

Improved basis-change transformations for 2×2 matrix multiplication

Hugo Møller Nielsen^{1*}

¹Mathematical Institute for Machine Learning and Data Science (MIDS), Catholic University of Eichstätt-Ingolstadt (KU), Ingolstadt, 85049, Germany.

Corresponding author(s). E-mail(s): hugomn2002@gmail.com;

Abstract

To the best of our knowledge, we present a state-of-the-art change-of-basis algorithm for 2×2 matrix multiplication under the Karstadt–Schwartz-style basis change regime [1]. We do this by introducing basis-change matrices B_U , B_V , and B_W that each require only a single addition and hence give rise to a matrix-multiplication algorithm performing

$$5n^{\log_2 7} - 4n^2 + \frac{3}{4}n^2 \log_2 n,$$

arithmetic operations when multiplying two $n \times n$ matrices, with n a power of 2. This improves on the previous best bound, which had last term $\frac{3}{2}n^2 \log_2 n$.

Keywords: matrix multiplication, addition minimization, additive complexity, Strassen’s algorithm

1 Introduction

The Karstadt–Schwartz-style basis change is a well-studied method, and we will not repeat the method here. For background on the Karstadt–Schwartz-style basis-change method, see [1–4]. Let R be an arbitrary ring, and let $A, B \in R^{2 \times 2}$ be matrices over this ring. We use column-major vectorization, $\vec{A} = [a_{11} \ a_{21} \ a_{12} \ a_{22}]^\top$. Here and below, when counting additions, we will use the convention that a subtraction also counts as an addition, see [4–6]. Let the matrices $U, V, W \in \mathbb{Z}^{7 \times 4}$ satisfy

$$\vec{A}B = W^\top \left((U\vec{A}) \odot (V\vec{B}) \right), \quad (1.1)$$

for all matrices $A, B \in R^{2 \times 2}$. Moreover, let $B_U, B_V, B_W \in \mathbb{Z}^{4 \times 4}$ be invertible over $\mathbb{Z}$. For arbitrary $x \in R^4$, let $\beta_U, \beta_V, \beta_W$ denote the number of additions required to compute respectively $B_U x$, $B_V x$, and $B_W^\top x$. Finally, denote by α_U , α_V , and α_W the number of additions required to compute respectively $\tilde{U}x$, $\tilde{V}x$, and $\tilde{W}^\top y$ for arbitrary $x \in R^4$, and $y \in R^7$, with

$$\tilde{U} = UB_U^{-1}, \quad \tilde{V} = VB_V^{-1}, \quad \tilde{W} = WB_W^{-1}.$$

Theorem 1.1. *For $n \in \mathbb{N}$ an integer-power of 2 we can multiply matrices $A, B \in R^{n \times n}$ in*

$$\left(1 + \frac{\alpha_U + \alpha_V + \alpha_W}{3} \right) n^{\log_2 7} - \frac{\alpha_U + \alpha_V + \alpha_W}{3} n^2 + \frac{\beta_U + \beta_V + \beta_W}{4} n^2 \log_2 n \quad (1.2)$$

arithmetic operations.

Proof See Claim 2.9, Claim 2.12, and Corollary 2.15 in [1]. □

Now consider matrices

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & -1 & 0 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix}, \quad V = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 1 & 1 \\ 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad W = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & -1 & 1 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix},$$

with basis-change matrices

$$S = B_U = B_V = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad S^{-1} = B_W = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \quad (1.3)$$

It is straightforward to verify that the matrix triple U, V, W satisfies equation (1.1), and that B_U, B_V , and B_W are invertible over $\mathbb{Z}$. Moreover, clearly $\beta_U = \beta_V = \beta_W = 1$, giving us the trailing term $\frac{3}{4}n^2 \log_2 n$ in equation (1.2).

We can compute $\tilde{U}, \tilde{V}$, and $\tilde{W}$ as follows

$$\tilde{U} = UB_U^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & 1 \\ -1 & -1 & 0 & 1 \end{bmatrix}, \quad \tilde{V} = VB_V^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \tilde{W} = WB_W^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

The linear transformations induced by $\tilde{U}$ and $\tilde{V}$ can be computed using 3 additions each as follows:

$$s_1 = x_4 - x_2, \quad s_2 = s_1 - x_3, \quad s_3 = s_1 - x_1.$$

such that

$$\begin{aligned} \tilde{U}x &= [x_1 \ x_3 \ x_2 \ s_1 \ x_4 \ s_2 \ s_3]^\top \\ \tilde{V}x &= [x_1 \ x_2 \ s_3 \ s_1 \ x_4 \ s_2 \ x_3]^\top \end{aligned}$$

for all $x \in R^4$. Therefore $\alpha_U = \alpha_V = 3$. The linear transformation induced by $\tilde{W}^\top$ can be computed using 6 additions as follows:

$$\begin{aligned} t_1 &= M_5 - M_4, & t_2 &= t_1 - M_2, \\ y_1 &= M_1 + M_2, & y_2 &= t_2 - M_3, \\ y_3 &= M_5 - M_7, & y_4 &= t_2 + M_6, \end{aligned}$$

such that

$$\tilde{W}^\top M = [y_1 \ y_2 \ y_3 \ y_4]^\top$$

for any $M \in R^7$. From which it follows that $\alpha_W = 6$.

By Theorem 1.1 we can thus multiply two $n \times n$ matrices using a Karstadt–Schwartz-style basis-change in

$$5n^{\log_2 7} - 4n^2 + \frac{3}{4}n^2 \log_2 n$$

arithmetic operations, using the basis-change induced by the matrices B_U, B_V , and B_W defined in equation (1.3).

A Verification

The following Python program provides a simple verification of the scheme.

Listing 1: Small verification script (verify.py).

```
1  """
2  Artisanally hand-crafted code
3  """
4  import numpy as np
5  import sympy as sp
6
7  inv = np.linalg.inv
8  S = np.array([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 1, 1]]).astype(int)
9  assert(not any(((S @ inv(S)).astype(int)) - np.eye(4)).flatten())) # inv over Z sanity
10                                     check
11  U = np.array([ [1, 0, 0, 0], [0, 0, 1, 0], [0, 1, 0, 0], [0, -1, 1, 1], [0, 0, 1, 1],
12                [0, -1, 0, 1], [-1, -1, 1, 1] ]).astype(int)
13  V = np.array([ [1, 0, 0, 0], [0, 1, 0, 0], [-1, -1, 1, 1], [0, -1, 1, 1], [0, 0, 1,
14                1], [0, -1, 0, 1], [0, 0, 1, 0] ]).astype(int)
15  W = np.array([ [1, 0, 0, 0], [1, -1, 1, -1], [0, -1, 0, 0], [0, -1, 1, -1], [0, 1, 0,
16                1], [0, 0, -1, 1], [0, 0, -1, 0] ]).astype(int)
17
18  A_vec = sp.Matrix([sp.Symbol(f"A_{j}{i}", commutative=False) for i in range(1,3) for j
19                    in range(1,3)])
20  B_vec = sp.Matrix([sp.Symbol(f"B_{j}{i}", commutative=False) for i in range(1,3) for j
21                    in range(1,3)])
22
23  # sanity check that U,V,W encodes matrix multiplication
24  print("#" * 32)
25  hadamard = sp.matrices.dense.matrix_multiply_elementwise
26  print(f"{sp.simplify(W.T @ hadamard((U @ A_vec), (V @ B_vec)))=}")
27
28  U_tilde = U @ inv(S).astype(int)
29  V_tilde = V @ inv(S).astype(int)
30  W_tilde = W @ S
31  print("#" * 32)
32  print(f"{U_tilde=}")
33  print(f"{V_tilde=}")
34  print(f"{W_tilde.T=}")
35
36  def compute_U(x):
37      s_1 = x[3] - x[1] # 1 add
38      s_2 = s_1 - x[2] # 2 add
39      s_3 = s_1 - x[0] # 3 add
40      return sp.Matrix([x[0], x[2], x[1], s_1, x[3], s_2, s_3])
41  def compute_V(x):
42      s_1 = x[3] - x[1] # 1 add
43      s_2 = s_1 - x[2] # 2 add
44      s_3 = s_1 - x[0] # 3 add
45      return sp.Matrix([x[0], x[1], s_3, s_1, x[3], s_2, x[2]])
46  def compute_W(M):
47      t_1 = M[4] - M[3] # 1 add
48      t_2 = t_1 - M[1] # 2 add
49      y_1 = M[0] + M[1] # 3 add
50      y_2 = t_2 - M[2] # 4 add
51      y_3 = M[4] - M[6] # 5 add
52      y_4 = t_2 + M[5] # 6 add
53      return sp.Matrix([y_1, y_2, y_3, y_4])
54
55  print("#" * 32)
56  x = sp.Matrix([sp.Symbol(f"x_{i}") for i in range(1,5)])
57  y = sp.Matrix([sp.Symbol(f"x_{i}") for i in range(1,8)])
58  print("Should be a bunch of zeros only:")
59  print(f"{U_tilde @ x - compute_U(x)=}")
60  print(f"{V_tilde @ x - compute_V(x)=}")
61  print(f"{W_tilde.T @ y - compute_W(y)=}")
```

Declaration of use of generative AI. GPT-6-Astra was used for finding this result. The manuscript is prepared by me, with minor language editing and grammar help from **the machine**. I take full responsibility for the final manuscript.

References

- [1] Schwartz, O., Vaknin, N.: Pebbling game and alternative basis for high performance matrix multiplication. *SIAM Journal on Scientific Computing* **45**(6), C277–C303 (2023) <https://doi.org/10.1137/22M1502719>
- [2] Karstadt, E., Schwartz, O.: Matrix multiplication, a little faster. *Journal of the ACM* **67**(1), 1–31 (2020) <https://doi.org/10.1145/3364504>
- [3] Karstadt, E., Schwartz, O.: Matrix multiplication, a little faster. In: *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures. SPAA '17*, pp. 101–110. Association for Computing Machinery, New York, NY, USA (2017). <https://doi.org/10.1145/3087556.3087579> . <https://doi.org/10.1145/3087556.3087579>
- [4] Nielsen, H.: A 52-Addition, Rank-23 Scheme for Exact 3×3 Matrix Multiplication (2026). <https://doi.org/10.48550/arXiv.2609.06588> . <https://arxiv.org/abs/2609.06588>
- [5] Van Zee, F.G., Smith, T.M.: Inducing complex matrix multiplication via the 3m and 4m methods. *FLAME Working Note 81*, The University of Texas at Austin, Austin, TX, USA (October 2016). <https://www.cs.utexas.edu/%7Eflame/pubs/flawn81.pdf>
- [6] Bläser, M.: Fast matrix multiplication. *Theory of Computing Library, Graduate Surveys* **5**, 1–60 (2013) <https://doi.org/10.4086/toc.gs.2013.005>