

The 2M Multiplication Algorithm for Complex Matrices

Peter Caday
NVIDIA Corporation
pcaday@nvidia.com

October 7, 2026

Abstract

Complex matrix multiplication is typically computed using 4 real matrix multiplications (GEMMs) of the same size. The well-known 3M multiplication algorithm reduces this cost to 3 real GEMMs, together with quadratic time pre- and post-processing steps. In this paper, we reduce 3M to 2M for matrices with integer real and imaginary parts, performing complex GEMM with only 2 real GEMMs of the same size, along with quadratic time pre- and post-processing. For floating-point matrices, 2M multiplication combines naturally with the Ozaki-II scheme, yielding a practical, high-performance algorithm for computing a complex floating-point GEMM in roughly twice the time of a real GEMM of the same size. As corollaries, we derive new algorithms for symmetric rank- k updates (SYRK/HERK) that internally use full rectangular GEMMs.

1 Introduction

The classical approach to multiply complex matrices $A = A_r + iA_i \in \mathbb{C}^{m \times k}$ and $B = B_r + iB_i \in \mathbb{C}^{k \times n}$ is to expand the product into real matrix multiplications:

$$C = AB = (A_r B_r - A_i B_i) + i(A_r B_i + A_i B_r).$$

Let us call this the 4M algorithm, following [23], since it involves 4 real matrix multiplications. 4M multiplication typically forms the backbone of complex floating-point matrix multiplications (GEMMs) in high-performance libraries.

The well-known 3M algorithm reduces this cost to 3 real matrix multiplications, by first computing $A_r + A_i$ and $B_r + B_i$:

$$C = (A_r B_r - A_i B_i) + i((A_r + A_i)(B_r + B_i) - A_r B_r - A_i B_i). \quad (1)$$

Note that while (1) is mathematically exact, in floating-point arithmetic the additions $A_r + A_i$, $B_r + B_i$ can reduce numerical stability [9]. Hence, numerical libraries typically offer 3M GEMMs as an opt-in feature.

For scalar complex products, it can be proven that 3 real multiplications is indeed minimal [6,24]. In this paper, we will show that we can do better for matrix products.

2 2M Complex Multiplication

To start, we consider input matrices $A = A_r + iA_i$, $B = B_r + iB_i$ with integral real and imaginary parts, and their product $C = AB$. Suppose m is an integer chosen large enough that $|\operatorname{Re}(c_{ij})|, |\operatorname{Im}(c_{ij})| \leq \frac{m}{2}$ for each entry c_{ij} of C . Then, if we know $C \bmod m$, we can evidently recover C .

Now further suppose that m is odd and there is an integer s such that $s^2 \equiv -1 \pmod{m}$, in effect a “real surrogate” for i . It is well-known from number theory that such an s exists iff m is a product of primes congruent to 1 mod 4 [8, Thm. 80]. We call such an m a *2M modulus*.

To multiply AB , first perform the 3M-like preprocessing steps

$$\begin{aligned} A_+ &\equiv A_r + sA_i \pmod{m}, & B_+ &\equiv B_r + sB_i \pmod{m}, \\ A_- &\equiv A_r - sA_i \pmod{m}, & B_- &\equiv B_r - sB_i \pmod{m}. \end{aligned} \tag{2a}$$

Next, perform the 2 real multiplications

$$C_+ \equiv A_+B_+ \pmod{m}, \quad C_- \equiv A_-B_- \pmod{m}. \tag{2b}$$

Then we will see shortly that

$$C = AB \equiv 2^{-1}(C_- + C_+) + 2^{-1}s(C_- - C_+)i \pmod{m}, \tag{2c}$$

where $2^{-1} = (m+1)/2$ is the inverse of 2 modulo m .

Together, equations (2a–2c) form the *2M multiplication algorithm*. As with 3M, the new algorithm requires $\Theta(mn + mk + nk)$ scalar operations in addition to the matrix multiplications. We start by proving correctness.

Proposition 1. *Equation (2c) holds for every A, B .*

Proof I. Expanding (2b) yields

$$\begin{aligned} C_{\pm} &\equiv A_rB_r \pm s(A_rB_i + A_iB_r) + s^2A_iB_i \\ &\equiv (A_rB_r - A_iB_i) \pm s(A_rB_i + A_iB_r) \pmod{m}. \end{aligned}$$

Hence

$$\begin{aligned} C_+ + C_- &\equiv 2(A_rB_r - A_iB_i) \pmod{m}, \\ C_+ - C_- &\equiv 2s(A_rB_i + A_iB_r) \pmod{m}. \end{aligned}$$

Both 2 and s are invertible modulo m , and $s^{-1} \equiv -s \pmod{m}$, from which (2c) follows immediately. $\square$

Proof I, while valid, is not very illuminating. In the next section, we will invoke some basic algebraic number theory in order to better motivate 2M multiplication and provide a second, deeper proof of Proposition 1.

2.1 Algebraic Underpinnings

Algebraically, A, B, C are matrices over the ring of Gaussian integers $\mathbb{Z}[i] = \{x + yi \mid x, y \in \mathbb{Z}\}$. Gaussian integers have a rich algebraic structure, and in particular, they form a Euclidean domain, meaning that there is a well-defined notion of division with remainder. Modular arithmetic and the Chinese remainder theorem (CRT) are likewise valid in $\mathbb{Z}[i]$ [8].

The 2M algorithm then follows from two observations:

- First, a 2M modulus m factors into a product of Gaussian integers $w\overline{w}$. Hence we can compute $AB \pmod{m}$ by first evaluating $AB \pmod{w}$ and $AB \pmod{\overline{w}}$, then applying CRT.
- Second, there is an isomorphism $\mathbb{Z}[i]/w\mathbb{Z}[i] \cong \mathbb{Z}/m\mathbb{Z}$. Under this isomorphism, the complex residues $A, B \pmod{w}, \overline{w}$ become real-valued, and the complex products $AB \pmod{w}$ and $AB \pmod{\overline{w}}$ can be re-expressed as real multiplications mod m .

Proof II of Proposition (1). Since $m \equiv 1 \pmod{4}$ there exist positive coprime $u, v \in \mathbb{Z}$ such that $m = u^2 + v^2$ (cf. [8, Thm. 366]). Letting $w = u + vi \in \mathbb{Z}[i]$, it follows that $m = w\bar{w}$. Note that $u^2v^{-2} \equiv -1 \pmod{m}$, so $s \equiv \pm uv^{-1} \pmod{m}$. By negating v if necessary, we can force $s \equiv -uv^{-1} \pmod{m}$.

Next, we note that the mapping $\varphi_+ : i \mapsto s$ induces an isomorphism of quotient rings $\mathbb{Z}[i]/w\mathbb{Z}[i] \cong \mathbb{Z}/m\mathbb{Z}$. Indeed, as a homomorphism φ_+ is well-defined since $s^2 \equiv -1 \pmod{m}$, and it has a trivial inverse $\mathbb{Z}/m\mathbb{Z} \hookrightarrow \mathbb{Z}[i]/m\mathbb{Z}[i] \rightarrow \mathbb{Z}[i]/w\mathbb{Z}[i]$, well-defined as $w \mid m$. Similarly, the mapping $\varphi_- : i \mapsto -s$ is an isomorphism from $\mathbb{Z}[i]/\bar{w}\mathbb{Z}[i]$ to $\mathbb{Z}/m\mathbb{Z}$.

We can now interpret the 2M algorithm in the light of these isomorphisms. Clearly, $A_{\pm} = \varphi_{\pm}(A)$, $B_{\pm} = \varphi_{\pm}(B)$, so the preprocessing steps (2a) are equivalent to reducing A, B modulo $w, \bar{w}$ and applying $\varphi_{\pm}$. Next, the multiplication steps (2b) compute

$$\begin{aligned} C_+ &= \varphi_+(A \bmod w) \cdot \varphi_+(B \bmod w) = \varphi_+(C \bmod w), \\ C_- &= \varphi_-(A \bmod \bar{w}) \cdot \varphi_-(B \bmod \bar{w}) = \varphi_-(C \bmod \bar{w}). \end{aligned}$$

More simply, since the $\varphi_{\pm}^{-1}$ are inclusions, $C_+ \equiv C \bmod w$ and $C_- \equiv C \bmod \bar{w}$.

Finally, (2c) proves to be a specialization of the Chinese Remainder Theorem in $\mathbb{Z}[i]$, reconstructing $C \bmod m$ from $C \bmod w$ and $C \bmod \bar{w}$. Namely, the CRT states

$$C \equiv \chi_+ \bar{w} C_+ + \chi_- w C_- \pmod{m},$$

where $\chi_+ \equiv \bar{w}^{-1} \pmod{w}$, $\chi_- \equiv w^{-1} \pmod{\bar{w}}$. We have

$$\chi_+ \equiv (u - vi)^{-1} \equiv (2u)^{-1} \pmod{w} \quad \Rightarrow \quad \chi_+ \bar{w} \equiv 2^{-1} u^{-1} (u - vi) \equiv 2^{-1} (1 - is) \pmod{m}.$$

Similarly $\chi_- w \equiv 2^{-1} (1 + is) \pmod{m}$, establishing (2c). $\square$

2.2 Non-2M Moduli

We conclude by noting a negative result: no 2M-like algorithm is possible for non-2M odd prime moduli. However, the 3M method (1) is still valid, being applicable in any ring, and is therefore optimal.

Proposition 2. *If $m \equiv 3 \pmod{4}$ is prime, there are no constants p_{ij}, q_{ij}, r_{ij} such that*

$$AB \equiv (C_1 r_{11} + C_2 r_{21}) + (C_1 r_{12} + C_2 r_{22}) i \pmod{m} \quad (3)$$

holds for all A, B , where

$$C_j = A_j B_j, \quad A_j = p_{i1} A_r + p_{i2} A_i, \quad B_j = q_{i1} B_r + q_{i2} B_i. \quad (4)$$

Proof. This is an immediate corollary of the general multiplicative complexity results of Fiduccia and Zalcstein [6]. Because -1 is not a quadratic residue for any $m \equiv 3 \pmod{4}$, the polynomial $s^2 + 1$ is irreducible modulo m , and $\mathbb{Z}[i]/m\mathbb{Z}[i] \cong (\mathbb{Z}/m\mathbb{Z})[s]/(s^2 + 1) \cong \mathbb{F}_{m^2}$ is a degree-2 extension of $\mathbb{Z}/m\mathbb{Z}$. Theorem 8 in [6] then states that multiplication in $\mathbb{Z}[i]/m\mathbb{Z}[i]$ can be performed by 3 multiplications in $\mathbb{Z}/m\mathbb{Z}$ by non-constants, and no fewer. In particular, it is impossible to choose p_{ij}, q_{ij}, r_{ij} to make (3)–(4) true, even in the special case where the input matrices A, B are scalars. $\square$

3 2M Multiplication In Floating Point

3.1 The Ozaki-II Scheme

To apply 2M to non-integer GEMMs, we employ the Ozaki-II scheme [16, 19], a family of algorithms for computing floating-point GEMMs to arbitrary accuracy. Ozaki-II works by scaling and quantizing its inputs to integers, then performing fast integer matrix multiplication with modular arithmetic. By leveraging the dedicated low-precision matrix multiplication units available on modern GPUs, Ozaki-II can often significantly outperform standard floating-point GEMM while maintaining comparable accuracy. Moreover, the level of accuracy may be tuned: either reduced, for additional performance; or increased, to provide additional accuracy beyond that available with a standard floating-point implementation.

We briefly restate the method here. Given real matrices $\mathbf{A} = (\mathbf{a}_{ih}) \in \mathbb{R}^{m \times k}$, $\mathbf{B} = (\mathbf{b}_{hj}) \in \mathbb{R}^{k \times n}$, we choose per-row and per-column scaling factors $\sigma \in \mathbb{R}^m$, $\tau \in \mathbb{R}^n$ for $\mathbf{A}$ and $\mathbf{B}$, respectively, and quantize to integer matrices

$$A = \left(\underset{\mathbb{Z}}{\text{round}}(\mathbf{a}_{ih} \sigma_i^{-1}) \right) \in \mathbb{Z}^{m \times k}, \quad B = \left(\underset{\mathbb{Z}}{\text{round}}(\mathbf{b}_{hj} \tau_j^{-1}) \right) \in \mathbb{Z}^{k \times n}.$$

Here, $\text{round}_S(\cdot)$ represents rounding to a nearest¹ element of a set S . Throughout, we employ boldface letters to denote real- and complex-valued variables and italics for their quantized integer counterparts.

Next, we choose a collection of coprime moduli $m_1, \dots, m_N \in \mathbb{Z}_+$ whose product M is large enough that each element of the integer product $C = AB$ has absolute value at most $\frac{M}{2} - 1$. Then $C = (c_{ij})$ can be uniquely recovered from its residue modulo M , and we do so by computing the product residues

$$C_\ell \equiv (A \bmod m_\ell)(B \bmod m_\ell) \pmod{m_\ell}, \quad \ell = 1, \dots, N, \quad (5)$$

and applying the Chinese remainder theorem (CRT) elementwise:

$$C \equiv \sum_{\ell=1}^N C_\ell \cdot \left(\frac{M}{m_\ell} \bmod m_\ell \right)^{-1} \frac{M}{m_\ell} \pmod{M}. \quad (6)$$

Finally, we re-apply the scaling factors to reconstruct an approximation $\tilde{\mathbf{C}} = (c_{ij} \sigma_i \tau_j)$ to the exact product $\mathbf{C} = (\mathbf{c}_{ij})$.

Ozaki-II derives its speed from computing the products (5) using dedicated low-precision matrix multiplication hardware, which guides the choice of the moduli m_ℓ (e.g., $m_\ell \leq 2^8$ for 8-bit integer GEMMs).

The accuracy of the Ozaki-II method, much like the accuracy of floating-point GEMMs, is an involved topic and depends strongly on the input data, as well as the scaling factors σ, τ [1, 5, 17, 20, 22]. However, for *fixed* inputs $\mathbf{A}, \mathbf{B}$, elementwise error $|c_{ij} - c_{ij} \sigma_i \tau_j|$ scales as $O(M^{-1/2})$ for properly chosen σ, τ , so it is often useful to view M as an accuracy parameter. We will study accuracy and the choice of σ, τ in more detail in section 4.

¹In the original Ozaki-II implementation, round-to-zero is used instead of round-to-nearest [19]. Rounding to nearest provides an additional bit of accuracy, but any rounding is valid.

Algorithm 1 – 2M Ozaki-II

- 1: Fix a set of coprime 2M moduli $m_1, \dots, m_N$; let $M = \prod_{\ell=1}^N m_\ell$.
 - 2: Choose scaling factors $\sigma \in \mathbb{R}^m, \tau \in \mathbb{R}^n$.
 - 3: Quantize input matrices: $A = (\text{round}_{\mathbb{Z}[i]}(\sigma_i^{-1} \mathbf{a}_{ih}))$, $B = (\text{round}_{\mathbb{Z}[i]}(\tau_j^{-1} \mathbf{b}_{hj}))$.
 - 4: **for** $\ell = 1, \dots, N$ **do**
 - 5: Reduce A, B modulo m_ℓ .
 - 6: Compute $C_\ell = C \bmod m_\ell$ with (2a–2c).
 - 7: **end for**
 - 8: Reconstruct $C \bmod M$ from $(C \bmod m_1, \dots, C \bmod m_N)$ via the Chinese Remainder Theorem.
 - 9: Lift C to $\mathbb{Z}[i]^{m \times n}$ by choosing representatives with real and imaginary parts in $[-M/2, M/2)$.
 - 10: Reconstruct $\tilde{\mathbf{C}} = (c_{ij} \sigma_i \tau_j) \in \mathbb{C}^{m \times n}$.
-

3.2 Complex Ozaki-II With 2M

Ozaki-II extends readily to complex-valued inputs $\mathbf{A} = \mathbf{A}_r + i\mathbf{A}_i \in \mathbb{C}^{m \times k}$, $\mathbf{B} = \mathbf{B}_r + i\mathbf{B}_i \in \mathbb{C}^{k \times n}$. In Uchino et al.’s algorithm [18], we first choose (real) scaling factors σ, τ and quantize $\mathbf{A}_r, \mathbf{A}_i, \mathbf{B}_r, \mathbf{B}_i$ to integer matrices as above. The resulting integer matrices are reduced modulo each m_ℓ , multiplied, and then the real and imaginary parts of C are reconstructed separately using CRT. Scaling factors may be chosen using straightforward extensions of any of the existing real algorithms [17, 19] or based on a desired integer bit width 7.

For better performance, Uchino et al. apply the 3M method to each small integer multiplication, after modular reduction²; since 3M is an algebraic identity, it is valid modulo any m_ℓ . By substituting 2M multiplication for 3M, we immediately derive a 2M Ozaki-II algorithm for complex floating-point GEMMs (Algorithm 1).

In Algorithm 1 we note several points of customization:

- *Choice of scaling factors (line 2).* σ, τ may be chosen using one of Uchino et al.’s scaling algorithms (“fast” or “accurate”) [18], or based on a fixed integer bit width (equation (7)). See Section 4 for more details.
- *Rounding method (line 3).* The choice of rounding method is not essential to the algorithm. We suggest round-to-nearest for best accuracy, though round-to-zero may also be used, as in [18, 19].
- *Low-precision matrix multiplication (line 6).* Individual matrix products modulo m_ℓ may be performed with any available accelerated matrix hardware. We will consider int8, fp8, and fp4 specifically in Section 5.
- *CRT reconstruction algorithm (line 8).* Numerous algorithms are available for CRT reconstruction, including direct reconstruction via (6) and mixed-radix conversion [10]. Uchino et al. propose direct reconstruction in double-double arithmetic, though this loses precision for small entries of C , when $\log_2 |c_{ij}|/M \lesssim -53 + \lceil \log_2 N \rceil$.

We may also readily hybridize the 2M and 3M Ozaki-II methods. Algorithm 2 presents a hybrid approach that supports any set of coprime moduli, using 2M multiplication for moduli m_ℓ that

²It is also possible to apply 3M *prior* to modular reduction, but this has the disadvantage of requiring high-precision integer arithmetic in 3M’s pre- and post-processing steps.

satisfy the 2M conditions (odd, with a square root of -1), and 3M multiplication for the remaining moduli. For a practical implementation, this hybrid approach can attain higher precision (larger M) than Algorithm 1, since more moduli are available. In section 5, we will discuss how to pick modulus sets in Algorithm 2 for best performance; generally speaking, we prefer 2M moduli to non-2M moduli, but there are several subtleties.

Algorithm 2 – Hybrid 2M/3M Ozaki-II

- 1: Fix a set of arbitrary coprime moduli $m_1, \dots, m_N$; let $M = \prod_{\ell=1}^N m_\ell$.
 - 2: Choose scaling factors $\sigma \in \mathbb{R}^m, \tau \in \mathbb{R}^n$.
 - 3: Quantize input matrices: $A = (\text{round}_{\mathbb{Z}[i]}(\sigma_i^{-1} \mathbf{a}_{ih}))$, $B = (\text{round}_{\mathbb{Z}[i]}(\tau_j^{-1} \mathbf{b}_{hj}))$.
 - 4: **for** $\ell = 1, \dots, N$ **do**
 - 5: Reduce A, B modulo m_ℓ .
 - 6: **if** m_ℓ is a 2M modulus **then**
 - 7: Compute $C_\ell = C \bmod m_\ell$ with (2a–2c).
 - 8: **else**
 - 9: Compute C_ℓ with (1).
 - 10: **end if**
 - 11: **end for**
 - 12: Reconstruct $C \bmod M$ from $(C \bmod m_1, \dots, C \bmod m_N)$ via CRT.
 - 13: Lift C to $\mathbb{Z}[i]^{m \times n}$ by choosing representatives with real and imaginary parts in $[-M/2, M/2)$.
 - 14: Reconstruct $\tilde{C} = (c_{ij} \sigma_i \tau_j) \in \mathbb{C}^{m \times n}$.
-

3.3 Algebraic Variants of 2M Ozaki-II

As before, we can obtain additional insight into 2M Ozaki-II by further examining the algebra. After quantization, Algorithm 1 is presented as a two-level method. At the top level we reduce integer inputs A, B modulo each real m_ℓ , multiply modulo m_ℓ with 2M, and reconstruct the product C with CRT. At the lower level, as shown in Section 2.1, the 2M multiplication steps further factor each m_ℓ as a product $w_\ell \bar{w}_\ell$, reduce modulo w_ℓ and $\bar{w}_\ell$, multiply, and apply CRT.

Hence, we may reinterpret Algorithm 1 as *the usual Ozaki-II scheme operating over the Gaussian integers, with conjugate pairs of complex moduli*. Similarly, Uchino et al.’s 3M algorithm reduces to Ozaki-II over the Gaussian integers, but with *real* moduli, while hybrid 2M/3M Ozaki-II (Algorithm 2) uses both real and complex moduli.

This algebraic perspective is useful because it allows us to easily derive additional Ozaki-II variants. In particular, 2M multiplication and Algorithms 1 and 2 have direct analogues where we replace $\mathbb{Z}[i]$ by another imaginary quadratic number ring. A natural choice is the Eisenstein integer ring $\mathbb{Z}[e^{2\pi i/3}] = \{a + be^{2\pi i/3} \mid a, b \in \mathbb{Z}\}$, which forms a regular hexagonal grid in the complex plane. By varying the number ring, we vary not only the discretization grid, but also the set of valid 2M moduli. Asymptotically, half of all primes are 2M moduli w.r.t. any given imaginary quadratic number ring, by Dirichlet’s theorem [15]. In practice, however, since moduli are generally bounded by the input range of hardware matrix multiplication units, specific rings may perform slightly better than others, as we will see in Section 5.2.

4 Accuracy of Ozaki-II Complex GEMMs

In this section, we investigate in more detail the accuracy characteristics of Algorithms 1 and 2. Before starting, we make two notes:

- (a) Since the modular multiplications, whether 2M or 3M, are always performed exactly, Algorithms 1 and 2 produce identical results to each other and to Uchino et al.'s complex 3M Ozaki-II, when given the same scaling factors σ, τ and using the same rounding methods and CRT reconstruction algorithms. Hence, all accuracy results in this section are independent of the choice of complex integer multiplication algorithm.
- (b) We assume that CRT reconstruction is performed exactly, then correctly rounded to floating point. Under this assumption, the sole source of error in Algorithms 1 and 2 is quantization error in the scaled integer approximations $\mathbf{A} \approx \text{diag}(\sigma)A$ and $\mathbf{B} \approx B \text{diag}(\tau)$.

4.1 Absolute Error

Prior Results for Real-Valued GEMM

Accuracy of integer approximation algorithms for real-valued GEMMs has been studied by several authors. Uchino et al. provide absolute error bounds for Ozaki-II [22], using an approximate CRT reconstruction, as well for the related Ozaki-I integer approximation algorithm [20]. Abdelfattah et al. [1] likewise derive absolute error bounds for Ozaki-I.

The usual starting point is to consider scaling factors σ, τ that preserve a given number of bits q in each row of $\mathbf{A}$ and column of $\mathbf{B}$ ³. We first record the maximum elements in each row and column, rounding up to powers of 2:

$$\begin{aligned} \mathbf{a}_{i,\max} &= \max_{1 \leq h \leq k} |\mathbf{a}_{ih}|, & \mathbf{b}_{\max,j} &= \max_{1 \leq h \leq k} |\mathbf{b}_{hj}|, \\ \bar{\mathbf{a}}_{i,\max} &= 2^{\lceil \log_2 \mathbf{a}_{i,\max} \rceil}, & \bar{\mathbf{b}}_{\max,j} &= 2^{\lceil \log_2 \mathbf{b}_{\max,j} \rceil}, \end{aligned}$$

and set

$$\sigma_i = 2^{-q} \bar{\mathbf{a}}_{i,\max}, \quad \tau_j = 2^{-q} \bar{\mathbf{b}}_{\max,j}, \quad (7)$$

resulting in quantized matrices A, B with entries in the range $[-2^q, 2^q]$. The use of power-of-2 scaling factors both simplifies the implementation and avoids loss of accuracy in the final scaling $c_{ij}\sigma_i\tau_j$. With round-to-nearest, the elementwise quantization errors are bounded by

$$|\mathbf{a}_{ih} - \sigma_i a_{ih}| \leq \frac{\sigma_i}{2}, \quad |\mathbf{b}_{hj} - \tau_j b_{hj}| \leq \frac{\tau_j}{2}, \quad (8)$$

from which we can derive (see [5, eq. (3.1)]) the forward error bound⁴

$$|c_{ij} - \sigma_i \tau_j c_{ij}| \leq 2^{-q} k (1 + 2^{-q-2}) \bar{\mathbf{a}}_{i,\max} \bar{\mathbf{b}}_{\max,j} \quad (9)$$

Since the quantized data A, B lies in $[-2^q, 2^q]$, the integer product C is bounded elementwise by $2^{2q}k$, providing a lower bound $M_{\min} = 2^{2q+1}k + 1$ for the product of moduli M . Together with (9) this implies that for *fixed* inputs $\mathbf{A}, \mathbf{B}$, there is an $O(M^{-1/2})$ error dependence:

$$|c_{ij} - \sigma_i \tau_j c_{ij}| = O(M^{-1/2}). \quad (10)$$

³For some inputs there are advantages in asymmetric quantization, with $\mathbf{A}$ and $\mathbf{B}$ quantized to different bit counts; see [1, 5].

⁴There is a factor of 2 difference between (9) and [5, eq. (3.1)] since we use round-to-nearest here, rather than truncation.

Complex GEMM

Handling complex-valued $\mathbf{A}$, $\mathbf{B}$ is a straightforward generalization of the real case. Defining σ , τ as in (7), the quantization error (8) must be augmented by a factor of $\sqrt{2}$ to account for both real and imaginary parts:

$$|\mathbf{a}_{ih} - \sigma_i a_{ih}| \leq \frac{\sqrt{2}}{2} \sigma_i, \quad |\mathbf{b}_{hj} - \tau_j b_{hj}| \leq \frac{\sqrt{2}}{2} \tau_j. \quad (11)$$

The rest of the derivation in [5] continues as before, yielding the forward error bound

$$|\mathbf{c}_{ij} - \sigma_i \tau_j c_{ij}| \leq 2^{-q+1/2} k (1 + 2^{-q-3/2}) \bar{\mathbf{a}}_{i,\max} \bar{\mathbf{b}}_{\max,j}. \quad (12)$$

As in the real case, after q is chosen we have an immediate lower bound $M_{\min} = 2^{2q+2}k + 1$, with an additional factor of 2 from complex multiplication, and once again (10) holds.

4.2 Matching Floating-Point Accuracy

Prior Results for Real-Valued GEMM

Given the absolute error bound (9), the natural question is how to choose q to achieve GEMM accuracy comparable to standard floating-point arithmetic. Let Q be the floating-point mantissa width, including the implicit leading 1. Evidently, q should be at least Q , but in general additional bits are required, driven by exponent variations within a row of $\mathbf{A}$ or column of $\mathbf{B}$ [1]. Schwarz et al. [17] capture this effect more precisely with the *exponent span capacity* (ESC), defined as:

$$\text{esc}(\mathbf{A}, \mathbf{B}) = \max_{i,j} \left[\left\lfloor \log_2 \max_h |\mathbf{a}_{ih}| \right\rfloor + \left\lfloor \log_2 \max_h |\mathbf{b}_{hj}| \right\rfloor - \left\lfloor \log_2 \max_h |\mathbf{a}_{ih} \mathbf{b}_{hj}| \right\rfloor \right].$$

Schwarz et al. suggested setting $q = Q + \text{esc}(\mathbf{A}, \mathbf{B}) + 1$. Fasi et al. [5, Section 3.1] carry the analysis further, proving that by choosing $q = q^* = Q + \text{esc}(\mathbf{A}, \mathbf{B}) + 3$ the *worst-case* error bound (9) is no larger than Jeannerod and Rump’s worst-case floating-point error bound for dot products [13]. We recall the latter bound here as we will revisit it in the complex domain:

Theorem 1 (Jeannerod and Rump [13], Theorem 4.2). *Let $x, y \in \mathbb{F}^k$ be length- k vectors of finite values in a floating point system $\mathbb{F}$ of mantissa width Q (including leading 1). Let $\text{float}(x \cdot y) \in \mathbb{F}$ denote the result of evaluating the inner product $x \cdot y$ using a sequence of correctly rounded multiplications and additions, in any order. Then*

$$|\text{float}(x \cdot y) - x \cdot y| \leq k 2^{-Q} (|x| \cdot |y|), \quad (13)$$

assuming no underflow or overflow conditions are encountered, where the absolute values on x, y are taken elementwise.

It is important to note that using $q = Q + \text{esc}(A, B) + 3$ mantissa bits for Ozaki-II does not ensure the result will always have smaller error than a standard floating-point GEMM; for a given choice of input data, Ozaki-II may be more or less accurate than a standard floating-point GEMM. However, Fasi et al.’s analysis does provide a formal justification for using ESC to determine q .

Complex GEMM

ESC results for real GEMM readily generalize to the complex domain. The main ingredient is a complex analogue of Jeannerod and Rump's error bound, for standard 4M multiplication.

Proposition 3. *Let $x = x_r + ix_i$, $y = y_r + iy_i$, where $x_r, x_i, y_r, y_i \in \mathbb{F}^k$ are length- k vectors with finite values. Let $\hat{z} = \text{float}(x_r \cdot y_r - x_i \cdot y_i) + i \cdot \text{float}(x_r \cdot y_i + x_i \cdot y_r)$, where $\text{float}(x_1 \cdot y_1 \pm x_2 \cdot y_2)$ denotes the result of evaluating a sum or difference of inner products using a sequence of correctly rounded multiplications and additions, in any order. Then*

$$|\hat{z} - x \cdot y| \leq ck2^{-Q}(|x| \cdot |y|), \quad c = 2\sqrt{2}, \quad (14)$$

in the absence of underflow and overflow.

The constant $c = 2\sqrt{2}$ is likely not optimal. For the special case $n = 1$ it is known that the optimal constants are $c = \sqrt{5}$ without fused multiply-add (FMA) [2], and $c = 2$ with FMA [12]. A proof that $c = 2\sqrt{2}$ suffices is at the end of the section.

Once we have Proposition 3, we may reuse Fasi et al.'s derivation to prove:

Proposition 4. *If $q \geq q^* = Q + \text{esc}(\mathbf{A}, \mathbf{B}) + 2$, then the worst-case error bound (12) for Algorithms 1 and 2 is no greater than the 4M floating-point worst-case error bound (14).*

Proof of Proposition 4. Let $\hat{c}_{ij}$ be the 4M floating-point dot product of the i^{th} row of A and j^{th} column of B . By Proposition 3,

$$|c_{ij} - \hat{c}_{ij}| \leq ck \cdot 2^{-Q} \sum_{h=1}^k |a_{ih} b_{hj}|. \quad (15)$$

Substituting $q = Q + \text{esc}(\mathbf{A}, \mathbf{B}) + 2$ into the right-hand side of (12),

$$\begin{aligned} |c_{ij} - \sigma_i \tau_j c_{ij}| &\leq \frac{\sqrt{2}}{4} k \cdot 2^{-Q} \cdot 2^{-\text{esc}(\mathbf{A}, \mathbf{B})} (1 + 2^{-q-3/2}) \mathbf{a}_{i,\max} \mathbf{b}_{\max,j} \\ &\leq \frac{\sqrt{2}}{4} k \cdot 2^{-Q} \cdot 2^{-\lfloor \log_2 \mathbf{a}_{i,\max} \rfloor} 2^{-\lfloor \log_2 \mathbf{b}_{\max,j} \rfloor} 2^{\lfloor \log_2 \max_h |a_{ih} b_{hj}| \rfloor} (1 + 2^{-q-3/2}) \bar{\mathbf{a}}_{i,\max} \bar{\mathbf{b}}_{\max,j} \\ &\leq \sqrt{2} k \cdot 2^{-Q} \cdot \max_h |a_{ih} b_{hj}| \cdot (1 + 2^{-q-3/2}) \\ &\leq ck \cdot 2^{-Q} \sum_h |a_{ih} b_{hj}|. \end{aligned}$$

This is the desired result. Note that the inequality remains true even if we take the stricter bound $c = 2$ in (14). $\square$

Proof of Proposition 3. Rewrite $\hat{z}$ as $\text{float}(X \cdot Y_r) + i \text{float}(X \cdot Y_i)$, where

$$X = \begin{bmatrix} x_r \\ x_i \end{bmatrix}, \quad Y_r = \begin{bmatrix} y_r \\ -y_i \end{bmatrix}, \quad Y_i = \begin{bmatrix} y_i \\ y_r \end{bmatrix}.$$

Applying Theorem 1 to each real dot product gives

$$\begin{aligned} |\text{float}(X \cdot Y_r) - \text{Re}(x \cdot y)| &\leq 2k \cdot 2^{-Q} (|X| \cdot |Y_r|) = 2k \cdot 2^{-Q} \sum_{h=1}^k [|x_{r,h}| |y_{r,h}| + |x_{i,h}| |y_{i,h}|] \leq 2k \cdot 2^{-Q} (|x| \cdot |y|), \\ |\text{float}(X \cdot Y_i) - \text{Im}(x \cdot y)| &\leq 2k \cdot 2^{-Q} (|X| \cdot |Y_i|) = 2k \cdot 2^{-Q} \sum_{h=1}^k [|x_{r,h}| |y_{i,h}| + |x_{i,h}| |y_{r,h}|] \leq 2k \cdot 2^{-Q} (|x| \cdot |y|). \end{aligned}$$

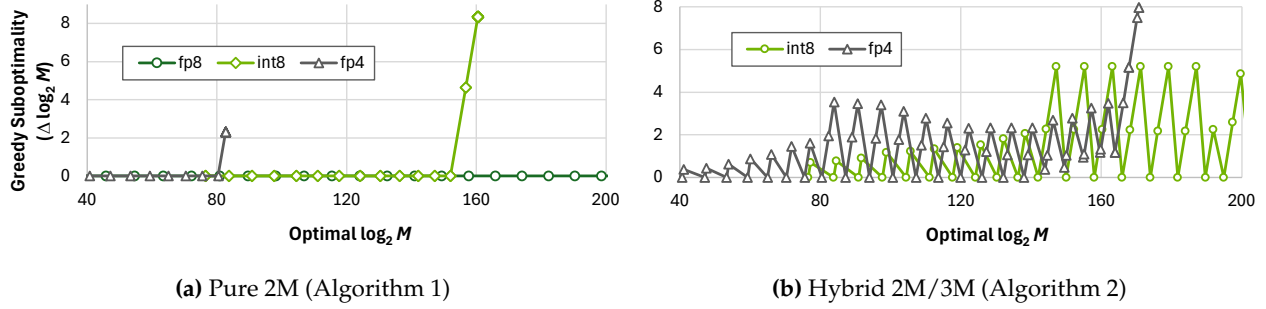


Figure 1 – Suboptimality of greedy modulus selection. For each optimal set of moduli, a greedy set of moduli is computed with the same maximum cost N_r . The gap $\Delta \log_2 M$ between optimal and greedy $\log_2 M$ values is displayed.

Here we have used Cauchy-Schwarz for the final inequality on both sides. Equation (14) immediately follows. $\square$

5 Implementing 2M ZGEMM

As concrete examples of Algorithms 1 and 2, let us consider complex double-precision GEMM (ZGEMM) with 8-bit integer or floating-point GEMMs underneath. We start by discussing general methodology for choosing moduli.

5.1 Selecting Modulus Sets

Given a minimum accumulator precision $M_{\min}$ dictated by the accuracy requirements and input data, our high-level goal is to find a set of moduli $m_1, \dots, m_N$, with product $M \geq M_{\min}$, minimizing runtime. Absolute runtime, of course, is highly sensitive to hardware and implementation details. As a starting point, we may model runtime (for fixed inputs $\mathbf{A}, \mathbf{B}, \sigma, \tau$) as some increasing function of N_r , the number of real low-precision matrix multiplications⁵.

Under this model, our original goal reduces to maximizing M for a given N_r . We assume that moduli will be taken from a predetermined set $\mathcal{M}$ of candidate moduli, based on the low-precision matrix multiplications used (e.g., $\mathcal{M} = \{2, \dots, 256\}$ for 8-bit integers). For 2M Ozaki-II, $\mathcal{M}$ is further restricted to 2M moduli.

For real Ozaki-II, where $N_r = N$, existing implementations generally select moduli via a greedy algorithm [19, 21], choosing each m_ℓ to be the largest element of $\mathcal{M}$ coprime to $m_1, \dots, m_{\ell-1}$. The same greedy approach extends immediately to 2M Ozaki-II. Alternatively, one may exhaustively search for the optimal modulus set with a branch-and-bound method. Figure 1(a) compares the $\log_2 M$ values for greedy and exhaustive searches for several data types of interest. In each case, greedy selection is in fact optimal until moduli become scarce.

To extend the greedy algorithm to hybrid 2M/3M Ozaki-II (Algorithm 2), we take into account the relative efficiency of 2M vs. 3M by ranking moduli by *bits per real multiplication*, $r(m) = (\log_2 m)/p(m)$, where $p(m) = 2$ if m is a 2M modulus, and 3 otherwise. Then m_ℓ is chosen to be the maximize $r(m)$ among moduli coprime to $m_1, \dots, m_{\ell-1}$. As Figure 1(b) shows, greedy

⁵For hybrid 2M/3M, this model neglects the differences in pre- and post-processing operation counts between 2M and 3M. In practice, pre- and post-processing are typically implemented in separate memory-bound kernels, and the relatively small operation counts of both 2M and 3M are hidden behind much larger memory or cache latencies. Note that the total memory traffic is an increasing function of N_r .

ℓ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2M	241	233	229	221	205	197	193	181	173	157	149	137	113	109	101
Hybrid	(same as 2M)														
3M	256	255	253	251	247	241	239	233	229	227	223	217	211	199	197
ℓ	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2M	97	89	73	61	53	37	29	-	-	-	-	-	-	-	-
Hybrid	(same as 2M)					256	253	251	249	239	227	37	223	217	211
3M	193	191	181	179	173	167	163	157	151	149	139	137	131	127	113

Table 1 – Greedy int8 moduli sets for 2M, 3M, and hybrid 2M/3M GEMM, with 2M moduli italicized.

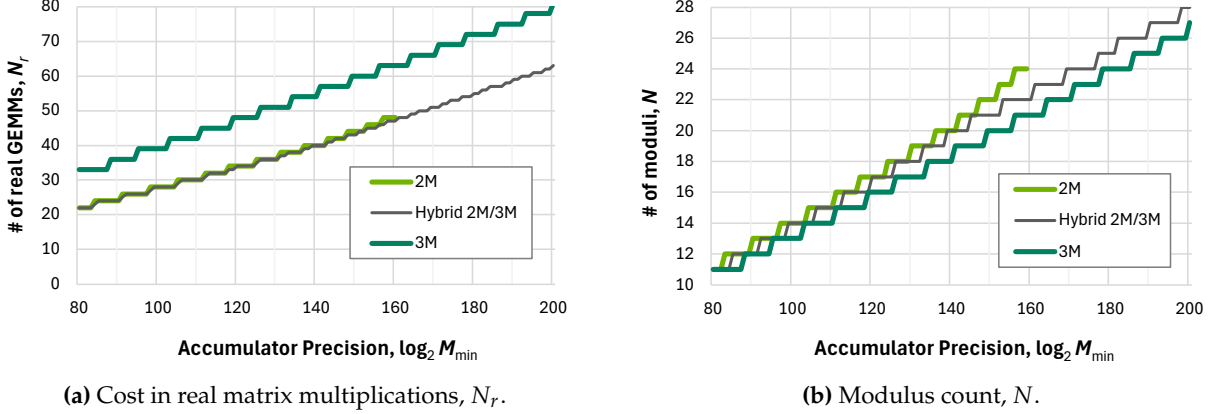


Figure 2 – Modulus count N and real matrix multiplication count N_r , as functions of required accumulator precision, $\log_2(M_{\min})$, for optimal sets of int8 moduli.

selection is optimal for some values of N_r but shows quantization effects that reduce its efficacy at others. A simple example illustrates the issue: consider 8-bit integer moduli with a budget of $N_r = 5$. The greedy algorithm selects two 2M moduli $\{241, 233\}$, wasting one of the available real multiplications, while the optimal set $\{241, 256\}$ uses the additional real multiplication to trade a 2M modulus for a slightly larger non-2M modulus.

5.2 ZGEMM with 8-bit Integer GEMMs

The original Ozaki-II scheme targets 8-bit integer (int8) matrix multiplication hardware, which accommodates moduli up to 256 without recourse to multiword arithmetic. Table 1 lists the modulus sequences generated the greedy method of section 5.1 for the 2M, 3M, and hybrid 2M/3M algorithms.

We notice immediately that the 2M moduli diminish more rapidly than 3M moduli, as expected. The relative sparsity of M_{2M} w.r.t. M_{real} reflects the fact that asymptotically only half of the prime numbers are valid 2M moduli, as observed in Section 3.3. Moreover, after $N = 22$ moduli ($\log_2 M \approx 152.1$) the greedy method has exhausted the available 2M moduli, though by exhaustive search it is possible to extend to $N = 24$ coprime moduli ($\log_2 M \approx 160.4$). If higher precision is needed (e.g., due to large ESC) we must use hybrid 2M/3M, Algorithm 2.

Figure 2 compares the cost of the different algorithms as we vary the required precision $M_{\min}$. Because of the sparsity of 2M moduli, the new algorithms may require an additional 1–2 moduli for a given precision compared to 3M, but nevertheless maintain a significant reduction in overall cost.

Note that Figure 2 is computed using *optimal* modulus sets rather than the greedy choices from Table 1; however, in all cases greedy selection requires at most one additional modulus compared to the optimum.

As mentioned in Section 3.3, the set of 2M moduli may be varied by discretizing to different imaginary quadratic number rings. Indeed, for 8-bit integer moduli, the 2M moduli for the Eisenstein ring $\mathbb{Z}[e^{2\pi i/3}]$ are marginally denser than 2M moduli in $\mathbb{Z}[i]$, achieving $\log_2 M \approx 167.4$ before exhausting 2M moduli (versus 160.4 for $\mathbb{Z}[i]$). Similarly, for hybrid 2M/3M, we may obtain up to approximately 3.7 additional bits of accumulator precision for the same number of real GEMMs by using $\mathbb{Z}[e^{2\pi i/3}]$ in Algorithm 2. However, this advantage is offset by slightly more complicated quantization/dequantization steps; in particular, quantization involves rounding each input in the complex plane to the nearest point on a hexagonal lattice.

5.3 ZGEMM with 4-bit and 8-bit Floating-Point GEMMs

On some GPUs, such as the NVIDIA B300, low-precision floating-point GEMMs are much faster in hardware than integer GEMMs, leading to the development of fp8 [21] and fp4 [11] variants of Ozaki-II, which use the E4M3 and E2M1 formats [3, 14], respectively. Modular integer matrices are embedded in the target floating-point format (possibly rescaled), and the floating-point results converted back to integers, carefully avoiding loss of accuracy due to rounding.

Because of the narrower mantissa fields in fp4 and fp8 formats, moduli are restricted to a much smaller range when using single-word arithmetic—for E2M1, $m_\ell \in \{2, \dots, 11, 13\}$, while for E4M3, Uchino et al. limit⁶ $m_\ell \leq 33$. As the available moduli are not enough to obtain even single-precision accuracy, existing work uses multi-word arithmetic (multiple fp4/fp8 matrix multiplications per modulus) to accommodate larger moduli. More specifically, for each modulus m_ℓ we choose a base $b_\ell \bmod m_\ell$ and split A and B so that

$$A \equiv A_\ell^{(1)}b_\ell + A_\ell^{(0)} \pmod{m_\ell}, \quad B \equiv B_\ell^{(1)}b_\ell + B_\ell^{(0)} \pmod{m_\ell},$$

where $A_\ell^{(i)}, B_\ell^{(i)}$ are representable in fp4/fp8. We can then compute $AB \bmod m_\ell$ with 4 fp4/fp8 multiplies modulo m_ℓ . For certain choices of m_ℓ it is possible to use either Karatsuba multiplication or square-optimized multiplication to compute AB with only 3 multiplies, as detailed in [21].

Both 2M and hybrid 2M/3M Ozaki-II carry over to this setting. We first apply 2M (or 3M) to reduce the problem of computing a complex product $AB \bmod m$ to 2 or 3 real products $A_q B_q \bmod m$, then apply any double-word multiplication algorithm (schoolbook, Karatsuba, or square-optimized) to reduce each of these real products to a set of fp4 or fp8 matrix multiplications.

With Uchino et al.’s double-word fp8 algorithm, the maximum modulus $m = 1089$ is relatively large, allowing 2M Ozaki-II to attain sufficiently high precision ($\log_2 M \approx 350.2$) that hybrid 2M/3M is practically unnecessary. Hayashi et al.’s double-word fp4 has a much more limited range, with maximum modulus $m = 169$. Pure 2M can only achieve $\log_2 M \approx 82.7$, so hybrid 2M/3M is always necessary to obtain double precision accuracy.

⁶In fact, single-word E4M3 multiplication can be extended to certain moduli larger than 33, even though $[-16, 16]$ is the largest consecutive integer range representable in E4M3. This is because it suffices to find E4M3 representatives of every residue class modulo m_ℓ , as Hayashi et al. [11] already leverage for E2M1. Extending E4M3 to larger moduli has some additional complications that are outside the scope of this paper; for simplicity, we follow Uchino et al.’s modulus restrictions here.

6 2M Rank- k Updates⁷

The 2M method also yields computationally efficient methods for both real and complex BLAS-3 rank- k update routines (SYRK, HERK). We recall these routines' definitions, omitting alpha and beta scaling for simplicity:

- SYRK: $C = AA^T$, $A \in \mathbb{R}^{n \times k}$.
- HERK: $C = AA^H$, $A \in \mathbb{C}^{n \times k}$.

In each case C is (Hermitian) symmetric, so only the lower or upper triangle of C is computed and stored. In theory, the symmetry of C halves the operation count (e.g., $n^2k + o(n^2k)$ flops for SYRK vs. $2n^2k$ for a same-sized GEMM), but library implementations generally cannot obtain the full 2 $\times$ speedup. The reason is that SYRK-like routines are typically implemented with a modified GEMM kernel that masks its output to the desired triangle of the output matrix C [7]. As each kernel operates on a fixed-size tile of C , there is wasted computation near the diagonal, where tiles lie partly outside the desired triangle. Shrinking tile size reduces waste, but decreases arithmetic intensity, reducing performance.

With 2M, it is possible to implement symmetric products using full, unmasked GEMMs, while still maintaining the expected 2 $\times$ speedup. The key idea is that Hermitian symmetry in the problem induces a symmetry relation in the real integer products $C_{\pm}$, allowing us to compute only half as many integer GEMMs. We outline the methods below, which may be applied in either the integer domain (modulo m) or in floating point, in conjunction with Ozaki-II.

HERK Writing $A = A_r + iA_i$ and applying 2M, we have

$$\begin{aligned} C_+ &= (A_r + sA_i)(A_r - sA_i)^T \bmod m, \\ C_- &= (A_r - sA_i)(A_r + sA_i)^T \bmod m. \end{aligned}$$

In particular $C_- = C_+^T$, so we only need to compute a single real multiplication C_+ for each m , providing the expected 2 $\times$ speedup while using only full GEMMs.⁸

Real SYRK We first reduce an arbitrary (real) $n \times k$ SYRK AA^T to an $n \times (k/2)$ HERK ZZ^H . We may assume k is even by zero-padding if necessary. Split A evenly in the k dimension: $A = [A_1 \mid A_2]$, and form $Z = A_1 + iA_2 \in \mathbb{C}^{n \times k/2}$. Now

$$ZZ^H = (A_1A_1^T + A_2A_2^T) + i(A_2A_1^T - A_1A_2^T) = AA^T + i(A_2A_1^T - A_1A_2^T),$$

so that $AA^T = \text{Re}(ZZ^H)$. We compute ZZ^H with the 2M HERK algorithm above, discarding the imaginary part.

For the new 2M-based SYRK/HERK algorithms, we pay $\Theta(nk)$ additional pre-processing operations and $\Theta(n^2)$ post-processing operations, compared to the standard masked-GEMM approach, which itself wastes $\Theta(nk \cdot \max(t_R, t_C))$ operations, where $t_R \times t_C$ is the kernel tile size. Hence the benefits of 2M depend on the problem size, in addition to machine- and implementation-dependent details. In addition, when using Ozaki-II, the 2M SYRK algorithm may increase the number of required moduli, particularly at higher accumulator precisions, as shown earlier in Figure 2(b).

⁷The author would like to thank Robert Parrish for pointing out this method of reducing SYRK to 2M GEMM.

⁸Dumas et al. [4] present a separate 2M HERK algorithm that reduces AA^H to one real SYRK and one real GEMM, over an arbitrary commutative ring. Asymptotically, their algorithm requires $\frac{3}{2}$ as many operations, but allows arbitrary moduli.

7 Computational Results

To demonstrate the efficacy of the 2M method, we present performance and accuracy results for int8-emulated 2M ZGEMM on a single NVIDIA B200 GPU. All results are with $\alpha = 1$, $\beta = 0$, and use the Gaussian integer version of 2M.

Accuracy Metric Accuracy is measured as maximum elementwise relative error, compared against a reference quadruple-precision implementation. Real inputs are taken from Uchino et al.’s data distribution [19]: $\mathbf{a}_{ih} = u_{ih} \exp(\phi v_{ih})$, with u_{ih} uniformly drawn from $[-\frac{1}{2}, \frac{1}{2})$ and $v_{ih} \sim \mathcal{N}(0, 1)$, and similarly for $\mathbf{B}$. The parameter $\phi > 0$ is a difficulty parameter, with higher ϕ values producing wider exponent ranges and larger ESC values. Complex input data follows a similar distribution $\mathbf{a}_{ih} = \frac{1}{2}u_{ih} \exp(\phi v_{ih} + 2\pi i w_{ih})$, where v_{ih} is as before, and u_{ih}, w_{ih} are uniformly drawn from $[0, 1)$.

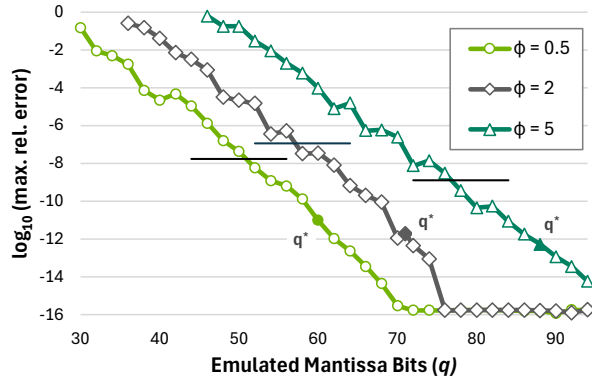
Performance Metric Performance of all implementations is measured in effective teraflops per second (TFLOP/s), calculated using the floating point operation count of the *baseline algorithm*: i.e., $8mnk$ operations for ZGEMM and $2mnk$ operations for DGEMM. Effective TFLOP/s numbers do not reflect actual operation counts for Ozaki-II GEMMs (and do not indicate any accuracy guarantees), but provide a standard metric for comparing implementations.

Algorithms We benchmark an internal implementation of Algorithm 2 (hybrid 2M/3M) with a greedy modulus set. Note that this is identical to Algorithm 1 for $N \leq 20$ moduli. For comparison, we also test Uchino et al.’s 3M-based GEMMu18 library⁹ as well as an internal implementation of the 3M Ozaki-II algorithm. `cublasZgemv` from cuBLAS 13.1, with emulation disabled, is used as the reference 4M floating-point GEMM. Scaling factors are determined using fixed mantissa bit count scaling (equation (7)) for accuracy tests, and Uchino et al.’s fast scaling method [19] for performance tests.

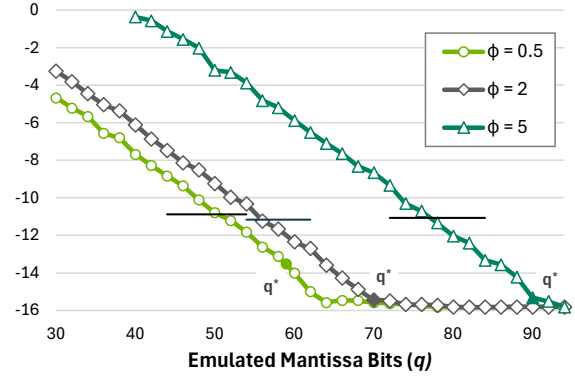
7.1 Accuracy

We begin by computationally verifying accuracy for 2M and hybrid 2M/3M Ozaki-II ZGEMM. Figure 3(b) plots maximum relative error as a function of mantissa bit count q for several choices of difficulty parameter $\phi \in \{0.5, 2, 5\}$, for matrices of size $m = n = k = 4096$. The modulus count N is deduced using the worst-case bound $M \geq 2^{2q+2}k$, ranging from $N = 10$ at $q = 30$ to $N = 28$ at $q = 94$.

The roughly linear slope in the log-plot reflect the expected $O(2^{-q})$ error decay (equation (12)), and is qualitatively consistent¹⁰ with Ozaki-II DGEMM (cf. Figure 3(a)). As ϕ increases, the ESC increases, as does the number of mantissa bits required to match floating-point GEMM. For each choice of ϕ , we also compute the ESC-based mantissa bit count q^* from Fasi et al. [5] and Proposition 4; across all test cases q^* consistently overpredicts the required number of mantissa bits.

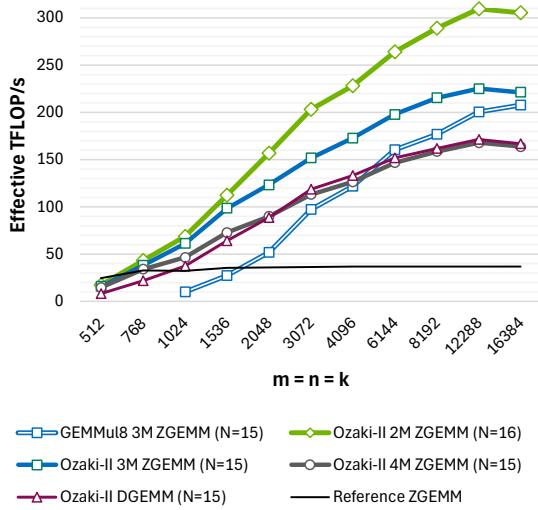


(a) DGEMM

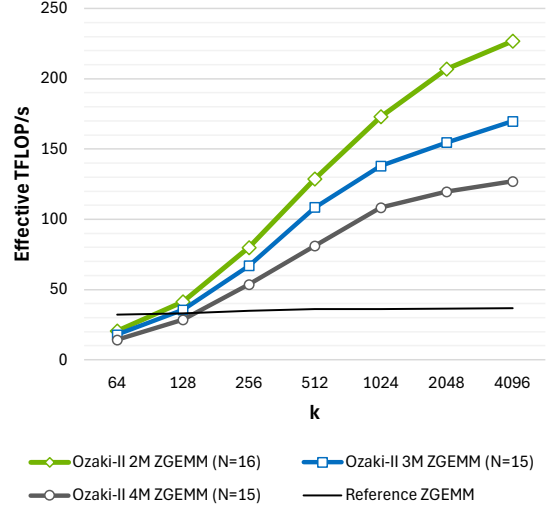


(b) Hybrid 2M/3M ZGEMM

Figure 3 – Maximum relative error of Ozaki-II DGEMM and hybrid 2M/3M Ozaki-II ZGEMM for input data of varying difficulties ϕ , as a function of mantissa bits, q . Horizontal lines show the accuracy attained by standard floating-point DGEMM/ZGEMM for each choice of ϕ , while q^* marks the value of q suggested by ESC (Proposition 4).

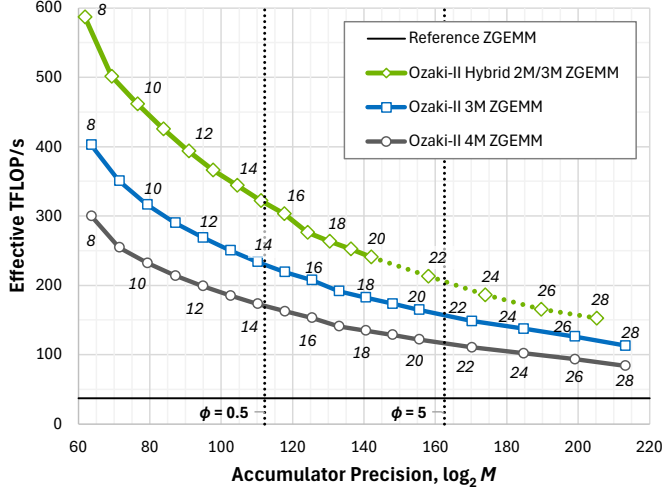


(a) Square sizes



(b) Small- k cases, $m = n = 4096$.

Figure 4 – ZGEMM performance benchmarks with fixed accuracy $\log_2 M \approx 117$ and variable sizes, on an NVIDIA B200 card.



(a) Performance vs. accumulator precision, $m = n = k = 16384$. Marked data points show modulus count N . Dotted vertical lines mark the precisions required to match reference floating-point GEMM accuracy for $\phi = 0.5, 5$.

$m = n = k$	1024	1536	2048	3072	4096	6144	8192	12288	16384
Modulus Count N	8	10	12	14	16	18			
	3.1	2.2	2.1	2	2	1.7			
	4.7	3.5	3.3	3.1	3	2.5			
	6.3	4.8	4.5	4.1	3.9	3.3			
	8.7	6.7	6.1	5.4	5	4.3			
	10.1	7.9	7.1	6.2	5.8	5			
	11.7	9.1	8	7.1	6.4	5.6			
	13.2	10.4	9	8	7.2	6.3			
	14.7	11.5	10	8.7	7.8	6.8			
	15.4	12	10.4	9.1	8.1	7.1			

(b) Ozaki-II 2M ZGEMM speedups over reference ZGEMM for varying square sizes and modulus counts.

Figure 5 – Additional 2M and hybrid 2M/3M ZGEMM performance data.

7.2 Performance

Figure 4(a) shows 2M, 3M, and 4M Ozaki-II ZGEMM performance across a range of problem sizes, with fixed accumulator precision $\log_2 M \approx 117$. For 2M, $N = 16$ moduli are needed to attain this precision; other Ozaki-II algorithms need one fewer modulus, $N = 15$. In addition, we compare performance in small- k scenarios (Figure 4(b)) to make any differences in 2M, 3M, and 4M post-processing cost more prominent. Uchino et al.’s fast scaling algorithm [19] was used for all Ozaki-II implementations.

At this precision, 2M ZGEMM is up to $8.4\times$ faster than reference 4M fp64 ZGEMM. It is uniformly faster than 3M and 4M Ozaki-II on all test cases, with speedups up to $1.38\times$ over 3M, and $1.86\times$ over 4M. 2M also achieves an average $1.83\times$ speedup in effective TFLOP/s compared to Ozaki-II DGEMM at the same precision. That is, in absolute time a single ZGEMM costs approximately $4/1.83 \approx 2.19$ DGEMMs, approaching the 2-DGEMM theoretical limit. The gap arises primarily because 2M requires an additional modulus to achieve a similar accuracy level (cf. Figure 2(b)).

In Figure 5(a) we fix the problem size to $m = n = k = 16384$ and vary N to show the tradeoff between performance and accumulator precision. Hybrid 2M/3M shows similar speedups over 3M ZGEMM and Ozaki-II DGEMM across the range of N , with performance up to 586 TFLOP/s ($15\times$ reference ZGEMM) at $N = 8$. Figure 5(b) provides a more detailed heatmap of speedups over reference ZGEMM across a range of sizes and modulus counts.

⁹Tested version: <https://github.com/RIKEN-RCSS/GEMMu18/commit/3115e709>

¹⁰The large maximum relative errors seen in Figure 3(a) as compared to Figure 3(b) are due to numerical cancellation, which leads to poorly-conditioned dot products. Ill-conditioning is less likely for random complex data since real and imaginary parts must simultaneously exhibit significant cancellation.

8 Conclusion

With each successive generation of CPU and GPU hardware, low-precision GEMM throughput continues to outpace that of traditional single and double precision arithmetic. The Ozaki-II scheme and other emulation methods take advantage of this trend, using integer quantization and small integer GEMMs to accelerate floating-point GEMMs.

In this paper, we have shown that this integer-quantized approach opens up new algorithms for complex GEMMs that leverage the algebraic structure of quadratic integer rings. In particular, 2M multiplication reduces a complex GEMM to two real GEMMs of the same size, instead of the three or four real GEMMs required by previous algorithms. For rank- k updates, 2M reduces a complex rank- k update ($C = AA^H$) to a real GEMM of the same size, and a real rank- k update ($C = AA^T$) to a real GEMM of half the size. The new 2M algorithm is easy to implement, and achieves practical speedups that closely match these theoretical predictions.

Acknowledgements

The author is grateful for helpful reviews from numerous colleagues, which greatly improved this paper.

References

- [1] A. ABDELFAH, J. DONGARRA, M. FASI, M. MIKAITIS, AND F. TISSEUR, *Analysis of floating-point matrix multiplication computed via integer arithmetic*, 2026.
- [2] R. BRENT, C. PERCIVAL, AND P. ZIMMERMANN, *Error bounds on complex floating-point multiplication*, *Mathematics of Computation*, 76 (2007), pp. 1469–1481.
- [3] B. DARVISH ROUHANI, N. GAREGRAT, T. SAVELL, A. MORE, K.-N. HAN, M. ZHAO, RITCHIE AND HALL, J. KLAR, E. CHUNG, Y. YU, M. SCHULTE, R. WITTIG, I. BRATT, N. STEPHENS, J. MILANOVIC, J. BROTHERS, P. DUBEY, M. CORNEA, A. HEINECKE, A. RODRIGUEZ, M. LANGHAMMER, S. DENG, M. NAUMOV, P. MICEKVICIUS, M. SIU, AND C. VERRILLI, *OCP microscaling (MX) specification*. Open Compute Project, 2023.
- [4] J.-G. DUMAS, C. PERNET, AND A. SEDOGLAVIC, *Some fast algorithms multiplying a matrix by its adjoint*, *Journal of Symbolic Computation*, 115 (2023), pp. 285–315.
- [5] M. FASI, T. MARY, AND M. MIKAITIS, *Sharper bounds on the required fixed-point precision for emulating floating-point matrix multiplication*. Preprint, June 2026.
- [6] C. M. FIDUCCIA AND Y. ZALCSTEIN, *Algebras having linear multiplicative complexities*, *J. ACM*, 24 (1977), pp. 311–331.
- [7] K. GOTO AND R. VAN DE GEIJN, *High-performance implementation of the level-3 blas*, *ACM Trans. Math. Softw.*, 35 (2008).
- [8] G. H. HARDY AND E. M. WRIGHT, *An introduction to the theory of numbers*, Oxford University Press, 6th ed., 2008.
- [9] N. J. HIGHAM, *Stability of a method for multiplying complex matrices with three real matrix multiplications*, *SIAM Journal on Matrix Analysis and Applications*, 13 (1992), pp. 681–687.

- [10] C. H. HUANG, *A fully parallel mixed-radix conversion algorithm for residue number applications*, IEEE Transactions on Computers, C-32 (1983), pp. 398–402.
- [11] S. ICHIRO HAYASHI, D. MUKUNOKI, T. HOSHINO, AND T. KATAGIRI, *DGEMM with Ozaki scheme I/II on FP4 tensor cores: A base-13 E2M1 limb representation*, 2026.
- [12] C.-P. JEANNEROD, P. KORNERUP, N. LOUVET, AND J.-M. MULLER, *Error bounds on complex floating-point multiplication with an fma*, Mathematics of Computation, 86 (2017), pp. 881–898.
- [13] C.-P. JEANNEROD AND S. M. RUMP, *Improved error bounds for inner products in floating-point arithmetic*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 338–344.
- [14] P. MICIKEVICIUS, S. OBERMAN, P. DUBEY, M. CORNEA, A. RODRIGUEZ, I. BRATT, R. GRISENTHWAITE, N. JOUPPI, C. CHOU, A. HUFFMAN, M. SCHULTE, R. WITTIG, D. JANI, AND S. DENG, *OCP 8-bit floating point specification (OFP8)*. Open Compute Project, 2023.
- [15] I. NIVEN, H. ZUCKERMAN, AND H. MONTGOMERY, *An introduction to the theory of numbers*, John Wiley & Sons, 1991.
- [16] K. OZAKI, Y. UCHINO, AND T. IMAMURA, *Ozaki scheme II: A GEMM-oriented emulation of floating-point matrix multiplication using an integer modular technique*, The International Journal of High Performance Computing Applications, (2026).
- [17] A. SCHWARZ, A. ANDERS, C. BROWER, H. BAYRAKTAR, J. GUNNELS, K. CLARK, R. G. XU, S. RODRIGUEZ, S. CAYROLS, P. TABASZEWSKI, AND V. PODLOZHNYUK, *Guaranteed DGEMM accuracy while using reduced precision tensor cores through extensions of the Ozaki scheme*, in Proceedings of the Supercomputing Asia and International Conference on High Performance Computing in Asia Pacific Region, SCA/HPCAsia '26, New York, NY, USA, 2026, Association for Computing Machinery, pp. 91–101.
- [18] Y. UCHINO, Q. MA, T. IMAMURA, K. OZAKI, AND P. L. GUTSCHE, *Emulation of complex matrix multiplication based on the Chinese remainder theorem*, 2025.
- [19] Y. UCHINO, K. OZAKI, AND T. IMAMURA, *High-performance and power-efficient emulation of matrix multiplication using INT8 matrix engines*, in Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC Workshops '25, ACM, Nov. 2025, pp. 1824–1831.
- [20] Y. UCHINO, K. OZAKI, AND T. IMAMURA, *Performance enhancement of the ozaki scheme on integer matrix multiplication unit*, The International Journal of High Performance Computing Applications, 39 (2025), pp. 462–476.
- [21] —, *Double-precision matrix multiplication emulation via Ozaki-II scheme with FP8 quantization*, 2026.
- [22] —, *Error analysis of matrix multiplication emulation using Ozaki-II scheme*, 2026.
- [23] F. G. VAN ZEE AND T. M. SMITH, *Inducing complex matrix multiplication via the 3m and 4m methods*, FLAME Working Note 81, University of Texas at Austin, 2016.
- [24] S. WINOGRAD, *On multiplication of 2×2 matrices*, Linear algebra and its applications, (1971).