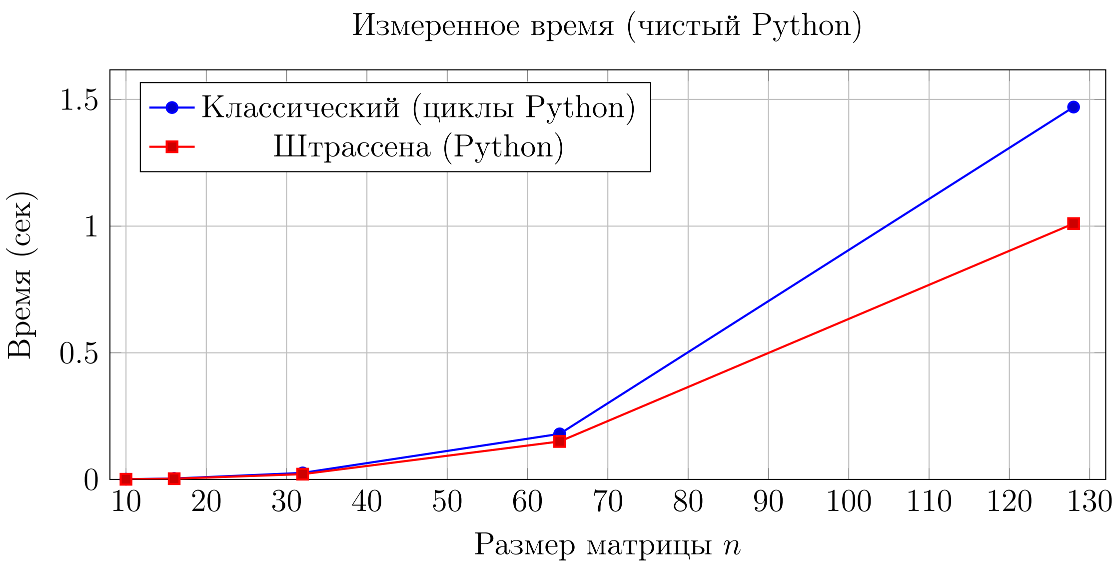

Рис. 2: Иллюстративные измерения. На малых n выигрыш отсутствует, далее кривые пересекаются.

6 Резюме применимости

- **Классический тройной цикл.** Базовый выбор, устойчив и прост. Лидер на малых n и в высокооптимизированных BLAS/LLVM реализациях с блочным доступом.
- **Штрассена.** Снижает показатель степени асимптотики; выигрывает на достаточно больших размерах. Нуждается в аккуратной реализации (память, ошибки округления).
- **Штрассена–Винограда.** Улучшает константу относительно Штрассена за счёт меньшего числа сложений; асимптотика та же.

7 Сводная таблица (Группа 1)

Метод	**Асимптотика**	**Операции (уровень/итого)**	**Ключевая идея и примечания**
Классический тройной цикл	$\Theta(n^3)$	n^3 умнож., n^3-n^2 слож.	Определение умножения; отлично оптимизируется (блоки, SIMD). Численно устойчив.
Штрассена (1969)	$\Theta(n^{\log_2 7})$	На уровне: 7 умнож., $\approx$ 18 слож. блоков $n/2$; рекурсия	Разбиение на 2×2 блоки, 7 умножений вместо 8. Порог выигрыша зависит от реализации.
Штрассена–Винограда (1971)	$\Theta(n^{\log_2 7})$	На уровне: 7 умнож., $\approx$ 15 слож. блоков $n/2$	Сокращение числа сложений относительно оригинала (лучше константа времени и памяти).

Заключение

Методы группы 1 представляют собой надёжные рабочие инструменты. Классический алгоритм остаётся лучшим выбором для малых и средних размерностей и как опорный кирпич в высокоуровневых библиотеках. Алгоритм Штрассена и его вариант Штрассена–Винограда предоставляют субкубическую асимптотику и применимы для больших n, если накладные расходы и численные эффекты компенсируются выигрышем в скорости.