Greatly improved higher-order product formulae for quantum simulation

Mauro E. S. Morales,[1, 2] Pedro C. S. Costa,[3] Daniel K. Burgarth,[3] Yuval R. Sanders,[2, 3] and Dominic W. Berry[3]

[1] *Centre for Quantum Computation and Communication Technology, University of Technology Sydney, Sydney, NSW 2007, AU*
[2] *Centre for Quantum Software and Information, University of Technology Sydney, Sydney, NSW 2007, AU*
[3] *School of Mathematical and Physical Sciences, Macquarie University, Sydney, NSW 2109, AU*

(Dated: October 31, 2022)

Quantum algorithms for simulation of Hamiltonian evolution are often based on product formulae like Trotter. The fractal method of Suzuki gives a systematic way to find arbitrarily high-order product formulae, but results in a large number of exponentials. On the other hand, an alternative way, with a minimum number of exponentials, for 6th and 8th order product formulae is given by Yoshida. In this work, we not only extended Yoshida's method to 10th order, but we also found hundreds of new 8th order product formulae. Moreover, we found an 8th order product formula that is over 1000 times more accurate than the best product formula of Yoshida.

CONTENTS

I. INTRODUCTION

The Lie-Trotter product formula is commonly used in quantum algorithms for Hamiltonian simulation, where one can approximate the Hamiltonian evolution of $H = A + B$ in terms of exponentials of A and B when these operators do not commute. For short time one can use the symmetric formula $e^{-iHt} = e^{-iAt/2}e^{-iBt}e^{-iAt/2} + \mathcal{O}(t^3)$, and longer times are simulated by breaking the evolution into many repetitions of a short time.

Better performance is provided by higher-order product formulae with error scaling as $\mathcal{O}(t^{k+1})$, where k is the order. A systematic method to produce arbitrarily high order formulae is the fractal method of Suzuki [1, 2], which has found applications in Hamiltonian simulation [3].

Recent work has shown that despite its simplicity, the Lie-Trotter product formula can compete with other Hamiltonian simulation algorithms due to the low error that it achieves in practice [4]. Methods based on linear combinations of unitaries [5] or quantum signal processing [6] give complexity logarithmic in the inverse error, but the error is not required to be extremely small, meaning those methods do not provide a large advantage.

Moreover, Trotter formulae are expected to be relevant for both noisy intermediate-scale quantum computation and fault-tolerant computation. It is then of great importance to seek efficient implementations of product formulae as it can have a great impact on the efficiency of Hamiltonian simulation algorithms in practice.

The downside of the Suzuki method to generate higher-order formulae is that the number of exponential operators to implement it grows very rapidly. Suzuki's product formulae are usually assumed in quantum computing, but they can be greatly improved upon. An alternative method by Yoshida [7] can be used to obtain product formulae with a smaller number of exponentials. This method is based on an ansatz written as the product of symmetric product formulae of 2nd order, then by imposing certain conditions on this ansatz a set of polynomial equations is obtained in terms of some free parameters that define the product formula. Solving these polynomial equations in terms of the free parameters requires using numerical methods. Yoshida gives what appears to be all 6th order solutions but only some 8th order integrators, and did not consider any orders beyond that. It has not been proven that those are all possible 6th order solutions, but numerical evidence suggests there are no more. That work from 1990 was limited by computing power, because it requires numerical solution of a system of nonlinear equations.

In this work we have extended Yoshida's method to derive a set of polynomial equations that need to be satisfied for a 10th order product formula. We also devised an alternative method using Taylor expansions of the exponentials. We solved these equations to find 10th order product formulae, and found over 600 examples. In addition, we have found nearly another 600 examples of 8th order formulae. We have an example of an 8th order product formula with greatly reduced error, more than 100 times better than any of Yoshida's 8th order product formulae. This formula also provides greatly improved performance beyond that for any of the product formulae found by Suzuki's method when accounting for the number of exponentials. As well as finding solutions with a minimal number of exponentials, we have found solutions with extra exponentials at both 8th and 10th order that provide a further order of magnitude reduction in the error (so a factor of 1000 improvement over Yoshida for 8th order).

When comparing product formulae of different orders, it is better to use higher-order formulae for larger values of T/ϵ, where T is the total evolution time and ϵ is the required precision. We find that the threshold where our best 8th order formula outperforms lower-order formulae is quite modest at about 1600, which is well below the typical values for quantum chemistry applications. Moreover, even the best 6th order formula has poor performance, so as T/ϵ is increased, one should go straight from 4th to 8th order, and not use 6th order at all. The threshold of T/ϵ for using 10th order instead of 8th order is large, over 10^{11}, which is beyond expected values for quantum computing. As a result we expect that our 8th order product formula will be the best to use for quantum algorithms for applications in quantum chemistry.

The organisation of this work is as follows. In Section II we introduce the background necessary for the rest of the paper. First we define product formulae and introduce the Suzuki method for generating higher-order product formulae in Section II A, then in Section II B we give a summary of Yoshida's method. In Section III we introduce the Taylor series method to find new product formulae that is distinct from Yoshida's method. In Sections IV and V we present the results regarding new 8th and 10th order product formulae, and then in Section VI we give a discussion on how to compare the product formulae we have found at different order.

II. BACKGROUND

In this section we give a summary of the background for our work. We begin by defining product formulae and the Baker-Campbell-Haudorff formula, then we introduce the Suzuki method and Yoshida method to obtain higher-order formulae.

A. Product formulae

It is well known that, for any non-commuting operators X and Y,

$$\exp((X+Y)t) = \exp(Xt)\exp(Yt) + \mathcal{O}(t^2). \tag{1}$$

where we have absorbed the $-i$ factor used in quantum simulation into X and Y. The above equation demonstrates that the exponential of a sum of two operators is, to first order, equal to the product of the exponential of those operators. The above equation is often referred as a 'first-order product formula'. Higher-order terms can be computed via the Baker-Campbell-Haussdorff (BCH) formula.

Theorem 1 (Baker-Campbell-Haussdorff formula [8]). *Let X and Y be any operators such that $\|X\| + \|Y\| < \ln 2$. We have $\exp(X)\exp(Y) = \exp(Z)$, with*

$$Z = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} \sum_{\substack{r_1+s_1>0 \\ \vdots \\ r_n+s_n>0}} \frac{[X^{r_1}, Y^{s_1}, \cdots X^{r_n}, Y^{s_n}]}{\left(\sum_{j=1}^n r_i + s_i\right) \prod_{i=1}^n r_i! s_i!}, \tag{2}$$

where

$$[X^{r_1}, Y^{s_1}, \cdots X^{r_n}, Y^{s_n}] = [\underbrace{X, [X, \cdots [X,}_{r_1} \underbrace{[Y, [Y, \cdots [Y,}_{s_1} \cdots \underbrace{[X, [X, \cdots [X,}_{r_n} \underbrace{[Y, [Y, \cdots Y]}_{s_n}] \cdots]].$$

The standard second-order symmetric product formula is as given in the definition below.

Definition 2 (Symmetric product formula of order two). *Let X and Y be non-commuting operators and let t be a real variable. Define*

$$S_2(t) := \exp\left(\frac{1}{2}Xt\right) \exp(Yt) \exp\left(\frac{1}{2}Xt\right). \tag{3}$$

The operators X and Y used in the definition of S_2 should always be clear from context. More generally, when there is a sum of X_j, the product formula is

$$S_2(t) := \left[\prod_{j=1}^{J} \exp\left(\frac{1}{2}X_j t\right)\right] \left[\prod_{j=J}^{1} \exp\left(\frac{1}{2}X_j t\right)\right]. \tag{4}$$

When there are $J = 2$ terms, the expression in the definition is obtained. The corollary below describes the form of the error terms in the symmetric product formula, and also implies that it is second order.

Corollary 3 (Symmetric BCH formula [7]). *Let X and Y be any operators such that $\|X\| + \|Y\| < \ln 2$ and let t be a real variable. Define Z such that $S_2(t) = \exp(Z)$. Then there is a sequence α_ℓ consisting of linear combinations of ℓ-term commutators in X and Y such that*

$$Z = \sum_{\ell=1}^{\infty} \alpha_\ell t^\ell. \tag{5}$$

Moreover, $\alpha_\ell \equiv 0$ whenever ℓ is even.

Reference [7] also shows that even terms are zero for more general symmetric product formulae. The first three non-zero α_ℓ terms from above are

$$\alpha_1 = X + Y, \tag{6}$$

$$\alpha_3 = \frac{1}{12}[Y, [Y, X]] - \frac{1}{24}[X, [X, Y]], \tag{7}$$

$$\begin{aligned}\alpha_5 = {} & \frac{7}{5760}[X, X, X, X, Y] - \frac{1}{720}[Y, Y, Y, Y, X] + \frac{1}{360}[X, Y, Y, Y, X] + \frac{1}{360}[Y, X, X, X, Y] - \frac{1}{480}[X, X, Y, Y, X] \\ & + \frac{1}{120}[Y, Y, X, X, Y].\end{aligned} \tag{8}$$

Here the square brackets are used to indicate multicommutator expressions, for example

$$[Y, Y, X, X, Y] \equiv [Y, [Y, [X, [X, Y]]]]. \tag{9}$$

Expressions for each α_ℓ, and hence the proof of the symmetric BCH formula, arise from two applications of the usual BCH formula.

Definition 4 (Product formula). *Let X and Y be any non-commuting operators. Given a natural number n, a* product formula of order n *is a pair $(\vec{c}, \vec{d})$ with $\vec{c}, \vec{d} \in \mathbb{R}^\ell$ and ℓ a natural number such that for all $t \in \mathbb{R}$*

$$\exp((X + Y)t) = \prod_{j=1}^{\ell} \exp(c_j X t) \exp(d_j Y t) + \mathcal{O}\big(t^{n+1}\big). \tag{10}$$

We refer to the number of non-zero coefficients in $\left(\vec{c}, \vec{d}\right)$ as the length of the product formula.

The ordinary BCH formula is a length-2 product formula of order 1. The symmetric BCH formula is a length-3 product formula of order 2.

Problem 1 (Minimal-length product formulae)**.** *Given a natural number n, find a natural number ℓ_n such that there is an n^{th}-order product formula of length ℓ_n but no n^{th}-order product formula of length $\ell_n - 1$.*

It is easy to prove using Taylor expansions (for lower bounds) and the BCH formulae (for equality) that $\ell_1 = 2$ and $\ell_2 = 3$. As we discuss below, Suzuki's 'fractal' method demonstrates that $\ell_n = \mathcal{O}(\exp n)$. But it may be possible to do better.

a. Suzuki method for generating higher-order product formulae. Here we describe Suzuki's fractal methods from [1, 2] to obtain higher-order product formulae. Starting from the symmetrised product formula in Eq. (3), the fractal method generalises this expression to obtain product formulae at all even orders. Suzuki's first fractal method to generate a product formula of order $k = 2\kappa$ is [1]

$$S_{2\kappa}(t) = S_{2\kappa-2}(s_\kappa t) S_{2\kappa-2}((1-2s_\kappa)t) S_{2\kappa-2}(s_\kappa t), \tag{11}$$

where $s_\kappa = 1/(2 - 2^{1/(2\kappa-1)})$. This method can be used to generate even orders starting at S_2. A drawback to this method is that both s_k and $1 - 2s_\kappa$ are greater than 1, so the coefficients in the higher-order methods are large, causing greater error.

Alternatively to Eq. (11), an order 2κ product formula can be obtained via Suzuki's second fractal method [2]

$$\tilde{S}_{2\kappa}(t) = \tilde{S}_{2\kappa-2}(u_\kappa t)^2 \tilde{S}_{2\kappa-2}((1-4u_\kappa)t) \tilde{S}_{2\kappa-2}(u_\kappa t)^2, \tag{12}$$

where $u_\kappa = 1/(4 - 4^{1/(2\kappa-1)})$. This method has the advantage that both u_κ and $1 - 4u_\kappa$ are less than 1, so the coefficients of higher-order formulae are small resulting in small error. The drawback is that far more exponentials are required. Each iteration uses 5 copies of the lower-order formula, whereas the previous one uses 3 copies. One can use either of these iterative methods from any product formula of order $2\kappa - 2$ to obtain a formula of order 2κ, rather than simply iterating from S_2.

One of the virtues of the fractal method is that it allows one to generate arbitrarily high-order product formulae easily, as there are formulae for the coefficients s_κ and u_κ. A further advantage is that these methods work with arbitrary numbers of terms in the sum X_j. The number of exponentials used in the product formula is a crucial measure of its efficiency, and in quantum simulation it is proportional to the required number of gates.

b. Exponential length scaling of the Suzuki method. We can easily compute the number of exponentials that the Suzuki method gives assuming we start from S_2 and generate higher-order integrators. The total number of exponentials for a given order $2\kappa = 4, 6, 8, ...$ in the product formula $S_{2\kappa}$ is given by

$$2(J-1)3^{\kappa-1} + 1, \tag{13}$$

when considering exponentials of sums of J terms, so for example $J = 2$ for $X + Y$ and $\kappa = 1$ for second order gives 3. For the product formula $\tilde{S}_{2\kappa}$ the number of exponentials is

$$2(J-1)5^{\kappa-1} + 1. \tag{14}$$

The number of exponential operators in both cases of the Suzuki method grows very rapidly. Thus one may be interested in alternative method to obtain product formulae with a lower count, such as the method of Yoshida in the next section.

B. Yoshida's method for deriving 6th order product formulae

a. Approach. Rather than using Eqs. (11) and (12), Yoshida uses the general procedure

$$S^{(m)}(t) = \left(\prod_{j=1}^{m} S_2(w_{m-j+1} t) \right) S_2(w_0 t) \left(\prod_{j=1}^{m} S_2(w_j t) \right), \tag{15}$$

where $w_j \in \mathbb{R}$ for $j = 0, 1, \cdots, m$ are parameters to be determined. Note the number of exponentials in this product is given by $(4m+2)(J-1)+1$. Given this ansatz, the task becomes to find m and w_i such that $S^{(m)}$ is an order k product formula. We will illustrate Yoshida's method by deriving the result for 6th order.

b. Expand Yoshida product using Baker-Campbell-Haussdorf formula. The method is to expand Eq. (15) using the BCH formula from Theorem 1 recursively as follows. First, note that from Corollary 3, $S_2(x) = e^{\frac{t}{2}X}e^{tY}e^{\frac{t}{2}X} = e^{t\alpha_1 + t^3\alpha_3\cdots}$. We are for the moment considering sums of two terms $X+Y$. Define $C = tw_1\alpha_1 + t^3w_1^3\alpha_3 + t^5w_1^5\alpha_5 + \mathcal{O}(t^7)$ and $D = tw_0\alpha_1 + t^3w_0^3\alpha_3 + tw_0^5\alpha_5 + \mathcal{O}(t^7)$. Then,

$$
\begin{aligned}
&S_2(w_1t)S_2(w_0t)S_2(w_1t)\\
&= e^Ce^De^C\\
&= \exp\left\{tw_1\alpha_1 + t^3w_1^3\alpha_3 + t^5w_1^5\alpha_5 + \mathcal{O}(t^7)\right\}\exp\left\{tw_0\alpha_1 + t^3w_0^3\alpha_3 + tw_0^5\alpha_5 + \mathcal{O}(t^7)\right\}\\
&\quad\times\exp\left\{tw_1\alpha_1 + t^3w_1^3\alpha_3 + t^5w_1^5\alpha_5 + \mathcal{O}(t^7)\right\}\\
&= \exp\left\{\tau(2w_1+w_0)\alpha_1 + \tau^3(2w_1^3+w_0^3)\alpha_3 + \tau^5(2w_1^5+w_0^5)\alpha_5 + \frac{1}{6}([D,D,C]-[C,C,D]) + \mathcal{O}(\tau^7)\right\}. \qquad (16)
\end{aligned}
$$

A simple computation shows

$$
[D,D,C]-[C,C,D] = \tau^5(w_0^2w_1^3 - w_1^2w_0^3 + w_1^4w_0 - w_0^4w_1)[\alpha_1,\alpha_1,\alpha_3]. \qquad (17)
$$

Define $\beta_5 = [\alpha_1,\alpha_1,\alpha_3]$ so

$$
\begin{aligned}
S_2(w_1\tau)S_2(w_0\tau)S_2(w_1\tau) = \exp\Big\{&\tau(2w_1+w_0)\alpha_1 + \tau^3(2w_1^3+w_0^3)\alpha_3 + \tau^5(2w_1^5+w_0^5)\alpha_5\\
&+\tau^5\frac{1}{6}(w_0^2w_1^3 - w_1^2w_0^3 + w_0w_1^4 - w_0^4w_1)\beta_5 + O(\tau^7)\Big\}. \qquad (18)
\end{aligned}
$$

By an induction argument one shows that

$$
S^{(m)} = \exp\left\{\tau A_{1,m}\alpha_1 + \tau^3A_{3,m}\alpha_3 + \tau^5(A_{5,m}\alpha_5 + B_{5,m}\beta_5) + O(\tau^7)\right\}, \qquad (19)
$$

where $A_{j,m}$ and $B_{5,m}$ are polynomials on the variables $w_0,\dots,w_m$. The recursive formulae for the polynomials are obtained by using the following equation

$$
\begin{aligned}
S^{(m+1)}(\tau) &= S_2(w_{m+1}\tau)S^{(m)}(\tau)S_2(w_{m+1}\tau)\\
&= \exp\left\{\tau w_{m+1}\alpha_1 + \tau^3w_{m+1}^3\alpha_3 + O(\tau^5)\right\}\exp\left\{\tau A_{1,m}\alpha_1 + \tau^3A_{3,m}\alpha_3 + \tau^5(A_{5,m}\alpha_3 + B_{5,m}\beta_5)\right\}\\
&\quad\times\exp\left\{\tau w_{m+1}\alpha_1 + \tau^3w_{m+1}^3\alpha_3 + O(\tau^5)\right\}\\
&= \exp\Big\{2\tau w_{m+1}\alpha_1 + \tau A_{1,m}\alpha_1 + 2\tau^3w_{m+1}^3\alpha_3 + \tau^3A_{3,m}\alpha_3 + 2\tau^5w_{m+1}^5\alpha_5 + \tau^5A_{5,m}\alpha_5 + \tau^5B_{5,m}\beta_5\\
&\quad+\frac{1}{6}\tau^5(A_{1,m}^2w_{m+1}^3 - A_{1,m}A_{3,m}w_{m+1} - w_{m+1}^2A_{3,m} + w_{m+1}^4A_{1,m})\beta_5\Big\}. \qquad (20)
\end{aligned}
$$

From this last equality one obtains

$$
\begin{aligned}
A_{1,m+1} &= 2w_{m+1} + A_{1,m},\\
A_{3,m+1} &= 2w_{m+1}^3 + A_{3,m},\\
A_{5,m+1} &= 2w_{m+1}^5 + A_{5,m},\\
B_{5,m+1} &= B_{5,m} + \frac{1}{6}(A_{1,m}^2w_{m+1}^3 - A_{1,m}A_{3,m}w_{m+1} - w_{m+1}^2A_{3,m} + w_{m+1}^4A_{1,m}), \qquad (21)
\end{aligned}
$$

with initial conditions

$$
\begin{aligned}
A_{1,0} &= w_0,\\
A_{3,0} &= w_0^3,\\
A_{5,0} &= w_0^5,\\
B_{5,0} &= 0. \qquad (22)
\end{aligned}
$$

c. Constraints to satisfy in order to derive 6th order formula. To derive a 6th order formula, the lower-order terms in the exponential in Eq. (19) must be zero (see also Eq. (5.16) of [7]), which gives the four conditions

$$A_{1,m} = 1, \quad A_{3,m} = 0, \quad A_{5,m} = 0, \quad B_{5,m} = 0. \tag{23}$$

For $m = 3$ there are four unknowns w_0 to w_3, and it can be expected there are solutions because there are the same number of equations as unknowns. In practice $A_{1,m} = 1$ is satisfied by taking $w_0 = 1 - 2\sum_j w_j$, so there are then three equations for three unknowns w_1, w_2, w_3. Whereas it is possible to do this using Newton-Raphson, the expression for the appropriate Jacobian matrix is complicated and so Yoshida uses the Brent method instead. Yoshida produced three $m = 3$ solutions in this way, and states "It seems that there is no other solution." We have performed an extensive search and also find no more solutions.

The product formulae obtained through the Yoshida method also work for sums of more operators. The S_2 product formula can again be used as the building block for the product formula, and we can write

$$S_2(t) = \exp\left(\sum_{\ell=0}^{\infty} \tilde{\alpha}_\ell t^\ell\right), \tag{24}$$

where $\tilde{\alpha}_\ell$ are now order-ℓ multicommutator expressions on the J terms. The reasoning for finding the product formula is entirely based on the construction with α_ℓ, but without taking advantage of its particular form, so exactly the same reasoning holds for $\tilde{\alpha}_\ell$. This immediately implies that the higher-order product formulae work in general. This is an advantage of deriving product formulae as products of S_2, because deriving coefficients separately for exponentials of X and Y would not generalise.

III. SOLUTION USING TAYLOR EXPANSION

a. Approach. Another solution method we use is based on computing the Taylor expansion of both the exact exponential and its product formula approximation with given w_j. This Taylor expansion is performed on both sides up to the desired order of approximation for the integrator. By imposing equality on terms of the same order we obtain a series of equations for w_j which can be solved. For higher orders, a large number of simultaneous equations are obtained, so we need an automated way of generating them.

b. Definitions used to specify problem. To make precise the problem we are solving, denote as $\mathcal{T}_k[\cdot]$ the map giving the Taylor expansion in t around 0, truncated at order k, so

$$\begin{aligned}\mathcal{T}_k[e^{t(X+Y)}] &= \sum_{p=0}^{k} \frac{t^p}{p!}(X+Y)^p \\ &= \sum_{p=0}^{k} \frac{t^p}{p!} \sum_{r_1,\cdots,r_p=0}^{1} X^{r_1} Y^{1-r_1} \cdots X^{r_p} Y^{1-r_p}.\end{aligned} \tag{25}$$

We consider a sum of two operators $X + Y$, but this approach for solving for w_j will also be sufficient to provide product formulae for sums of arbitrary numbers of terms. That is because the solutions must also be solutions of the equations derived using Yoshida's method, and as explained above those equations will be the same when considering sums of arbitrary numbers of terms. Now consider the ansatz operator of Yoshida from Eq. (15)

$$\begin{aligned}S^{(m)}(t, w_1, \cdots, w_m) &= e^{tw_m X/2} e^{tw_m Y} e^{t(w_m + w_{m-1})X/2} e^{tw_{m-1} Y} e^{t(w_{m-1}+w_{m-2})X/2} \\ &\quad \cdots e^{tw_0 Y} e^{t(w_1+w_0)X/2} e^{tw_1 Y} \cdots e^{tw_m X/2} \\ &= e^{tc_1 X} e^{tc_2 Y} \cdots e^{tc_{4m+3} X},\end{aligned} \tag{26}$$

where in the last line we have defined the constants $c_1 = w_m/2$, $c_2 = w_m$, $c_3 = (w_m + w_{m-1})/2, \cdots$ $c_{4m+3} = w_m/2$. We Taylor expand this ansatz up order n, noting that the total number of exponentials in Yoshida's ansatz is $4m + 3$

$$\mathcal{T}_k[S^{(m)}(t, w_1, \ldots, w_m)] = \sum_{\substack{r_1,\cdots,r_{4m+3}=0 \\ r_1+\cdots+r_{4m+3}\leq k}}^{k} \frac{t^{r_1+\cdots+r_{4m+3}}}{r_1! \cdots r_{4m+3}!} c_1^{r_1} \ldots c_{4m+3}^{r_{4m+3}} X^{r_1} Y^{r_2} \ldots X^{r_{k-1}} Y^{r_{4m+3}}. \tag{27}$$

c. Data structure for coefficients. We require that the product formula obtained from our solution procedure be independent of the choice of X and Y, so need to match the coefficients for each sequence of products of X and Y. In order to do this in an automated way we construct a data structure to store the coefficients.

Given an operator of the form $e^{cX} = I + cX + \frac{c^2}{2!}X^2 + \frac{c^3}{3!}X^3 + \cdots$ and $e^{dY} = I + dY + \frac{d^2}{2!}Y^2 + \frac{d^3}{3!}Y^3 + \cdots$ with c a scalar coefficient and X, Y general operators, we can describe this Taylor expansion up to an order n using an array encoding. First, we write monomials composed of X and Y operators in lexicographical order and note that these operators can be mapped to binary numbers:

$$\begin{array}{cccccccccc} I & X & Y & XX & XY & YX & YY & XXX & XXY & \cdots \\ 1 & 10 & 11 & 100 & 101 & 110 & 111 & 1000 & 1001 & \cdots \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \cdots \end{array} \tag{28}$$

To construct a bit string, we map each X with 0 and each Y with 1, then place a 1 on the left to flag the length of the string, as shown in the second line of Eq. (28). Then, to obtain the operator products, simply remove the leading 1 and then map 0 to X and 1 to Y. The empty string corresponds to the identity I. Now, to store the coefficients in a sum of products of X and Y, convert each product to a binary integer as above, then store the coefficient in the corresponding location in a vector.

By way of illustration, consider the polynomial $10I + 3X + 2Y + 2X^2 + YX$. This operator would be stored in an array as $[10, 3, 2, 2, 0, 1, 0, \cdots]$. In this way, any polynomial of X and Y can be efficiently stored in a vector. We denote this vector storing the coefficients of operators of order up to k as $\mathrm{vec}_k[p(X,Y)]$, where k denotes that the vector will only store the coefficients of the corresponding operators up to order k (so a vector of length $2^{k+1} - 1$) and $p(X,Y)$ is the polynomial in terms of X and Y.

d. Tree product between coefficient arrays. We can define a product between these arrays that corresponds to the (noncommutative) product of polynomials in X and Y. Let p and q be such polynomials, and define the tree product $*$ as $\mathrm{vec}_k[p(X,Y)] * \mathrm{vec}_k[q(X,Y)] = \mathrm{vec}_k[p(X,Y) \cdot q(X,Y)]$, where "$\cdot$" is the usual operator product.

In order to obtain the coefficients w_j, we encode $\mathcal{T}_k[e^{t(X+Y)}]$ and $\mathcal{T}_k[S^{(m)}(t, w_1, \cdots, w_m)]$ from Eq. (25) and Eq. (27) using the encoding just described, disregarding the time t. Finally, the problem to be solved is as follows.

Problem 2 (Cost function error minimisation of product formula coefficients)**.** *Let X and Y be arbitrary non-commuting operators and let n be a natural number. Solve*

$$\mathrm{argmin}_{w_1, w_2, \cdots, w_m} \left\| \mathrm{vec}_k \left[\mathcal{T}_k[e^{t(X+Y)}] \right] - \mathrm{vec}_k \left[\mathcal{T}_k[S^{(m)}(t, w_1, \cdots, w_m)] \right] \right\|^2 . \tag{29}$$

Our numerical solution method involves a minimisation, and for the solution the minimum should be zero. We choose to use the Yoshida ansatz as it requires a lower number of parameters in comparison to the most general ansatz of length m, and also naturally generalises to sums of more terms. Nevertheless, it is also possible to use our technique for more general non-symmetric product formulae.

e. Solution strategy. We first need to choose m to be large enough such that the minimum (not argmin) is equal to zero; i.e. it is possible to find an order n product formula of length m. To choose a value of m that successfully yields an 8th order product formula, we can follow Yoshida's ansatz: we know $m = 7$ works. To choose an m that yields a 10th order product formula, we extend the work of Yoshida to determine that $m = 15$ works. See Appendix A for details.

Having chosen an appropriate value of m, we then can use a numerical nonlinear function solver. We tried various approaches and found that Matlab's `fsolve` was able to succeed if used as follows. Choose a random vector $\vec{w}$ and evaluate `fsolve` with the vector of errors. We find that generating the components of $\vec{w}$ according to the standard normal distribution works. We found that the best solutions were those with smaller values for the coefficients, so reduced the standard deviation for the initial $\vec{w}$ a little below 1. We tried standard deviations of 0.6 initially for 8th order, 0.9 for 10th order, and later tried a standard deviation of 1 for 8th order.

Matlab reports that `fsolve` uses the Levenberg-Marquardt algorithm, which interpolates between the Gauss–Newton algorithm and gradient descent. It uses information in the full vector of errors, so provides better performance than `fminsearch` with the sum of squares of errors. In comparison, Yoshida used the Brent method to solve the polynomial equations he derived [7]. In more recent work, other authors have also used this approach of minimising the errors [9, 10]. Although the coefficients obtained are not guaranteed to be formal solutions of the equations, the error minimisation is performed to very high accuracy.

In most cases the sum of the squares of errors was on the order of 10^{-27}, though some were larger, on the order of 10^{-22}. Solving further using extended precision (instead of double precision) resulted in the sum of squared errors being reduced by many more orders of magnitude. That is a strong indication that these are exact solutions. These solutions obtained through error minimisation work as practical implementation of product formulae given the small error they achieve.

In real analysis, the Poincaré-Miranda theorem [11] provides necessary and sufficient conditions to check whether there is a root of a set of non-linear equations in a hypercube. This theorem is a generalisation of the intermediate value theorem to multiple functions and variables. Unfortunately, this theorem requires the evaluation of the function at infinite points and does not provide a way to check that we have indeed reached a root of the system of equations. Nonetheless, evaluating the polynomials found through Yoshida's method on the vertices of the hypercube near the solutions found can help to discard those that are not near roots. We evaluated the polynomials and check that the conditions of the Poincaré-Miranda theorem are fulfilled for this vertices. We performed this evaluation on many of the solutions we found, and they passed the test for points at a distance of 10^{-10} from our solutions.

IV. IMPROVED 8TH ORDER PRODUCT FORMULAE

We have solved for product formulae both using our Taylor series procedure and the polynomial equations of Yoshida, and found nearly 600 product formulae of 8th order. In what follows we number our solutions according to the order in which we found them. To clarify the terminology, there is a distinction between the "cost function error" and the "product formula error". The cost function error refers to the minimised error in Problem 2, on the other hand the product formula error refers to the error of the product formula when compared to the total evolution, for example in Eq. (1) the product formula error is given by the expression $\mathcal{O}(t^2)$. Not only do we find many more solutions than Yoshida reports, but we also find product formulae with reduced product formula error. To show this we have compared the different product formula errors attained by the integrators under different Hamiltonians chosen randomly. The detail of how we compare product formulae is given in Section VI.

To be more precise, we have performed the search of product formulae by solving the optimisation in Problem 2 with $n = 8$ and $m = 7$. We have also numerically solved the polynomial equations of Yoshida for the search. Whenever we found certain parameters $w_1, \cdots, w_7$ giving a sum of squared errors below 10^{-20}, we considered these parameters as a potential product formula to be tested. The search now finds almost only repeated solutions and very few new solutions. This indicates that we have found nearly all solutions, but it is also possible that there are many more solutions with large values of $\vec{w}$. Indeed, the most recent new solutions we found have significantly larger values of $\vec{w}$. We find that large values of $\vec{w}$ correspond to worse product formulae with larger error. Therefore, even if there are many more solutions with large $\vec{w}$, they will not give improved performance over those we have already found.

Once we have obtained the potential solutions, we generate random Hamiltonians and compute the product formula errors as a function of time. We show these errors in Fig. 1 for an example Hamiltonian of dimension $d = 4$, and 5 examples of product formulae. For 8th order product formulae we know that the product formula error is $\mathcal{O}(t^9)$. We check the error scaling by picking two times t_1 and t_2 and computing errors δ_1 and δ_2 at these times, then we compute $\zeta = \log(\delta_1/\delta_2)/\log(t_1/t_2)$ and check that it is close to 9. As the error for our product formulae is given by $\delta(t) = \chi t^9$ where χ is a constant factor, we can compute χ for each of them by considering $\chi = \delta/t^9$. For each product formula, we compute an average constant factor error, this average corresponds to the geometric mean of the constant factors computed for each random Hamiltonian. This method of comparing the performance of product formulae through the estimation of the constant factor in the error has been used before (see for example [9]) and is considered a good approximation to the performance of the product formula in practice.

The two best performing 8th order product formulae as measured by the constant factor χ obtained are shown in Table I. These correspond to solution 42 with average constant factor $\chi = 5.8 \times 10^{-6}$ and solution 100 with average constant factor $\chi = 9.4 \times 10^{-6}$. These solutions are reported in extended precision; by using extended precision arithmetic we reduced the cost function error of solution 42 to 10^{-600}. For comparison, the worst performing product formula we found had a constant of $\chi = 2.7 \times 10^{5}$. We evaluated all the 8th order solutions of Yoshida, and found Solution D was best, with a constant $\chi = 9.7 \times 10^{-4}$.

We have also conducted a search for 8th order solutions with $m = 8$. Using $m = 8$ results in an underdetermined system of equations with continuous sets of solutions, and gives the flexibility to adjust the solution to reduce the error. The best solution found is given in Table II, with an average constant factor of $\chi = 5.7 \times 10^{-7}$, which is an order of magnitude improvement with only a slight increase in the number of exponentials.

There is a second way to compare the product formulae. Note that $\vec{v} = \mathrm{vec}_k[\mathcal{T}_k[e^{t(A+B)}] - \mathcal{T}_k[S^{(m)}(t, w_1, \cdots, w_m)]]$ will be a vector with entries very close to zero for a product formula solution. We could also pick a larger k than the solution was derived for, for example $k = 9$ for an 8th order solution (so the k here is no longer the order). Then we consider the entries of the vector $\vec{v}$ that includes the 9th order operators. We have computed 9th, 10th, 11th and 12th order cost function errors. For each order we take the absolute value of this errors and sum them. We find that the lowest sum of absolute errors at each order is achieved by the best solution determined by the previous method (solution 42), and in fact the lowest sum of errors is strongly correlated with the average constant factor χ mentioned above.

In Fig. 2 we plot the constant factor in the error versus the sum of absolute errors at 9th order. It can be seen that

	Solution 42	Solution 100
w_1	0.315293092396766596632056663811	0.37122062648117505118097053722986
w_2	0.33462491824529818378495797988218	0.40544709650967949690890447887218
w_3	0.2990641813036559238444635406886	0.16633724441837318387261356221838
w_4	−0.57386247111608226665638772663554	−0.62219910114766848553693391042818
w_5	0.19075471029623837995387625645037	0.26406879487125261601060713402535
w_6	−0.40910082580003159399730009589356	−0.45453364433377659463237935329715
w_7	0.74167036435061295344822780178381	0.79748609972350707868528219873049

TABLE I. Our two best-performing 8th order solutions with $m = 7$.

	Best 8th order with $m = 8$
w_1	0.29137384767986663096528500968049
w_2	0.26020394234904150277316667709864
w_3	0.18669648149540687549831902999911
w_4	−0.40049110428180105319963667975074
w_5	0.15982762208609923217390166127256
w_6	−0.38400573301491401473462588779099
w_7	0.56148845266356446893590729572808
w_8	0.12783360986284110837857554950443

TABLE II. Our best-performing 8th order solution when setting $m = 8$.

there is strong correlation, though the sum of absolute errors overestimates the error by a factor of 100 to 1000. The correlation coefficient between the sum of errors at order 9 and the constant factor is 0.977, at order 10 the correlation coefficient is 0.973, at order 11 it is 0.948, and at 12 it is 0.938. In Fig. 2 we compute the sum of absolute errors at 9th order and compare to the constant factor in the error.

We also have computed a bound on the error for 8th order integrators based on the work of [4]. In Appendix M of that work, the authors derive a bound on the error for the Suzuki product formula of 4th order. We have extended the bounds to 8th order by generalisation their method to this case by implementing it in Mathematica. We have computed this bound for two of our best 8th order product formulae with $m = 7$ and show the results in Fig. 3. The bounds are very loose, being about a factor of 10^6 times larger than the actual error, though they follow the same trend.

It can be shown that the total evolution time for a particular product formula can be used to bound the error in approximating the total evolution. Given a Hamiltonian $H = \sum_{i=1}^{\ell} H_i$ where the H_i do not necessarily commute, the total time of evolution for a product formula $e^{tc_1 X_1} e^{tc_2 X_2} \cdots e^{tc_\ell}$ is given by $\sum_{i=1}^{\ell} |tc_i|$. When considering Yoshida's ansatz from Eq. (15) for parameters $(w_0, w_1, \cdots, w_m)$, the total evolution time is given by $t\left(|w_0| + 2\sum_{i=1}^{m} |w_i|\right)$. We give a derivation of the bound given by the total evolution time in Appendix B. We also compare the estimated constant factor of the product formula error with the expression $|w_0| + 2\sum_{i=1}^{m} |w_i|$ for the best 8th order product formulae we have found. We find a strong correlation between these two quantities, characterised by a correlation coefficient of 0.78. A plot showing this correlation is given in Fig. 4.

V. FINDING 10TH ORDER PRODUCT FORMULAE

We have also used our solution procedure to find new 10th order product formulae. We have generalised Yoshida's method, and find 15 independent equations to be solved. This derivation is quite lengthy, so is given in Appendix A. We performed searches for solutions both with $m = 15$ (the minimal number) and $m = 16$. Again this gives the flexibility to adjust the solution to reduce the error. We report the best 10th order product formulae for $m = 15$ and 16 in Table III in extended precision.

As in Section IV, we compare the performance of product formulae of 10th order by computing the constant factor χ in the error χt^{11} for random Hamiltonians. For the best solution with $m = 16$ we have a constant factor of $\chi = 1.9 \times 10^{-8}$, and the best solution with $m = 15$ has $\chi = 9.4 \times 10^{-7}$, which is about a factor of 50 times worse.

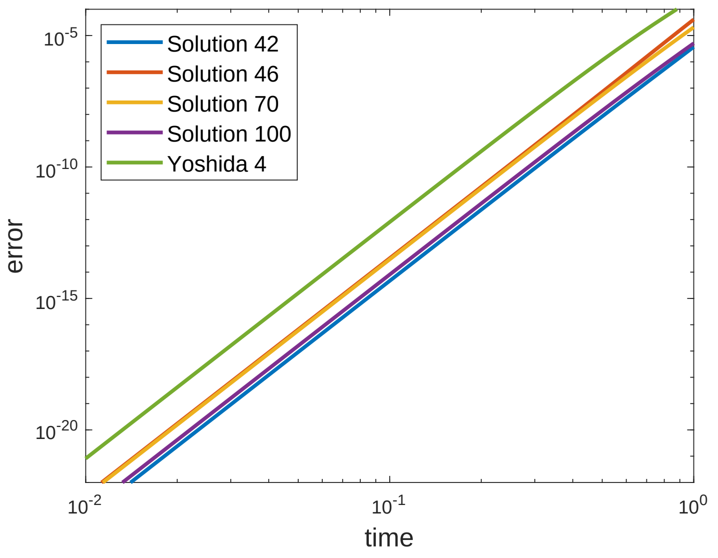

FIG. 1. Error in product formula as determined by the spectral norm of the difference of operators as a function of t. We have shown our four best-performing product formulae for 8th order; these correspond to solutions $42, 46, 70, 100$. For comparison we also show the best-performing solution of Yoshida, with errors an order of magnitude higher than the solutions we have obtained.

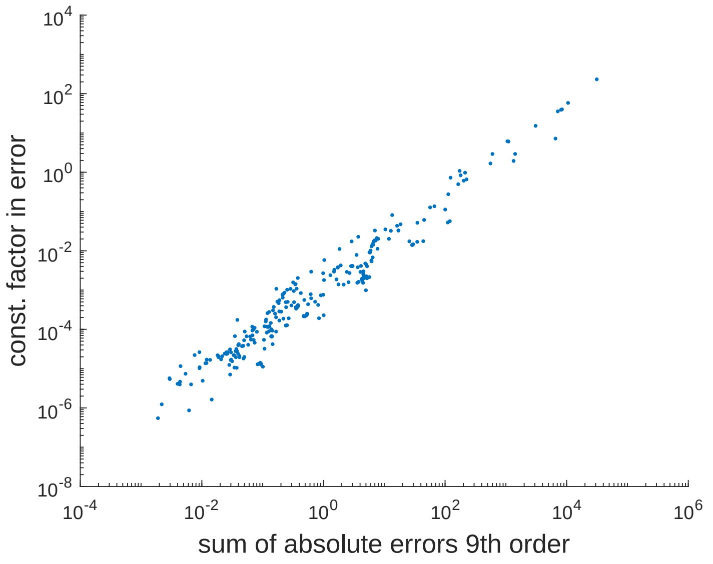

FIG. 2. Plot of average constant factor in the error χt^9 for 100 random Hamiltonians and the sum of absolute 9th order errors defined in the main text. Each of the points represents one product formula obtained with our optimisation procedure.

The far better constant factor for $m = 16$ is far more significant than the slightly larger number of exponentials in the product formula. For this reason we consider cases with $m = 16$ in the remainder of this discussion.

We compare our best-performing 10th order product formula to our best 8th order formula, and Yoshida's best, in Fig. 5. We also compare in Fig. 6 the best product formulae when the total Hamiltonian is given by a sum of ten terms. As explained in Section II B, the product formulae are also valid for Hamiltonians that are sums of arbitrary numbers of terms. This plot demonstrates that the correct scalings are still obtained with larger numbers of terms.

As in the 8th order case, we compute the total evolution time and compare with the constant error factor χ for a set

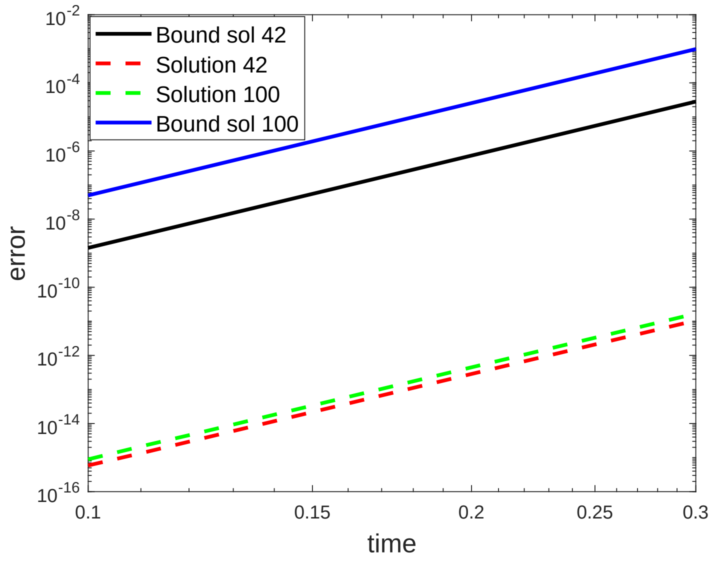

FIG. 3. Error for two of our best solutions for an example Hamiltonian, compared with the error bound computed using a method based on that in [4].

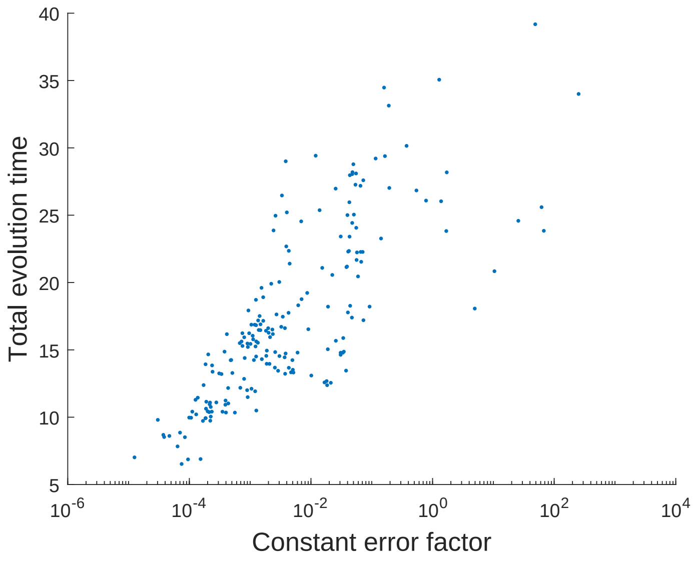

FIG. 4. Comparison of the total evolution time $|w_0| + 2\sum_{i=1}^{m} |w_i|$ and the constant factor in the product formula error for the best 8th order product formulae we have found.

of 10th product formulae found by the optimization procedure. The result is shown in Fig. 7. The correlation factor is smaller than in the 8th order case, but still shows a relationship between these two quantities with a correlation factor of 0.5. We also find a correlation between the RMS of the 10th order product formulae and the average constant factor in the error obtained by generating random Hamiltonians. We show the constant factor versus RMS values in Fig. 8. The correlation coefficient is 0.715 comparing the RMS of w to the log of χ. In the search for 10th order product formulae, unlike in the case of 8th order, we find that almost all new solutions found are different those found before. That indicates that there is an extremely large number of solutions, and we have only found a very small proportion of them. Potentially there are solutions with even better performance still to be found.

	Best 10th order solution with $m = 15$	Best 10th order solution with $m = 16$
w_1	0.14552859955499429739088135596618	−0.49450131799555718563471479776 44
w_2	−0.48773512068133537309419933740564	0.29043172229701214798784142920 93
w_3	0.12762011242429535909727342301656	0.34781541068705330937913890281003
w_4	0.70225450019485751220143080587959	−0.98828132118546184603769781410676
w_5	−0.62035679146761710925756521405042	0.98855187532756405235733957305613
w_6	0.39099152412786178133688869373114	−0.34622976933123177430694714630668
w_7	0.17860253604355465807791041367045	0.20218952619073117554714280367018
w_8	−0.80455783177921776295588528272593	0.13064273069786247787208895471461
w_9	0.05308721644275824211868738564 6283	−0.26441199183146805554735845490359
w_{10}	0.86836307910275556258687030904753	0.06099914055921040886909699229 1531
w_{11}	−0.85326297197907834671536254437991	−0.68554424896061413591089732670 28
w_{12}	−0.11732457198874083224967699358383	−0.15843692473786584550599206557006
w_{13}	0.03827345494186056632406947772047	0.15414691779958299150286452215575
w_{14}	0.74843529029532498233997793305357	0.66715205827214320371061839297055
w_{15}	0.30208715621975773712410948025906	0.20411874474696598289603677693511
w_{16}	NA	0.08120731821027259322508771144 1684

TABLE III. Our best performing solutions for 10th order with $m = 15$ and $m = 16$.

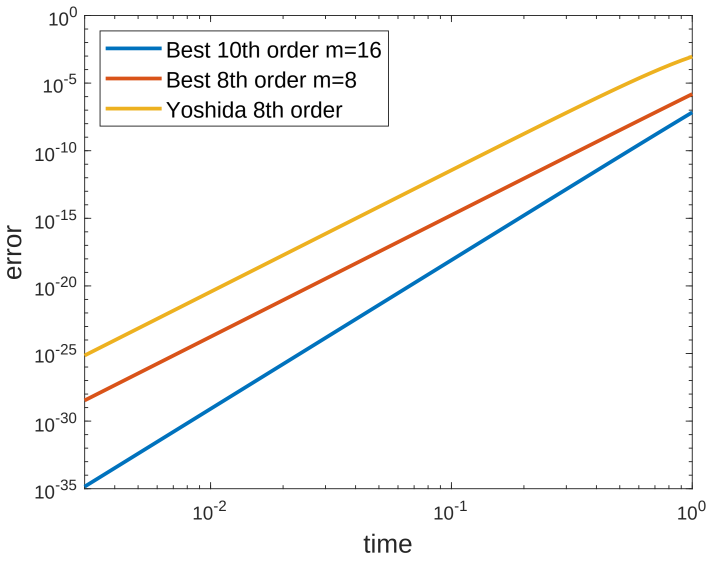

FIG. 5. Error of the best 8th and 10th product formula obtained by our optimisation procedure together with the best 8th order product formula from Yoshida. To compute the error, two pairs of random Hamiltonians A and B were generated and the error was evaluated comparing to the total evolution $e^{-it(A+B)}$.

VI. COMPARISON OF PRODUCT FORMULAE

To compare the product formulae we've obtained, we make the following considerations. A k order integrator for time t will have an error $\delta = \chi t^{k+1}$ where χ is a real constant. Let T be the total evolution time for an integrator of order k, and ε be the maximum allowable error. Subdivide the evolution time T into r subintervals, so $t = T/r$ is the length of each time subinterval. We thus have $\chi\left(T/r\right)^{k+1} \approx \epsilon/r$, which gives

$$r \approx \left(\frac{\chi T}{\epsilon}\right)^{1/k} T. \tag{30}$$

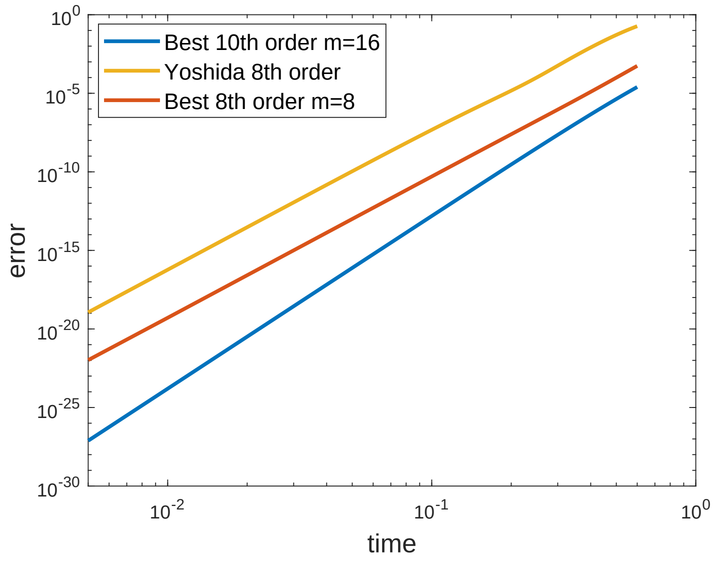

FIG. 6. Error in the case of the total Hamiltonian decomposed into 10 terms. We compare the best 8th and 10th product formula obtained by our optimisation procedure together with the best 8th order product formula from Yoshida. We generate a random tuple of Hamiltonian terms $(H_1, \cdots, H_{10})$ and compute the error comparing the product formula with the total evolution $e^{-it\sum_i H_i}$.

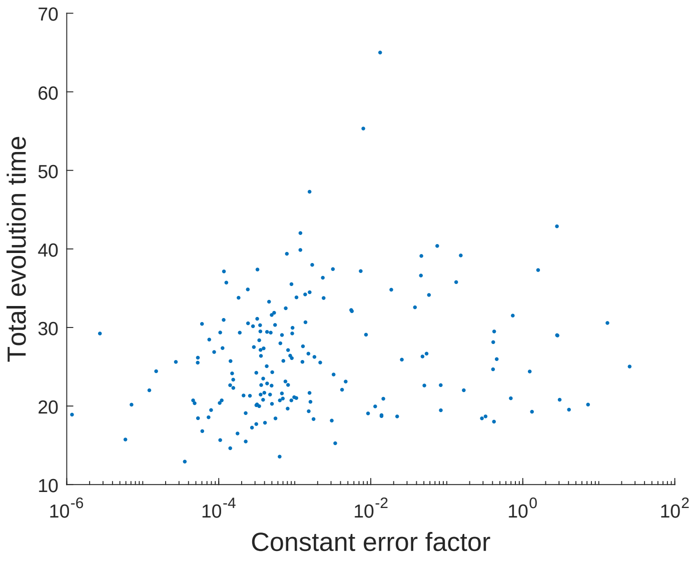

FIG. 7. Comparison of the total evolution time $|w_0| + 2\sum_{i=1}^{m} |w_i|$ and the constant factor in the product formula error for the best 10th order product formulae we have found.

As explained above, the number of exponentials in the product is $(4m+2)(J-1)+1$. When applying products of these product formulae, two exponentials can be combined, so the effective number for each is $(4m+2)(J-1)$. As a result, the total number of exponentials can be given as proportional to

$$(2m+1)\left(\frac{\chi T}{\epsilon}\right)^{1/k} T \tag{31}$$

where we have ignored a common factor of $2(J-1)$.

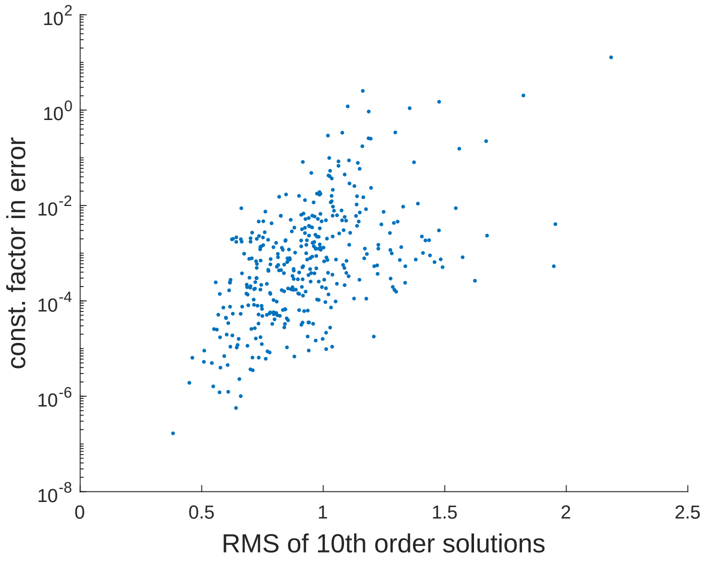

FIG. 8. For each 10th product formula we compute the average constant factor in the error χt^{11} for 20 random Hamiltonians. Moreover, for each product formula w, we compute the RMS. Each point in the plot represents a single product formula.

If we wish to compare product formulae of the same order, then we need only compare the values of $(2m+1)\chi^{1/k}$, and the one with the smaller value is the more efficient product formula. This clearly demonstrates that our best product formula with $m = 16$ is better than our best with $m = 15$. We also evaluated product formulae derived using Suzuki's fractal method via a number of approaches, such as constructing 10th order from the best 8th order, or simply iterating the fractal method. Although some of these gave better χ values, these were more than outweighed by the significantly larger values of m, meaning that they were not as efficient as our best product formulae (both for 8th and 10th order).

If we wish to compare product formulae of *different* order, then we need to take account of the values of T and ϵ. Assume we have two integrators of order k_1 and k_2, with corresponding constants χ_1, χ_2. Given T and ϵ, when the two integrators use the same number of exponentials we have $(2m_1+1)r_1 = (2m_2+1)r_2$, thus

$$(2m_1+1)\left(\frac{\chi_1 T}{\varepsilon}\right)^{1/k_1} T = (2m_2+1)\left(\frac{\chi_2 T}{\varepsilon}\right)^{1/k_2} T \tag{32}$$

$$\implies \frac{T}{\varepsilon} = \left(\frac{(2m_2+1)\chi_2^{1/k_2}}{(2m_1+1)\chi_1^{1/k_1}}\right)^{\frac{1}{\frac{1}{k_1}-\frac{1}{k_2}}}. \tag{33}$$

For $k_2 > k_1$, this gives the threshold beyond which we should use the higher-order product formula.

To compare our best 8th order and 10th order product formulae, we took $m_1 = 8, m_2 = 16, k_1 = 8, k_2 = 10$, and estimated χ_1 and χ_2 by averaging over $10,000$ random Hamiltonians. We found $\chi_1 = 5.7 \times 10^{-7}$ and $\chi_2 = 1.8 \times 10^{-8}$, which gives the estimated threshold for 10th order to provide an advantage as about $T/\epsilon \sim 6.4 \times 10^{11}$. Our best minimal 8th-order product formula with $m_1 = 7$ has a constant factor about ten times larger, at $\chi_1 = 5.8 \times 10^{-6}$. That would give a significantly lower threshold to use 10th order of $T/\epsilon \sim 8.4 \times 10^8$.

The corresponding value of χ for the best 6th order formula of Yoshida (Solution A) is 1.6×10^{-3}. As a result, the threshold T/ϵ for 8th order to improve over 6th order is only about 45. The value of χ for the 4th order product formula (using Suzuki's method with 5 copies of S_2) is about 2.5×10^{-3}. As a result, the threshold for 6th order to improve over 4th order is about 9.4×10^3, which is well above that for 8th order to beat 6th order. This means that, as T/ϵ is increased, one should change over directly from 4th order to 8th order, and not use 6th order. The threshold for using 8th order instead of 4th order is about 1.6×10^3.

To estimate the order of magnitude of T/ϵ for quantum chemistry, we can consider the values of λ from [12] of about 35,000 Hartree. The chemical accuracy required for the phase estimation is about 0.001 Hartree. The value of λ can be regarded as the approximate value to use to normalise the Hamiltonian, because we have computed χ for normalised Hamiltonians. The equivalent value of T to use for phase estimation would therefore be approximately

$\pi \times 35 \times 10^6 \sim 10^8$. When performing the phase estimation the error in the Hamiltonian evolution for the longest time can be allowed to be $\mathcal{O}(1)$, and so appropriate values of T/ϵ for quantum chemistry can be estimated to be about 10^8. Therefore the threshold to use 10th order instead of 8th order is well beyond the expected values needed for simulations, and it is still somewhat larger if we were to compare the 10th order solution with $m = 16$ with the 8th order with $m = 7$.

VII. CONCLUSION

We have extended the method of Yoshida to construct product formulae of 10th order with a minimum number of terms (for symmetric product formulae constructed from S_2). We have also constructed 10th order product formulae with more factors that are far more accurate. Yoshida only found five 8th order solutions, but we have found nearly 600, including one that improves over Yoshida's best by more than two orders of magnitude. We have also found hundreds of examples with more factors that further reduce the error.

We have provided a method of fairly comparing product formulae with different numbers of terms and different orders. This demonstrates that our best solutions for 8th and 10th order also improve over those obtained using Suzuki's fractal method with lower error but many more terms. For comparing our best 10th order to our best 8th order, simulations with $T/\epsilon \gtrsim 6 \times 10^{11}$ would be required. This is far larger than that expected to be needed for quantum chemistry simulations, indicating that our best 8th order solution would be best for most simulations.

An area for further work is to evaluate the product formulae specifically for the Hamiltonians needed for quantum chemistry, to determine if the error is distinct from that for the random Hamiltonians tested here. It is also possible that further increasing m could give even better performing product formulae, though there are already very good improvements just increasing m by 1. There is also the possibility to extend the solutions to even higher-order product formulae, though those would likely need much higher numbers of terms, and so the threshold for them to provide an advantage would be even higher.

VIII. ACKNOWLEDGEMENTS

MESM was supported by the ARC Centre of Excellence for Quantum Computation and Communication Technology (CQC2T), project number CE170100012 and a scholarship top-up and extension from the Sydney Quantum Academy. DB acknowledges funding by the Australian Research Council (project numbers FT190100106, DP210101367, CE170100009). DWB worked on this project under a sponsored research agreement with Google Quantum AI. DWB is also supported by Australian Research Council Discovery Project DP210101367.

[1] M. Suzuki, Physics Letters A **146**, 319 (1990).
[2] M. Suzuki, Journal of Mathematical Physics **32**, 400 (1991).
[3] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders, Communications in Mathematical Physics **270**, 359 (2007).
[4] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, Physical Review X **11**, 011020 (2021).
[5] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, Physical Review Letters **114**, 90502 (2015).
[6] G. H. Low and I. L. Chuang, Physical Review Letters **118**, 010501 (2017).
[7] H. Yoshida, Physics Letters A **150**, 262 (1990).
[8] S. Blanes and F. Casas, Linear Algebra and its Applications **378**, 135 (2004).
[9] S. Blanes, F. Casas, P. Chartier, and A. Murua, Mathematics of Computation **82**, 1559 (2013).
[10] W. Auzinger, H. Hofstätter, D. Ketcheson, and O. Koch, BIT Numerical Mathematics **57**, 55 (2017).
[11] W. Kulpa, The American Mathematical Monthly **104**, 545 (1997).
[12] D. W. Berry, C. Gidney, M. Motta, J. R. McClean, and R. Babbush, Quantum **3**, 208 (2019).

Appendix A: Extending Yoshida's method to 10th order

Here we explain how to extend the method of Yoshida to obtain the equations for a 10th order integrator. We begin by computing the ansatz in the case $m = 1$. Define $C = \tau w_1 \alpha_1 + \tau^3 w_1^3 \alpha_3 + \tau^5 w_1^5 \alpha_5 + \tau^7 w_1^7 \alpha_7 + \tau^9 w_1^9 \alpha_9 + O(\tau^{11})$ and $D = \tau w_0 \alpha_1 + \tau^3 w_0^3 \alpha_3 + \tau w_0^5 \alpha_5 + \tau^7 w_0^7 \alpha_7 + \tau^9 w_0^9 \alpha_9 + O(\tau^{11})$. Just as in (16), we obtain the following by applying

Corollary 3

$$S_2(w_1\tau)S_2(w_0\tau)S_2(w_1\tau) = \exp\left\{\tau w_1\alpha_1 + \tau^3 w_1^3\alpha_3 + \tau^5 w_1^5\alpha_5 + \tau^7 w_1^7\alpha_7 + \tau^9 w_1^9\alpha_9 + O(\tau^{11})\right\}$$
$$\times \exp\left\{\tau w_0\alpha_1 + \tau^3 w_0^3\alpha_3 + \tau w_0^5\alpha_5 + \tau^7 w_0^7\alpha_7 + \tau^9 w_0^9\alpha_9 + O(\tau^{11})\right\}$$
$$\times \exp\left\{\tau w_1\alpha_1 + \tau^3 w_1^3\alpha_3 + \tau^5 w_1^5\alpha_5 + \tau^7 w_1^7\alpha_7 + \tau^9 w_1^9\alpha_9 + O(\tau^{11})\right\}. \tag{A1}$$

We use now the symmetric BCH expansion for the following equation

$$e^C e^D e^C = e^Z, \tag{A2}$$

where

$$\begin{aligned} Z = {} & 2C + D + \frac{1}{6}([D,D,C] - [C,C,D]) \\ & + \frac{7}{360}[C,C,C,C,D] - \frac{1}{360}[D,D,D,D,C] \\ & + \frac{1}{90}[C,D,D,D,C] + \frac{1}{45}[D,C,C,C,D] \\ & - \frac{1}{60}[C,C,D,D,C] + \frac{1}{30}[D,D,C,C,D] \\ & - \frac{31}{15120}[C,C,C,C,C,C,D] - \frac{31}{5040}[D,C,C,C,C,C,D] \\ & - \frac{13}{1890}[D,D,C,C,C,C,D] - \frac{53}{15120}[D,D,D,C,C,C,D] \\ & - \frac{1}{1260}[D,D,D,D,C,C,D] - \frac{1}{15120}[D,D,D,D,D,C,D] + \mathcal{R}_{(9\leq)} \end{aligned} \tag{A3}$$

Where $\mathcal{R}_{(9\leq)}$ is an infinite sum with commutators of an odd number of operators (equal to or higher than 9). The commutators that appear depend on the BCH formula from Corollary 3.

Before computing the commutators it will be useful to define the following operators:

$$\beta_9 = [\alpha_1, \alpha_1, \alpha_7], \tag{A4}$$
$$\gamma_9^{(1)} = [\alpha_1, \alpha_3, \alpha_5], \tag{A5}$$
$$\gamma_9^{(2)} = [\alpha_3, \alpha_1, \alpha_5], \tag{A6}$$
$$\gamma_9^{(3)} = [\alpha_5, \alpha_1, \alpha_3], \tag{A7}$$
$$\delta_9^{(1)} = [\alpha_1^4, \alpha_5], \tag{A8}$$
$$\delta_9^{(2)} = [\alpha_3, \alpha_1^3, \alpha_3], \tag{A9}$$
$$\delta_9^{(3)} = [\alpha_1, \alpha_3, \alpha_1^2, \alpha_3], \tag{A10}$$
$$\epsilon_9 = [\alpha_1^6, \alpha_3]. \tag{A11}$$

By applying the BCH formula to Eq. (A1) and using induction over m, we find that

$$\begin{aligned} S^{(m)}(\tau) = \exp\Big\{ & \tau A_{1,m}\alpha_1 + \tau^3 A_{3,m}\alpha_3 + \tau^5(A_{5,m}\alpha_5 + B_{5,m}\beta_5) \\ & + \tau^7(A_{7,m}\alpha_7 + B_{7,m}\beta_7 + C_{7,m}\gamma_7 + D_{7,m}\delta_7) \\ & + \tau^9(A_{9,m}\alpha_9 + B_{9,m}\beta_{9,m} + C_{9,m}^{(1)}\gamma_9^{(1)} + C_{9,m}^{(2)}\gamma_9^{(2)} + C_{9,m}^{(3)}\gamma_9^{(3)} \\ & + D_{9,m}^{(1)}\delta_9^{(1)} + D_{9,m}^{(2)}\delta_9^{(2)} + D_{9,m}^{(3)}\delta_9^{(3)} + E_{9,m}\epsilon_9) + \mathcal{O}(\tau^{11})\Big\}. \end{aligned} \tag{A12}$$

To obtain the polynomials in this expression we solve the recursion $S^{(m+1)} = S_2(w_{m+1}\tau)S^{(m)}(\tau)S_2(w_{m+1})$, where

$$S_2(w_{m+1}\tau)S^{(m)}(\tau)S_2(w_{m+1}\tau) = \exp\left\{\tau w_{m+1}\alpha_1 + \tau^3 w_{m+1}^3\alpha_3 + \tau^5 w_{m+1}^5\alpha_5 + \tau^7 w_{m+1}^7\alpha_7 + \tau^9 w_{m+1}^9\alpha_9 + O(\tau^{11})\right\}$$

$$\times \exp\Big\{\tau A_{1,m}\alpha_1 + \tau^3 A_{3,m}\alpha_3 + \tau^5(A_{5,m}\alpha_5 + B_{5,m}\beta_5)$$
$$+ \tau^7(A_{7,m}\alpha_7 + B_{7,m}\beta_7 + C_{7,m}\gamma_7 + D_{7,m}\delta_7)$$
$$+ \tau^9(A_{9,m}\alpha_9 + B_{9,m}\beta_{9,m} + C^{(1)}_{9,m}\gamma^{(1)}_9 + C^{(2)}_{9,m}\gamma^{(2)}_9 + C^{(3)}_{9,m}\gamma^{(3)}_9$$
$$+ D^{(1)}_{9,m}\delta^{(1)}_9 + D^{(2)}_{9,m}\delta^{(2)}_9 + D^{(3)}_{9,m}\delta^{(3)}_9 + E_{9,m}\epsilon_9) + \mathcal{O}(\tau^{11})\Big\}$$
$$\times \exp\Big\{\tau w_{m+1}\alpha_1 + \tau^3 w^3_{m+1}\alpha_3 + \tau^5 w^5_{m+1}\alpha_5 + \tau^7 w^7_{m+1}\alpha_7 + \tau^9 w^9_{m+1}\alpha_9 + \mathcal{O}(\tau^{11})\Big\}. \tag{A13}$$

To solve the recursion, we express $S^{(m+1)}$ using the expression from Eq. (A12). Then we compute the right hand side of Eq. (A13) applying the symmetric BCH formula from Corollary 3. Writing the right hand side as $e^X e^Y e^X$, we have that

$$X = \tau w_{m+1}\alpha_1 + \tau^3 w^3_{m+1}\alpha_3 + \tau^5 w^5_{m+1}\alpha_5 + \tau^7 w^7_{m+1}\alpha_7 + \tau^9 w^9_{m+1}\alpha_9 + O(\tau^{11}) \tag{A14}$$
$$Y = \tau A_{1,m}\alpha_1 + \tau^3 A_{3,m}\alpha_3 + \tau^5(A_{5,m}\alpha_5 + B_{5,m}\beta_5) + \tau^7(A_{7,m}\alpha_7 + B_{7,m}\beta_7 + C_{7,m}\gamma_7 + D_{7,m}\delta_7)$$
$$+ \tau^9(A_{9,m}\alpha_9 + B_{9,m}\beta_{9,m} + C^{(1)}_{9,m}\gamma^{(1)}_9 + C^{(2)}_{9,m}\gamma^{(2)}_9 + C^{(3)}_{9,m}\gamma^{(3)}_9 + D^{(1)}_{9,m}\delta^{(1)}_9 + D^{(2)}_{9,m}\delta^{(2)}_9 + D^{(3)}_{9,m}\delta^{(3)}_9 + E_{9,m}\epsilon_9)$$
$$+ \mathcal{O}(\tau^{11}). \tag{A15}$$

We then compute the commutators of X and Y that appear in the symmetric BCH formula, here we give the resulting 9th order operators after applying the commutators. When we write $[X, Y, \cdots, X]_9$, the subscript indicates that we are only keeping the 9th order terms when expanding the commutator.

$$[Y,Y,X]_9 = \tau^9(A^2_{1,m}w^7_{m+1} - A_{1,m}A_{7,m}w_{m+1})\beta_9 - \tau^9 A_{1,m}B_{7,m}w_{m+1}\delta^{(1)}_9$$
$$+ \tau^9 A_{1,m}C_{7,m}w_{m+1}\delta^{(3)}_9 - \tau^9 A_{1,m}D_{7,m}w_{m+1}\epsilon_9$$
$$+ \tau^9(A_{1,m}A_{3,m}w^5_{m+1} - A_{1,m}A_{5,m}w^3_1)\gamma^{(1)}_9 - \tau^9 A_{1,m}B_{5,m}w^3_{m+1}\delta^{(3)}_9$$
$$+ \tau^9(A_{3,m}A_{1,m}w^5_{m+1} - A_{3,m}A_{5,m}w_{m+1})\gamma^{(2)}_9 - \tau^9 A_{3,m}B_{5,m}w_{m+1}\delta^{(2)}_9$$
$$+ \tau^9(A_{5,m}A_{1,m}w^3_{m+1} - A_{5,m}A_{3,m}w^2_{m+1})\gamma^{(3)}_9$$
$$+ \tau^9(B_{5,m}A_{1,m}w^3_{m+1} - B_{5,m}A_{3,m}w_{m+1})(\delta^{(2)}_9 - \delta^{(3)}_9) \tag{A16}$$
$$[X,X,Y]_9 = \tau^9(w^2_{m+1}A_{7,m} - w^8_{m+1}A_{1,m})\beta_9 + \tau^9 w^2_{m+1}B_{7,m}\delta^{(1)}_9$$
$$- \tau^9 w^2_{m+1}C_{7,m}\delta^{(3)}_9 + \tau^9 w^2_{m+1}D_{7,m}\epsilon_9$$
$$+ \tau^9(w^4_{m+1}A_{5,m} - w^6_{m+1}A_{3,m})\gamma^{(1)}_9 + \tau^9 w^4_{m+1}B_{5,m}\delta^{(3)}_9$$
$$+ \tau^9(w^4_{m+1}A_{5,m} - w^8_{m+1}A_{1,m})\gamma^{(2)}_9 + \tau^9 w^4_{m+1}B_{5,m}\delta^{(2)}_9$$
$$+ \tau^9(w^6_{m+1}A_{3,m} - w^8_{m+1}A_{1,m})\gamma^{(3)}_9 \tag{A17}$$
$$[X,X,X,X,Y]_9 = \tau^9(w^4_{m+1}A_{5,m} - w^8_{m+1}A_{1,m})\delta^{(1)}_9 + \tau^9 A^3_{1,m}B_{5,m}w_{m+1}\epsilon_9$$
$$+ \tau^9(w^6_{m+1}A_{3,m} - w^8_{m+1}A_{1,m})\delta^{(2)}_9$$
$$+ \tau^9 2(w^6_{m+1}A_{3,m} - w^8_{m+1}A_{1,m})\delta^{(3)}_9 \tag{A18}$$
$$[Y,Y,Y,Y,X]_9 = \tau^9(A^4_{1,m}w^5_{m+1} - A^3_{1,m}A_{5,m}w_{m+1})\delta^{(1)}_9 - \tau^9 A^3_{1,m}B_{5,m}w_{m+1}\epsilon_9$$
$$+ \tau^9(A_{3,m}A^3_{1,m}w^3_{m+1} - A^2_{3,m}A^2_{1,m}w_{m+1})\delta^{(2)}_9$$
$$+ \tau^9 2(A^3_{1,m}A_{3,m}w^3_{m+1} - A^2_{1,m}A^2_{3,m}w_{m+1})\delta^{(3)}_9 \tag{A19}$$
$$[X,Y,Y,Y,X]_9 = \tau^9(w^6_{m+1}A^3_{1,m} - w^2_{m+1}A^2_{1,m}A_{5,m})\delta^{(1)}_9 - \tau^9 A^2_{1,m}B_{5,m}w^2_{m+1}\epsilon_9$$
$$+ \tau^9(w^6_{m+1}A^3_{1,m} - w^4_{m+1}A^2_{1,m}A_{3,m})\delta^{(2)}_9$$
$$+ \tau^9 2(w^4_{m+1}A_{3,m}A^2_{1,m} - w^2_{m+1}A^2_{3,m}A_{1,m})\delta^{(3)}_9 \tag{A20}$$
$$[Y,X,X,X,Y]_9 = \tau^9(A_{1,m}A_{5,m}w^3_{m+1} - A^2_{1,m}w^7_{m+1})\delta^{(1)}_9 + \tau^9 w^3_{m+1}A_{1,m}B_{5,m}\epsilon_9$$

$$
\begin{aligned}
&+\tau^9(A_{3,m}^2 w_{m+1}^3 - A_{1,m}A_{3,m}w_{m+1}^5)\delta^{(2)} \\
&+\tau^9 2(A_{1,m}A_{3,m}w_{m+1}^5 - A_{1,m}^2 w_{m+1}^7)\delta_9^{(3)} \qquad \text{(A21)} \\
[X,X,Y,Y,X]_9 &= \tau^9(w_{m+1}^7 A_{1,m}^2 - w_{m+1}^3 A_{1,m}A_{5,m})\delta_9^{(1)} - w_{m+1}^3 A_{1,m}B_{5,m}\epsilon_9 \\
&+\tau^9(w_{m+1}^7 A_{1,m}^2 - w_{m+1}^5 A_{1,m}A_{3,m})\delta_9^{(2)} \\
&+\tau^9(w_{m+1}^5 A_{1,m}A_{3,m} - w_{m+1}^3 A_{3,m}^2)\delta_9^{(3)} \qquad \text{(A22)} \\
[Y,Y,X,X,Y]_9 &= \tau^9(A_{1,m}^2 A_{5,m}w_{m+1}^2 - A_{1,m}^3 w_{m+1}^6)\delta_9^{(1)} + A_{1,m}^2 B_{5,m}w_{m+1}^2\epsilon_9 \\
&+(A_{3,m}^2 A_{1,m}w_{m+1}^2 - A_{3,m}A_{1,m}^2 w_{m+1}^4)\delta_9^{(2)} \\
&+\tau^9(A_{3,m}^2 A_{1,m}w_{m+1}^2 - A_{3,m}A_{1,m}^2 w_{m+1}^4)\delta_9^{(3)} + \tau^9(A_{1,m}^2 A_{3,m}w_{m+1}^4 - A_{1,m}^3 w_{m+1}^6)\delta_9^{(3)} \qquad \text{(A23)} \\
[X,X,X,X,X,X,Y]_9 &= \tau^9(w_{m+1}^6 A_{3,m} - w_{m+1}^8 A_{1,m})\epsilon_9 \\
[Y,X,X,X,X,X,Y]_9 &= \tau^9(w_{m+1}^5 A_{1,m}A_{3,m} - w_{m+1}^7 A_{1,m}^2)\epsilon_9 \qquad \text{(A24)} \\
[Y,Y,X,X,X,X,Y]_9 &= \tau^9(A_{1,m}^2 A_{3,m}w_{m+1}^4 - A_{1,m}^3 w_{m+1}^6)\epsilon_9 \qquad \text{(A25)} \\
[Y,Y,Y,X,X,X,Y]_9 &= \tau^9(A_{1,m}^3 A_{3,m}w_{m+1}^3 - A_{1,m}^4 w_{m+1}^5)\epsilon_9 \qquad \text{(A26)} \\
[Y,Y,Y,Y,X,X,Y]_9 &= \tau^9(A_{1,m}^4 A_{3,m}w_{m+1}^2 - A_{1,m}^5 w_{m+1}^4)\epsilon_9 \qquad \text{(A27)} \\
[Y,Y,Y,Y,Y,X,Y]_9 &= \tau^9(A_{1,m}^5 A_{3,m}w_{m+1} - A_{1,m}^6 w_{m+1}^3)\epsilon_9 \qquad \text{(A28)}
\end{aligned}
$$

As in the 6th order case given by Yoshida [7], with these commutators we can write down the equations based on the recursion equation (A13),

$$A_{9,m+1} = A_{9,m} + 2w_{m+1}^9 \qquad \text{(A29)}$$

$$
\begin{aligned}
B_{9,m+1} &= B_{9,m} + \frac{1}{6}(A_{1,m}^2 w_{m+1}^7 - A_{1,m}A_{7,m}w_{m+1}) \\
&- \frac{1}{6}(A_{7,m}w_{m+1}^2 - A_{1,m}w_{m+1}^8) \qquad \text{(A30)}
\end{aligned}
$$

$$
\begin{aligned}
C_{9,m+1}^{(1)} &= C_{9,m}^{(1)} + \frac{1}{6}(A_{3,m}^2 A_{1,m}w_{m+1}^5 - A_{1,m}A_{5,m}w_{m+1}^3) \\
&- \frac{1}{6}(A_{5,m}w_{m+1}^4 - A_{3,m}w_{m+1}^6) \qquad \text{(A31)}
\end{aligned}
$$

$$
\begin{aligned}
C_{9,m+1}^{(2)} &= C_{9,m}^{(2)} + \frac{1}{6}(A_{3,m}^2 A_{1,m}w_{m+1}^5 - A_{3,m}A_{5,m}w_{m+1}) \\
&- \frac{1}{6}(A_{5,m}w_{m+1}^4 - A_{1,m}w_{m+1}^8) \qquad \text{(A32)}
\end{aligned}
$$

$$
\begin{aligned}
C_{9,m+1}^{(3)} &= C_{9,m}^{(3)} + \frac{1}{6}(A_{5,m}A_{1,m}w_{m+1}^3 - A_{3,m}A_{5,m}w_{m+1}) \\
&- \frac{1}{6}(A_{3,m}w_{m+1}^6 - A_{1,m}w_{m+1}^8) \qquad \text{(A33)}
\end{aligned}
$$

$$
\begin{aligned}
D_{9,m+1}^{(1)} &= D_{9,m}^{(1)} - \frac{1}{6}(A_{1,m}B_{7,m}w_{m+1} + w_{m+1}^2 B_{7,m}) \\
&+ \frac{7}{360}(A_{5,m}w_{m+1}^4 - w_{m+1}^8 A_{1,m}) \\
&- \frac{1}{360}(A_{1,m}^4 w_{m+1}^5 - A_{1,m}^3 A_{5,m}w_{m+1}) \\
&+ \frac{1}{90}(A_{1,m}^3 w_{m+1}^6 - A_{1,m}^2 A_{5,m}w_{m+1}^2) \\
&+ \frac{1}{45}(A_{1,m}A_{5,m}w_{m+1}^3 - A_{1,m}^2 w_{m+1}^7) \\
&- \frac{1}{60}(A_{1,m}^2 w_{m+1}^7 - A_{1,m}A_{5,m}w_{m+1}^3) \\
&+ \frac{1}{30}(A_{1,m}^2 A_{5,m}w_{m+1}^2 - A_{1,m}^3 w_{m+1}^6) \qquad \text{(A34)}
\end{aligned}
$$

$$D_{9,m+1}^{(2)} = D_{9,m}^{(2)} - \frac{1}{6}(A_{3,m}B_{5,m}w_{m+1} + w_{m+1}^4 B_{5,m})$$

$$
\begin{aligned}
&+\frac{7}{360}(A_{3,m}w_{m+1}^6 - w_{m+1}^8 A_{1,m})\\
&-\frac{1}{360}(A_{1,m}^3 A_{3,m} w_{m+1}^3 - A_{1,m}^2 A_{3,m}^2 w_{m+1})\\
&+\frac{1}{90}(A_{1,m}^3 w_{m+1}^6 - A_{1,m}^2 A_{3,m} w_{m+1}^4)\\
&+\frac{1}{45}(A_{3,m}^2 w_{m+1}^3 - A_{1,m}A_{3,m}w_{m+1}^5)\\
&-\frac{1}{60}(A_{1,m}^2 w_{m+1}^7 - A_{1,m}A_{3,m}w_{m+1}^5)\\
&+\frac{1}{30}(A_{3,m}^2 A_{1,m} w_{m+1}^2 - A_{1,m}^2 A_{3,m} w_{m+1}^4) \qquad \text{(A35)}
\end{aligned}
$$

$$
\begin{aligned}
D_{9,m+1}^{(3)} = D_{9,m}^{(3)} &- \frac{1}{6}(A_{1,m}B_{5,m}w_{m+1}^3 + w_{m+1}^4 B_{5,m})\\
&+\frac{14}{360}(A_{3,m}w_{m+1}^6 - w_{m+1}^8 A_{1,m})\\
&-\frac{2}{360}(A_{1,m}^3 A_{3,m} w_{m+1}^3 - A_{1,m}^2 A_{3,m}^2 w_{m+1})\\
&+\frac{2}{90}(A_{1,m}^2 A_{3,m} w_{m+1}^4 - A_{1,m}A_{3,m}^2 w_{m+1}^2)\\
&+\frac{2}{45}(A_{3,m}A_{1,m}w_{m+1}^5 - A_{1,m}^2 w_{m+1}^7)\\
&-\frac{1}{60}(A_{1,m}^2 w_{m+1}^7 - A_{1,m}A_{3,m}w_{m+1}^5)\\
&+\frac{1}{30}(A_{3,m}^2 A_{1,m} w_{m+1}^2 - A_{1,m}^2 A_{3,m} w_{m+1}^4)\\
&+\frac{1}{6}(A_{1,m}C_{7,m}w_{m+1} + w_{m+1}^2 c_{7,m})\\
&-\frac{1}{60}(w_{m+1}^5 A_{1,m}A_{3,m} - w_{m+1}^3 A_{3,m}^2)\\
&+\frac{1}{30}(A_{1,m}^2 A_{3,m} w_{m+1}^4 - A_{1,m}^3 w_{m+1}^6)\\
&-\frac{1}{6}(B_{5,m}A_{1,m}w_{m+1}^3 - B_{5,m}A_{3,m}w_{m+1}) \qquad \text{(A36)}
\end{aligned}
$$

$$
\begin{aligned}
E_{9,m+1} = E_{9,m} &- \frac{1}{6}(A_{1,m}D_{7,m} - w_{m+1}^2 D_{7,m})\\
&+\frac{7}{360}w_{m+1}^4 B_{5,m}\\
&+\frac{1}{360}A_{1,m}^3 B_{5,m} w_{m+1}\\
&-\frac{1}{90}A_{1,m}^2 B_{5,m} w_{m+1}^2\\
&+\frac{1}{45}A_{1,m}B_{5,m}w_{m+1}^3\\
&+\frac{1}{60}A_{1,m}B_{5,m}w_{m+1}^3\\
&+\frac{1}{30}A_{1,m}^2 B_{5,m} w_{m+1}^2\\
&-\frac{31}{15120}(w_{m+1}^6 A_{3,m} - w_{m+1}^8 A_{1,m})\\
&-\frac{31}{5040}(w_{m+1}^5 A_{1,m}A_{3,m} - w_{m+1}^7 A_{1,m}^2)\\
&-\frac{13}{1890}(A_{1,m}^2 A_{3,m} w_{m+1}^4 - A_{1,m}^3 w_{m+1}^6)\\
&-\frac{53}{15120}(A_{1,m}^3 A_{3,m} w_{m+1}^3 - A_{1,m}^4 w_{m+1}^5)
\end{aligned}
$$

$$- \frac{1}{1260}(A_{1,m}^4 A_{3,m} w_{m+1}^2 - A_{1,m}^5 w_{m+1}^4)$$
$$- \frac{1}{15120}(A_{1,m}^5 A_{3,m} w_{m+1} - A_{1,m}^6 w_{m+1}^3). \quad \text{(A37)}$$

Then we obtain the polynomial equations for the tenth order product formula by imposing that $A_{1,m} = 1$ and all other terms are equal to zero. It can be seen that $C_{9,m}^2 = C_{9,m}^1 + C_{9,m}^3$, eliminating one equation and providing a set of 15 equations to solve.

Appendix B: Bounding error by total time evolution

In this section we show how the total evolution time appears in the error bound of a product formula. For an operator $W = X + Y$, with X and Y non-commuting, we want to bound the expression $\|S^{(m)}(t) - e^{tW}\|$. We assume $S^{(m)}(t)$ is a kth order product formula following Yoshida's ansatz.

We shall consider the Taylor expansion of both $S^{(m)}(t)$ and e^{tW}

$$S^{(m)}(t) = \mathcal{T}_k[S^{(m)}] + \mathcal{T}_{k\leq}[S^{(m)}] \quad \text{(B1)}$$
$$e^{tW} = \mathcal{T}_k[e^{tW}] + \mathcal{T}_{k\leq}[e^{tW}], \quad \text{(B2)}$$

where $\mathcal{T}_k$ is defined as the Taylor expansion up to order k and $\mathcal{T}_{k\leq}$ has the terms of the Taylor expansion from order $k+1$ to infinity. Note that since $S^{(m)}$ is a kth order product formula then $\mathcal{T}_k[S^{(m)}] = \mathcal{T}_k[e^{tW}]$ and thus

$$\left\|S^{(m)}(t) - e^{tW}\right\| = \left\|\mathcal{T}_{k\leq}[S^{(m)}] - \mathcal{T}_{k\leq}[e^{tW}]\right\| \quad \text{(B3)}$$
$$\leq \left\|\mathcal{T}_{k\leq}[S^{(m)}]\right\| + \left\|\mathcal{T}_{k\leq}[e^{tW}]\right\|. \quad \text{(B4)}$$

Now, it can be seen that the following bounds apply

$$\left\|\mathcal{T}_{k\leq}[S^{(m)}]\right\| \leq \sum_{r=2k+1}^{\infty} \frac{(\xi t\Lambda)^r}{r!} \quad \text{(B5)}$$
$$\leq \frac{(\xi t\Lambda)^{2k+1}}{(2k+1)!} \sum_{r=0}^{\infty} \frac{(\xi t\Lambda)^r}{r!} \quad \text{(B6)}$$
$$= \frac{(\xi t\Lambda)^{2k+1}}{(2k+1)!} \exp(\xi t\Lambda), \quad \text{(B7)}$$

where $\Lambda = \|X\| + \|Y\|$ and $\xi = |w_0| + 2\sum_{i=1}^m |w_i|$ corresponds to the total evolution time. The first inequality comes from expanding out $\mathcal{T}_{k\leq}[S^{(m)}]$ and applying the triangle inequality to each term of the expansion, so that each operator X or Y in the sum becomes a scalar $\|X\|$ or $\|Y\|$ respectively. Similarly, we have

$$\left\|\mathcal{T}_{k\leq}[e^{tW}]\right\| = \sum_{r=2k+1}^{\infty} \frac{(t\|W\|)^r}{r!} \quad \text{(B8)}$$
$$\leq \frac{(t\|W\|)^{2k+1}}{(2k+1)!} \sum_{r=0}^{\infty} \frac{(t\|W\|)^r}{r!} \quad \text{(B9)}$$
$$= \frac{(t\|W\|)^{2k+1}}{(2k+1)!} \exp\{t\|W\|\}. \quad \text{(B10)}$$

Therefore the total error for the product formula may be bounded as

$$\left\|S^{(m)}(t) - e^{tW}\right\| \leq \frac{(\xi t\Lambda)^{2k+1}}{(2k+1)!} \exp(\xi t\Lambda) + \frac{(t\|W\|)^{2k+1}}{(2k+1)!} \exp(t\|W\|). \quad \text{(B11)}$$

It can be seen from this expression that the total evolution time ξ appears in this bound and in the simulations of Section IV the other quantities in the bound such as the norm of the operators are constant, then we can expect ξ to be correlated with the constant factor in the error.