

Fast and differentiable simulation of driven quantum systems

Ross Shillito,^{1,*} Jonathan A. Gross,^{1,†} Agustin Di Paolo,^{1,‡} Élie Genois,¹ and Alexandre Blais^{1,2}

¹*Institut quantique & Département de Physique, Université de Sherbrooke, Sherbrooke J1K2R1, Quebec, Canada*

²*Canadian Institute for Advanced Research, Toronto, M5G1M1 Ontario, Canada*

(Dated: December 18, 2020)

The controls enacting logical operations on quantum systems are described by time-dependent Hamiltonians that often include rapid oscillations. In order to accurately capture the resulting time dynamics in numerical simulations, a very small integration time step is required, which can severely impact the simulation run-time. Here, we introduce a semi-analytic method based on the Dyson expansion that allows us to time-evolve driven quantum systems much faster than standard numerical integrators. This solver, which we name `DySolve`, efficiently captures the effect of the highly oscillatory terms in the system Hamiltonian, significantly reducing the simulation's run time as well as its sensitivity to the time-step size. Furthermore, this solver provides the exact derivative of the time-evolution operator with respect to the drive amplitudes. This key feature allows for optimal control in the limit of strong drives and goes beyond common pulse-optimization approaches that rely on rotating-wave approximations. As an illustration of our method, we show results of the optimization of a two-qubit gate using transmon qubits in the circuit QED architecture.

I. INTRODUCTION

High-fidelity logical gates are paramount to the realization of useful quantum computation. It is important in the development of these gates that they be simulated with great precision to ensure that they meet the particularly strict requirements for fault-tolerant quantum computation. Several techniques are used for simulating the time dynamics of quantum devices, including dynamical solvers such as Runge-Kutta integrators and direct matrix exponentiation [1, 2]. However, capturing the full time dynamics with the necessary accuracy requires integration methods with a sufficiently small time step. As a result, simulations can be computationally very expensive, even for relatively simple cases such as optimizing a two-qubit gate.

To ensure that simulations of quantum systems are feasible, approximations must be made. For example, a common approximation is to neglect counter-rotating terms within the rotating wave approximation (RWA) to greatly reduce the complexity of the simulation whilst capturing the dominant dynamics. This approximation renders the Hamiltonian time independent, which can then simply be exponentiated to obtain the propagator. However, effects such as the Bloch-Siegert shift which do not appear under a RWA need to be taken into account to accurately model the system [3]. In addition, many gate optimization methods, such as GRAPE, require the calculation of gradients, something which greatly adds to the complexity of the numerical calculations [4]. More specifically, when including the effects of the counter-rotating terms, there exists no simple derivative of time-ordered unitaries with respect to the drive amplitudes,

and one must resort to approximating the gradients. Consequently, this approach may not converge to the optimal solution, which is problematic when targeting very high-fidelity gates.

In this work, we develop an algorithm based on a Dyson series expansion of the time ordered problem that addresses all of the above difficulties. In this approach, which we call `DySolve`, the time-ordered unitary evolution operator is written as a product of time-independent operators which are weighted by the drive amplitudes and dynamical phases. This algorithm captures the full fast-oscillatory dynamics irrespective of the integration step size, thereby decreasing the complexity of the numerical problem. This also greatly decreases the simulation time in comparison to traditional integration-based solvers without loss of numerical precision. Importantly, this approach trivializes the derivative with respect to the drive strength, which can be calculated to an accuracy equivalent to the order of the Dyson series. Moreover, this approach is compatible with non-Hermitian dynamics, allowing for the simulation of open quantum systems.

We begin by introducing our approach in Sec. II in the case of a single, sinusoidal drive with a constant amplitude, and then extend the formalism to the case of multiple drives, accounting for filtering effects on envelope functions. We then define the `DySolve` algorithm in Sec. III, and demonstrate its application to driven quantum systems with random drive envelopes. We proceed to apply our algorithm to the GRAPE optimization routine in Sec. IV, and show as an example optimized two-qubit gates in the circuit QED architecture.

II. OSCILLATORY DRIVE PROBLEM

A. Simple time-dependent Hamiltonians

We begin by considering a simple time-dependent Hamiltonian with a cosinusoidally oscillating control

* Ross.Shillito@USherbrooke.ca

† Jarthurgross@google.com

‡ adipaolo@mit.edu

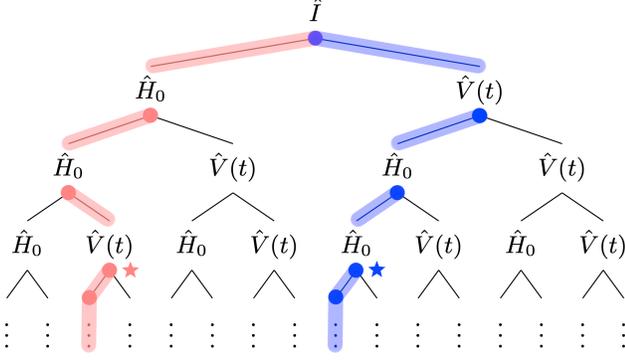


Figure 1. Tree diagram showing the different branches of the time ordered integral, with two example branches highlighted.

drive term

$$\hat{H}(t) = \hat{H}_0 + \hat{V}(t), \quad \hat{V}(t) = \hat{X} \cos(\omega t). \quad (1)$$

Here, $\hat{H}_0 = \sum_k \lambda_k |k\rangle\langle k|$ is a generic system Hamiltonian expressed in its eigenbasis, while \hat{X} is a dipole operator that connects the eigenstates of \hat{H}_0 and which we take to account for the amplitude of the drive. As will become important later, we note that we are not using the RWA.

The propagator under $\hat{H}(t)$ for some time increment δt takes the usual form of a time-ordered integral

$$\hat{U}(t, t + \delta t) = \mathcal{T} \exp\left(-i \int_t^{t+\delta t} dt' \hat{H}(t')\right), \quad (2)$$

with \mathcal{T} the time-ordering operator. Due to the fast oscillatory $\cos(\omega t)$ term, evaluating the propagator $\hat{U}(t, t + \delta t)$ is a numerically challenging problem, and there exists few analytic solutions to even the simplest case of a driven two level qubit [5]. In special cases, one can invoke the RWA to remove the explicit time dependence from the Hamiltonian, thereby allowing for calculation of the propagator from matrix exponentiation directly [6].

However, when the RWA cannot be used due to a breakdown of the approximation or because a greater degree of accuracy is needed, we propose using a truncated Dyson series. Consider Eq. (2), written using the definition of the time-ordering operator

$$\begin{aligned} \hat{U}(t, t + \delta t) = & \\ & \sum_{n=0}^{\infty} (-i)^n \int_t^{t+\delta t} \int_t^{t_n} \dots \int_t^{t_2} \hat{H}(t_n) \dots \hat{H}(t_1) dt_1 \dots dt_n. \end{aligned} \quad (3)$$

It is useful to express this expansion in terms of powers of the drive operator $\hat{V}(t)$, forming the Dyson series

$$\hat{U}(t, t + \delta t) = \sum_{n=0}^{\infty} \hat{U}^{(n)}(t, t + \delta t), \quad (4)$$

where we have defined

$$\hat{U}^{(n)}(t, t + \delta t) = \sum_{\omega_n} \exp\left(i \sum_{p=1}^n \omega_n [p] t\right) \hat{S}^{(n)}(\omega_n, \delta t). \quad (5)$$

Here, ω_n is an n -vector whose entries are $\pm\omega$ originating from the decomposition of the $\cos(\omega t)$ of the control into complex exponentials, and $\omega_n [p]$ is the p -th element of ω_n . The sum over ω_n implies a summation over all 2^n possible ω_n vectors.

Equation (5) also introduces the Dyson series operator $\hat{S}^{(n)}(\omega_n, \delta t)$ which takes the form

$$\hat{S}^{(n)}(\omega_n, \delta t) = \frac{1}{2^n} \sum_{\mathbf{m} \in \mathbf{Z}_+^{n+1}} \hat{S}_{\mathbf{m}}^{(n)}(\omega_n, \delta t), \quad (6)$$

and which corresponds to a summation over all Dyson path operators of n -th order, where $\mathbf{m} = [m_n, \dots, m_0]$ with each index m_i ranging from 0 to infinity. These n -th order path operators are given by

$$\begin{aligned} \hat{S}_{\mathbf{m}}^{(n)}(\omega_n, \delta t) = & \\ & \int_0^{\delta t} \int_0^{t'_M} \dots \int_0^{t'_2} (-i \hat{H}_0)^{m_n} \hat{X} (-i \hat{H}_0)^{m_{n-1}} \dots \hat{X} (-i \hat{H}_0)^{m_0} \\ & \times (-i)^n \prod_{p=1}^n \exp(i \omega_n [p] t_{\iota(p)}) dt'_1 \dots dt'_M \end{aligned} \quad (7)$$

where we have introduced

$$M = \sum_{i=0}^n m_i + n, \quad \iota(p) = \sum_{j=0}^{p-1} m_j + p. \quad (8)$$

Each $\hat{S}_{\mathbf{m}}^{(n)}(\omega_n, \delta t)$ correspond to different ways to have n contributions from the control \hat{X} (i.e. different terminated branches of the tree diagram), with the m_i 's labeling the number of applications of \hat{H}_0 before the subsequent application of \hat{X} . The total number of operators in the path operator is given by M . To simplify the notation, we introduce an indexing function $\iota(p)$, which can be interpreted as the total number of \hat{H}_0 and \hat{X} operators before the p -th application of the control \hat{X} . These definitions can be visualized as in Fig. 1 where the red and blue branches correspond to sets of path operators $\hat{S}_{[m_1, 2]}^{(n)}(\omega_n, \delta t)$ and $\hat{S}_{[m_1, 0]}^{(n)}(\omega_n, \delta t)$ respectively, with m_1 determined by where the path terminates. For example, $\hat{S}_{[0, 2]}^{(n)}(\omega_n, \delta t)$ and $\hat{S}_{[2, 0]}^{(n)}(\omega_n, \delta t)$ terminate at the points in the branches marked by a star.

Below, we give explicit expressions for these operators to zeroth and first order in the control. Building on these results, we then construct expressions that are easily amenable to efficient numerical evaluation to arbitrary orders.

B. Evaluation to zeroth and first order

The zeroth order corresponds to the leftmost branch of the tree diagram of Fig. 1 for which it is straightforward to obtain an explicit expression. Indeed, the path operator simply takes the form

$$\begin{aligned}\hat{S}_{[m_0]}^{(0)}(\mathbf{0}, \delta t) &= \int_0^{\delta t} \int_0^{t_{m_0}} \cdots \int_0^{t_2} (-i\hat{H}_0)^{m_0} dt_1 \cdots dt_{m_0-1} dt_{m_0} \\ &= \frac{(-i\hat{H}_0 \delta t)^{m_0}}{m_0!}.\end{aligned}\quad (9)$$

Summing all of the elements in the branch, we obtain

$$\hat{S}^{(0)}(\mathbf{0}, \delta t) = \sum_{m_0=0}^{\infty} \hat{S}_{[m_0]}^{(0)}(\mathbf{0}, \delta t) = e^{-i\hat{H}_0 \delta t}, \quad (10)$$

which corresponds, as expected, to the drift evolution of the Hamiltonian in the absence of drive. In a similar fashion, the path operator to first order takes the form

$$\begin{aligned}\hat{S}_{[m_1, m_0]}^{(1)}(\boldsymbol{\omega}_1, \delta t) &= \int_0^{\delta t} \int_0^{t_M} \cdots \int_0^{t_2} (-i\hat{H}_0)^{m_1} \hat{X} (-i\hat{H}_0)^{m_0} \\ &\times \exp(\pm i[\boldsymbol{\omega}]t_{i(1)}) dt_1 \cdots dt_M,\end{aligned}\quad (11)$$

and therefore leads to the following Dyson series operator

$$\hat{S}^{(1)}(\boldsymbol{\omega}_1, \delta t) = -\frac{i\delta t}{2} \sum_{k, k'} f(\lambda_k \delta t, (\lambda_{k'} \mp \boldsymbol{\omega}) \delta t) \langle k | \hat{X} | k' \rangle |k\rangle \langle k'|. \quad (12)$$

This operator weights the matrix elements of \hat{X} by a function f whose inputs are weighted eigenvalues $\{\lambda_k\}$ of the free Hamiltonian \hat{H}_0 with corresponding eigenstates $\{|k\rangle\}$, where the second eigenvalue is shifted by the drive frequency $\boldsymbol{\omega}_1[1] = \pm\boldsymbol{\omega}$. This function is defined as

$$f(\lambda_k, \lambda_{k'}) = \frac{i}{\lambda_k - \lambda_{k'}} (e^{-i\lambda_k} - e^{-i\lambda_{k'}}). \quad (13)$$

We refer to this function as the first order weighting function. The above expressions are derived in the Appendix, and are unsurprisingly in exact agreement with first-order time-dependent perturbation theory.

Crucially, the Dyson operators depend on the size δt of the time increment, but not the current time of the evolution t . As a result, for a total evolution time $T = P\delta t$, where P is an integer, the set of P incremental evolution operators $\hat{U}(p\delta t, (p+1)\delta t)$ can be evaluated simultaneously. As will become clearer in the next section, this holds true to arbitrary order.

C. Evaluation to n-th order

To model quantum systems with sufficient accuracy, it is necessary to consider second- and higher-order terms

in the Dyson series. This can be done following a similar approach as described above. Indeed, we introduce the n -th order Dyson operator in a similar fashion to Eq. (12)

$$\begin{aligned}\hat{S}^{(n)}(\boldsymbol{\omega}_n, \delta t) &= \\ &\left(\frac{-i\delta t}{2}\right)^n \sum_{\mathbf{k}_n} f(\boldsymbol{\lambda}_n(\mathbf{k}_n)\delta t - \mathbf{c}(\boldsymbol{\omega}_n)\delta t) \langle k^{(n)} | \hat{X} | k^{(n-1)} \rangle \\ &\times \langle k^{(n-1)} | \hat{X} \cdots | k^{(1)} \rangle \langle k^{(1)} | \hat{X} | k^{(0)} \rangle |k^{(n)}\rangle \langle k^{(0)}|. \end{aligned}\quad (14)$$

Here, each $\mathbf{k}_n = (k^{(0)}, k^{(1)}, \dots, k^{(n)})$ is a set of indices which specify a set of $(n+1)$ eigenstates $\{|k^{(m)}\rangle\}$ of \hat{H}_0 with corresponding eigenvalues $\boldsymbol{\lambda}_n(\mathbf{k}_n)$, and we sum over all possible \mathbf{k}_n . These eigenvalues are written in vector form:

$$\boldsymbol{\lambda}_n(\mathbf{k}_n) \equiv (\lambda_{k^{(0)}}, \dots, \lambda_{k^{(n)}}), \quad \hat{H}_0 |k^{(m)}\rangle = \lambda_{k^{(m)}} |k^{(m)}\rangle. \quad (15)$$

The sum over \mathbf{k}_n implies a summation over all sets of eigenstates which the dipole operator \hat{X} couples in Eq. (14). Additionally, we have introduced the cumulative vector

$$\mathbf{c}(\boldsymbol{\omega}_n) = \left(\sum_{p=0}^{n-1} \boldsymbol{\omega}_n[n-p], \sum_{p=0}^{n-2} \boldsymbol{\omega}_n[n-p], \dots, \boldsymbol{\omega}_n[n], 0 \right). \quad (16)$$

The n -th order weighting function $f(\boldsymbol{\lambda}_n)$ entering Eq. (14) can be obtained recursively using the relation (see Appendix A)

$$f(\boldsymbol{\lambda}_n) = i \frac{f(\mathbf{g}(\boldsymbol{\lambda}_n)) - f(\mathbf{g}^2(\boldsymbol{\lambda}_n) \cup \boldsymbol{\lambda}_n[n])}{\boldsymbol{\lambda}_n[n-1] - \boldsymbol{\lambda}_n[n]}, \quad (17)$$

where $\mathbf{g}(\mathbf{v}_n)$ returns \mathbf{v}_n without its last element, $\mathbf{g}^2(\mathbf{v}_n) = \mathbf{g}(\mathbf{g}(\mathbf{v}_n))$, and the notation \cup indicates appending an additional element to a vector such that

$$\boldsymbol{\lambda}_n = \mathbf{g}(\boldsymbol{\lambda}_n) \cup \boldsymbol{\lambda}_n[n]. \quad (18)$$

In the case of degenerate eigenvalues, we simply define Eq. (17) by taking the limit $\boldsymbol{\lambda}_n[n-1] \rightarrow \boldsymbol{\lambda}_n[n]$.

Using Eq. (4) with the above results, we can now form the truncated Dyson series yielding an approximation to the evolution operator to n -th order in the perturbation

$$\hat{U}_p \approx \sum_{r=0}^n \hat{U}_p^{(r)}, \quad \hat{U}_p^{(n)} \equiv \hat{U}^{(n)}(p\delta t, (p+1)\delta t). \quad (19)$$

thus yielding the total evolution operator for time $T = P\delta t$ to the same order

$$\hat{U}(0, P\delta t) \approx \mathcal{T} \prod_{p=0}^P \hat{U}_p \quad (20)$$

In this formalism, the time-ordering operator \mathcal{T} becomes a trivial operation since we need only arrange the set of matrices $\{\hat{U}_p\}$ in ascending order from right to left. Using this method, we can thus calculate an arbitrary number of terms in the Dyson series, with each subsequent order increasing the accuracy of the approximation to the propagator operator $\hat{U}(0, T)$.

D. Generalization to more complex drives

So far, we have only considered a single, cosinusoidal drive. In practice, for applications such as DRAG [7], it is useful to consider more complicated drives of the form

$$\begin{aligned}\hat{V}(t) &= [\Omega_x \cos(\omega t) + \Omega_y \sin(\omega t)] \hat{X}, \\ &= (\Omega e^{i\omega t} + \Omega^* e^{-i\omega t}) \hat{X},\end{aligned}\quad (21)$$

where $\Omega = \Omega_x + i\Omega_y$ is the complex drive amplitude. This generalization requires only a minor modification to the result of Eq. (5) which now reads

$$\hat{U}^{(n)}(t, t+\delta t) = \sum_{\omega_n} \exp\left(i \sum_{p=1}^n \omega_n[p]t\right) \Omega(\omega_n) \hat{S}^{(n)}(\omega_n, \delta t),\quad (22)$$

and where we have introduced

$$\Omega(\omega_n) = \Omega^{\mu(\omega_n)} \Omega^{*(n-\mu(\omega_n))}.\quad (23)$$

We see from Eq. (21) that Ω and Ω^* will appear according to the number of positive and negative frequency elements in the vector ω respectively, leading to a simple expression for $\mu(\omega_n)$ in $\Omega(\omega_n)$:

$$\mu(\omega_n) = (n + \sum_{p=1}^n \omega_n[p]) / (2\omega).\quad (24)$$

As shown in the Appendix, this formalism can be further extended to an arbitrary number of drives of different frequencies, and acting on different system operators.

E. Envelope Functions and Gaussian Filtering

Having established the necessary notation, we can now take into account control drives with time-dependent amplitudes $\Omega(t)$. To do so, we discretize the drive envelope of total time $T = N_p \Delta t$ into N_p increments, called pixels, each of duration Δt . For a given pixel i , the drive is characterized by its complex amplitude u_i such that the envelope can be expressed as [4]

$$\Omega(t) = \sum_{i=0}^{N_p-1} u_i \Pi(i\Delta t, (i+1)\Delta t),\quad (25)$$

where $\Pi(t, t + \Delta t) = \Theta(t) - \Theta(t - \Delta t)$, with $\Theta(t)$ the Heaviside function.

Additionally, following Motzoi *et al.* [8], we take into account the finite bandwidth of the control by applying a Gaussian filter on the discretized envelope. To do so, each pixel is subdivided in N_s subpixels of width δt with $\Delta t = N_s \delta t$, see Fig. 2. The subpixel amplitudes s_l are then defined as

$$s_l = \sum_{j=0}^{N_s-1} T_{l,j} u_j,\quad (26)$$

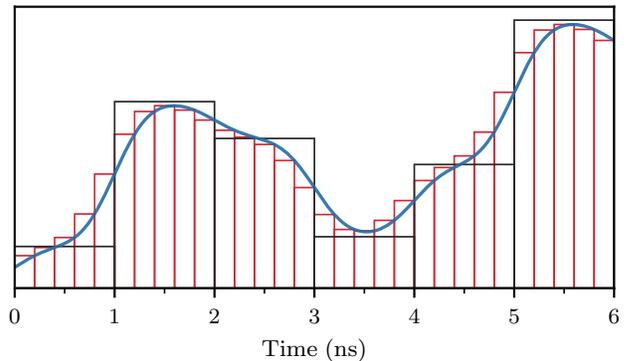


Figure 2. An example set of pulse amplitudes. The black bars indicate the chosen drive amplitudes u_j , where $\Delta t = 1$ ns. The red bars indicate the subpixels, which provide an approximate interpolation of the true pulse delivered to the system (blue), with a bandwidth $\omega_0/2\pi = 851$ MHz.

where the Gaussian filter matrix T has elements

$$\begin{aligned}T_{l,j} &= \frac{1}{2} \left\{ \operatorname{erf} \left[\omega_0 \left(\frac{l\delta t - j\Delta t}{2} \right) \right] \right. \\ &\quad \left. - \operatorname{erf} \left[\omega_0 \left(\frac{l\delta t - (j+1)\Delta t}{2} \right) \right] \right\}.\end{aligned}\quad (27)$$

Following Eq. (22), the n -th order evolution operator over the l -th subpixel takes the form

$$\hat{U}_l^{(n)} = \sum_{\omega_n} \exp\left(i \sum_{p=1}^n \omega_n[p]l\delta t\right) \Omega_l(\omega_n) \hat{S}^{(n)}(\omega_n, \delta t),\quad (28)$$

where the drive function in Eq. (23) picks up an additional subscript l to denote the l -th subpixel

$$\Omega_l(\omega_n) = s_l^{\mu(\omega_n)} s_l^{*(n-\mu(\omega_n))}.\quad (29)$$

As can be seen in Fig. 2, the subpixels (red) will often overestimate or underestimate the filtered pulse (blue) amplitude depending on its gradient, something which can become a leading contribution to the error in simulations. In Appendix C, we generalize the amplitudes s_l to have a linear time dependence to compensate for the change in amplitude of the continuous pulse across a single subpixel, providing a more accurate approximation to the filtered pulse.

III. THE DYSOLVE ALGORITHM

As already explained, the Dyson series operators $\hat{S}^{(n)}(\omega_n, \delta t)$ for which we have expressions at arbitrary order n are functions of the time increment δt and, crucially, are independent of the total evolution time T . The **Dysolve** algorithm leverages this fact to parallelize the time evolution.

The algorithm operates in two parts: a *preparation* stage and a *contraction* stage. In the preparation stage, the Dyson operators $\hat{S}^{(n)}(\boldsymbol{\omega}_n, \delta t)$ for a Hilbert space size N are computed up to a chosen truncation order n , and arranged in a tensor. This tensor has dimensions $(R \times N \times N)$, where R is the total number of Dyson operators. In the case of a single drive without linear interpolation, $R = 2^{n+1} - 1$, where n is the order of the expansion.

Once the preparation stage is completed, it is in principle possible to consider arbitrary gate times and drive envelopes. Suppose we wish to simulate a time-evolution of length $T = P\delta t$, where P is an integer. We first generate a $(P \times R)$ tensor whose elements are the envelopes and oscillatory terms $\exp\left(i \sum_{p=1}^n \boldsymbol{\omega}_n[p]l\delta t\right)\Omega_l(\boldsymbol{\omega}_n)$ in Eq. (28). We then multiply these tensors to obtain a $(P \times N \times N)$ time-evolution tensor, where the p -th $(N \times N)$ matrix corresponds to a time-step operator \hat{U}_p . This multiplication constitutes the parallelized portion of the algorithm, with all P time-evolution operators calculated simultaneously. As in Eq. (20), we multiply all of the individual evolution operators in the time evolution tensor to obtain the final evolution operator $\hat{U}(0, T)$. This whole procedure forms the contraction step of our algorithm.

Below, we will refer to the computation of the evolution operator $\hat{U}(0, T)$ to n -order following the above approach as `Dysolve-n`.

A. Benchmarking

Before turning to examples of application of `Dysolve`, we first benchmark this algorithm. To do so, there are a number of factors to consider: *i*) the size of the Hilbert space, *ii*) the drive amplitude, *iii*) the number of independent drive channels, and *iv*) the shape of the envelope function. To quantify the performance of our algorithm, we use two metrics. Given a number of subpixels N_s , we evaluate: 1) the computation time, and 2) the Frobenius norm distance between $\hat{U}(0, T)$, the propagator calculated under the `Dysolve` algorithm with a chosen number of subpixels, and a reference $\hat{U}_{\text{ref}}(0, T)$, the same unitary calculated with very high precision. As discussed further in Appendix D, we use `Dysolve-4` with 10^4 subpixels to compute $\hat{U}_{\text{ref}}(0, T)$, since `Dysolve` is able to reach precisions on the benchmark setup that are up to three orders of magnitude greater than QuTiP's `propagator`, a comparable dynamical solver [9].

For our benchmarks, we use diagonal system Hamiltonians with an Hilbert space size $N = 25$. As a concrete example, we take the eigenvalues to be normally distributed about 7 GHz, the typical operating frequency of superconducting qubits [10]. We consider between one and three input drive operators, at the frequency of the $|0\rangle \leftrightarrow |1\rangle$, $|2\rangle \leftrightarrow |3\rangle$ and $|4\rangle \leftrightarrow |5\rangle$ transitions with $|k\rangle$ the k -th lowest lying system eigenstate. The matrix elements of each operator corresponding to these transitions are set to 1. To emulate an arbitrary drive operator

with off-resonant terms, we populate 20% of the remaining matrix elements of the drive operators with complex numbers normally distributed about 0, after which hermiticity is enforced by the addition of the complex conjugate. The envelope function associated to each drive operator is centered around an amplitude such that the duration of the simulation is equivalent, in the absence of the other drives, to a total of 20 Rabi oscillations. In the context of superconducting qubits with their microwave drives, this corresponds to the simulation a 500 ns evolution with a drive amplitude of 40 MHz. We take the pixel amplitudes u_j of the envelope functions to be normally distributed about 40 MHz with a standard deviation of 1 MHz. To reduce statistical fluctuations, the results presented in Fig. 3 are averaged over 30 simulations, each with different random envelope functions and system eigenvalues.

The numerical simulations reported in Fig. 3 are performed on a 2.8 GHz, Intel Xeon Gold 6242 Processor (16 cores/32 threads) using `python`. Since the preparation stage only needs to be performed once for a particular system Hamiltonian, the reported computation time accounts only for the contraction stage of the `Dysolve` algorithm. We report the preparation times in the figure caption for reference. The top three panels of Fig. 3 present the norm distance between `Dysolve-n` for $n = 2, 3$ and 4 as a function of the number of subpixels, and for increasing number of input drives. Even with large drive amplitudes and long evolution times, we obtain an excellent approximation to the final unitary operation with relatively few subpixels. For example, using a fourth order Dyson expansion with 40 subpixels yields a norm distance error of less than 10^{-5} in all cases. Importantly, as shown in the bottom panels of Fig. 3, the computation time needed to reach this level of accuracy is only on the order of a few seconds (less than 3 s). In comparison, QuTiP's `propagator` function takes about 16 minutes to perform the calculation to an equivalent accuracy for a single input drive. Such a significant speedup proves to be particularly useful when, as discussed in section Sec. IV, the contraction stage of the algorithm needs to be repeated many times in an optimization loop. Additional benchmarking results are provided in Appendix D.

IV. APPLICATION TO OPTIMAL CONTROL

Optimal control is an essential tool in the development of high-fidelity gates for quantum computation. Most optimization algorithms, such as GRAPE [4], rely on calculating the gradient of the gate fidelity with respect to the drive amplitude at each pixel. With most approaches, this requires recalculating the evolution operator at each time interval, something which can be as expensive as the original calculation of the unitaries. Moreover, this calculation is generally performed with the presumption that the Hamiltonian is time independent over the duration of a subpixel, thus invoking a form of the RWA. In

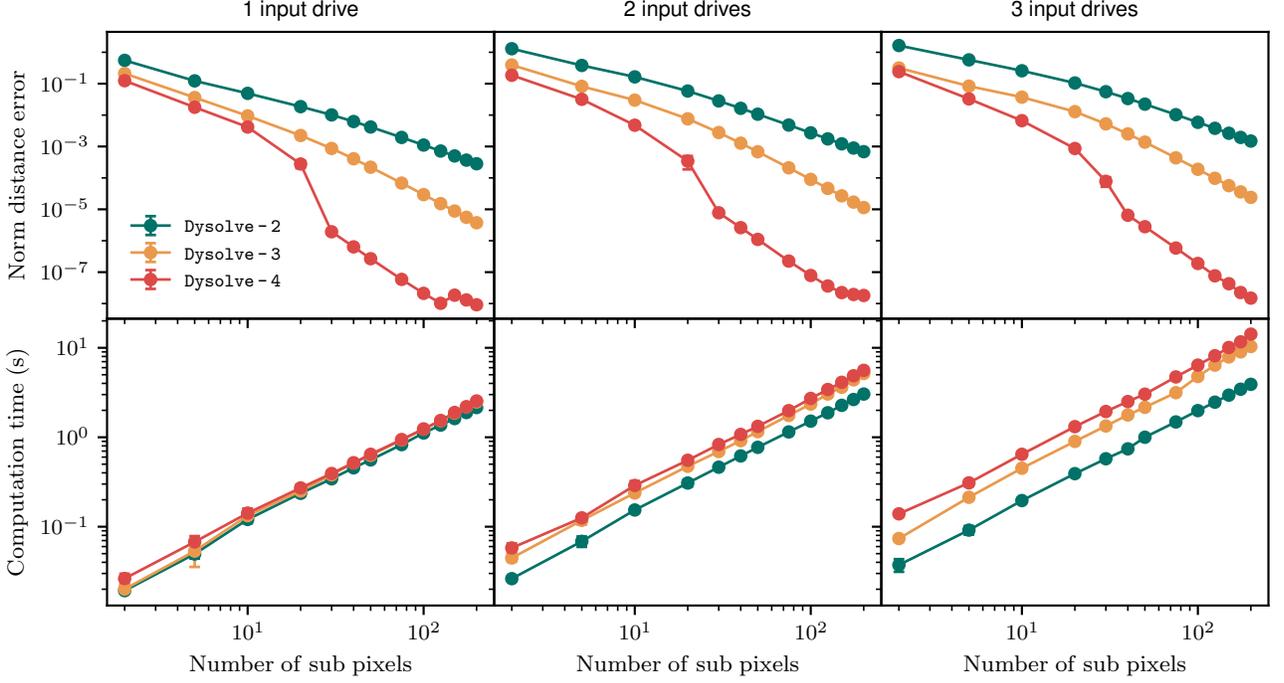


Figure 3. **Dysolve** benchmarks. First row: Frobenius norm distance from the propagator $\hat{U}_{\text{ref}}(0, T)$ for the **Dysolve** algorithm for $T = 500$ ns. Second row: Contraction time of the **Dysolve** algorithm as a function of the subpixel number N_s . The slope of the data is precisely 1, meaning that the computational time scales linearly with the number of subpixels. Preparation stage computation time: (35 ms, 1.9 s, 6.8 s) for (**Dysolve**-2, 3, 4) and 1 input drive, (0.35 s, 13 s, 87 s) for 2 input drives and (0.78 s, 47 s, 408 s) for 3 input drives.

our expansion, such an assumption is not required.

Here, we show how **Dysolve** can be applied to optimizing control pulse envelopes to maximize gate fidelity. More precisely, we consider optimizing the fidelity of an evolution $\hat{U}(T)$ with respect to a target gate unitary \hat{U}_{target} . In general, $\hat{U}(T)$ acts on the full Hilbert space of the system while \hat{U}_{target} is defined on its computational subspace. The objective is thus to maximize the performance function [8]

$$\Phi = \frac{1}{d^2} \left| \text{Tr} \left(\hat{U}_{\text{target}}^\dagger \hat{U}(T) \hat{\mathcal{P}} \right) \right|^2, \quad (30)$$

where d is the dimension of the computational subspace ($d = 2$ for a single qubit), and $\hat{\mathcal{P}}$ is the projector on that subspace. Although our approach can in principle deal with an arbitrary number of drives, for simplicity here we consider control of a single set of complex drive amplitudes u_j .

We use a GRAPE-like approach to maximize the gate fidelity which requires the gradient of the cost function Φ with respect to the drive amplitude at each pixel u_j

and subpixel s_l [8]

$$\begin{aligned} \frac{\partial \Phi}{\partial u_j} &= \sum_{l=0}^{M-1} T_{l,j} \frac{\partial \Phi}{\partial s_l}, \\ \frac{\partial \Phi}{\partial s_l} &= \frac{1}{d^2} 2 \text{Re} \left\{ \text{Tr} \left[\hat{U}_{\text{target}}^\dagger \frac{\partial \hat{U}}{\partial s_l} \mathcal{P} \right] \text{Tr} \left[\hat{U}_{\text{target}} \hat{U}^\dagger \mathcal{P} \right] \right\}. \end{aligned} \quad (31)$$

Within the framework of the **Dysolve** algorithm, it is simple to evaluate the operator $(\partial \hat{U} / \partial s_l)$. Indeed, using Eq. (20), we find that for the n -th order Dyson expansion, the derivatives of the unitaries take the form

$$\begin{aligned} \frac{\partial \hat{U}}{\partial s_l} &= \left(\prod_{m=l+1}^M \hat{U}_m \right) \frac{\partial \hat{U}_l}{\partial s_l} \left(\prod_{p=0}^{l-1} \hat{U}_p \right), \\ \frac{\partial \hat{U}_l}{\partial s_l} &= \sum_{m=0}^n \sum_{\omega_m} \exp \left(i \sum_{p=1}^m \omega_m[p] l \delta t \right) \frac{\partial \Omega_l(\omega_m)}{\partial s_l} \hat{S}^{(m)}(\omega_m, \delta t), \\ \frac{\partial \Omega_l(\omega_m)}{\partial s_l} &= \mu(\omega_m) s_l^{\mu(\omega_m)-1} s_l^{*(n-\mu(\omega_m))}. \end{aligned} \quad (32)$$

Importantly, note that the Dyson operators $\hat{S}^{(n)}(\omega_n, \delta t)$ remain unchanged by the derivative. As such we only need to perform the preparation stage of the **Dysolve** algorithm once, with the calculation of $\partial \hat{U} / \partial s_l$. Thus, the optimization iterations only require the contraction stage

computation to be performed. Further, these derivatives are exact. Consequently, the effects of off-resonant and counter-rotating terms are accounted for in the derivatives. This is the strength of the `Dysolve` algorithm for optimization.

Recall from Eq. (21) that the real and imaginary components of the drive amplitudes are associated with the magnitudes of the cosine and sine quadratures, respectively. Thus, to calculate the relevant amplitude derivatives for the cosine and sine drive envelopes, one simply calculates the appropriate sum or difference of the derivatives in Eq. (31):

$$\begin{aligned}\frac{\partial\Phi}{\partial u_{j,x}} &= \frac{1}{2} \left(\frac{\partial\Phi}{\partial u_j} + \frac{\partial\Phi}{\partial u_j^*} \right), \\ \frac{\partial\Phi}{\partial u_{j,y}} &= \frac{-i}{2} \left(\frac{\partial\Phi}{\partial u_j} - \frac{\partial\Phi}{\partial u_j^*} \right),\end{aligned}\quad (33)$$

where $u_j = u_{j,x} + iu_{j,y}$. To perform the GRAPE algorithm, the set of drive amplitudes are simply updated by taking steps in the direction of the gradient of the fidelity [4]

$$u_{j,(x,y)} \rightarrow u_{j,(x,y)} + \epsilon \frac{\partial\Phi}{\partial u_{j,(x,y)}}, \quad (34)$$

where ϵ is a small, positive number which need not be fixed during the optimization process. For example, one could use a backtracking line search to maximize the gain in fidelity [11, 12]. In the examples presented below, we simply use a sufficiently small ϵ for the updating process to be stable.

A. Example: Cross-Resonance Gate for Coupled Transmons

As an example of application of our implementation of the GRAPE algorithm with `Dysolve`, we present results of the optimization of a two-qubit gate. To this end, we consider the system of two fixed-frequency transmon qubits, illustrated in Fig. 4. Each qubit is described by a Hamiltonian of the form [10]

$$\hat{H}_j = 4E_{Cj}\hat{n}_j^2 - E_{Jj}\cos\hat{\varphi}_j, \quad (35)$$

where \hat{n}_j and $\hat{\varphi}_j$ with $j = c, t$ are the conjugate charge and phase operators of the transmon, while E_{Cj} and E_{Jj} are the charging and Josephson energies. More precisely, we consider the cross-resonance gate, which performs an X rotation on a target qubit conditional on the state of a control qubit which is driven at the target qubit's frequency [13, 14]. Optimal control has been applied to this gate before in Refs. [15, 16]. However, these approaches have made use of simplified models for the device and of the rotating wave approximation. Taking advantage of the `Dysolve` algorithm, our approach generalizes previous work on the cross-resonance gate by considering

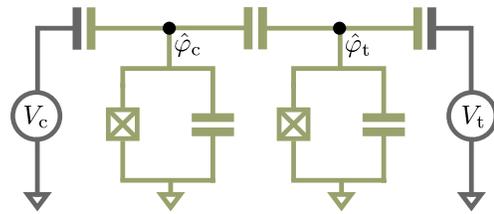


Figure 4. Superconducting circuit for cross-resonance gates between transmon qubits. The leftmost transmon, of frequency $\omega_c/2\pi$, plays the role of the control qubit and is strongly driven by the voltage source V_c at the frequency $\omega_t/2\pi$ of the target qubit (rightmost transmon). The target qubit is also driven by V_t using a relatively weak tone that serves to give additional control. $\hat{\varphi}_c$ and $\hat{\varphi}_t$ correspond to the phase degree of freedom associated with the control and target qubits, respectively.

the full circuit Hamiltonian and including all counter-rotating terms.

Following the standard circuit-quantization procedure [17], the two-transmon Hamiltonian can be put in the form $\hat{H}(t) = \hat{H}_0 + \hat{V}(t)$ with

$$\begin{aligned}\hat{H}_0 &= \hat{H}_c + \hat{H}_t + \hbar g \hat{n}_c \hat{n}_t, \\ \hat{V}(t) &= \Omega_c(t) \hat{n}_c + \Omega_t(t) \hat{n}_t,\end{aligned}\quad (36)$$

where \hat{H}_c, \hat{H}_t are the Hamiltonian of the control and target qubits, respectively, and $\hbar g \hat{n}_c \hat{n}_t$ results from the capacitive interaction between the qubits. The amplitudes $\Omega_c(t), \Omega_t(t)$ are the time-dependent drives

$$\begin{aligned}\Omega_c(t) &= [\Omega_{cx}(t) \sin(\omega_t t) + \Omega_{cy}(t) \cos(\omega_t t)], \\ \Omega_t(t) &= [\Omega_{tx}(t) \sin(\omega_t t) + \Omega_{ty}(t) \cos(\omega_t t)],\end{aligned}\quad (37)$$

with $\{\Omega_{cx}, \Omega_{cy}, \Omega_{tx}, \Omega_{ty}\}$ the drive envelope functions to be optimized by GRAPE. While activating the cross-resonance gate only requires driving the control qubit at the frequency of the target, the additional control over the target qubit $\hat{\Omega}_t(t)$ is useful to eliminate single qubit rotations and obtain higher fidelities to the target unitary [18, 19].

We choose to operate this gate with control and target frequencies of $\omega_c/2\pi = 5.1$ GHz, $\omega_t/2\pi = 4.9$ GHz, and anharmonicities $\alpha_c/2\pi = -355$ MHz and $\alpha_t/2\pi = -352$ MHz, respectively. Moreover, the qubit-qubit coupling is set to $g/2\pi = 4.29$ MHz and the target unitary is taken to be ZX_{90} gate [20, 21]. Figure 5 shows the infidelity of the cross-resonance gate as a function of the gate time. We find that for a flat pulse a gate fidelity of 98.5% in a 300 ns gate time is possible (red symbols), while values approaching 99% can be reached when correcting for additional single-qubit rotations (orange symbols). In contrast, GRAPE reaches significantly higher fidelities, going beyond 99.99% in a shorter gate time, even when accounting for all off-resonant and counter rotating terms (blue symbols).

To further demonstrate that GRAPE under `Dysolve` can successfully optimize the cross-resonance gate more

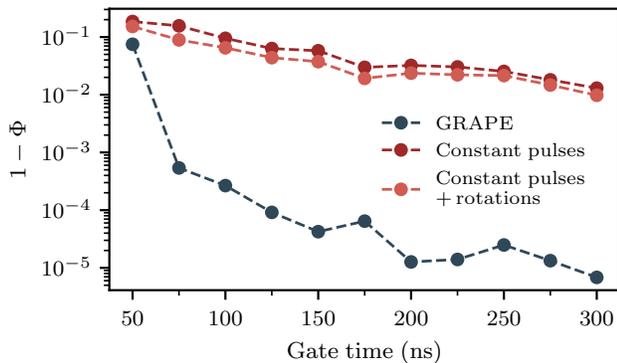


Figure 5. Infidelity of the CR gate as a function of the chosen gate time when the gate is operated at the qubit-qubit detuning $\Delta/2\pi = (\omega_c - \omega_t)/2\pi = 210$ MHz. The red curve indicates the best obtainable infidelity from a constant amplitude pulse, with the orange curve reporting the infidelity up to arbitrary single qubit rotations on both qubits, applied before and after the cross-resonance gate channel. The blue curve reports the infidelity when optimized under the GRAPE algorithm until convergence.

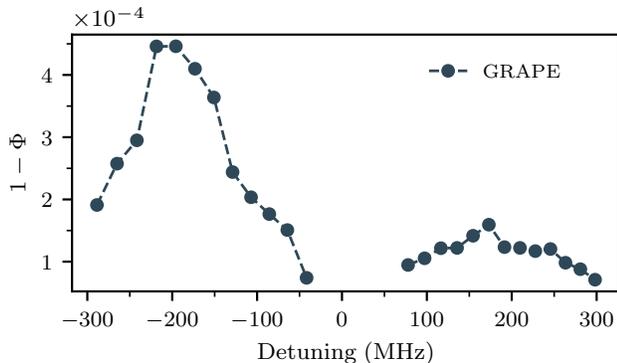


Figure 6. Infidelity of the CR gate as a function of the detuning of the qubits when optimized under GRAPE for 5000 iterations. The control qubit frequency is fixed to $\omega_c/2\pi = 5.10$ GHz.

generally, Fig. 6 shows the fidelity of an optimized 300-ns gate as a function of the qubit-qubit detuning $\Delta/2\pi = (\omega_c - \omega_t)/2\pi$. In the numerical simulations, this detuning is varied by changing the target qubit frequency whilst keeping the qubit-qubit coupling and the control-qubit frequency fixed. We note that the performance of the gate at positive detunings is approximately in line with those predicted by Schrieffer-Wolff perturbation theory for an echoed-cross resonance gate performed on a similar device [22]. The results are excluded at $\Delta = 0, \pm\alpha$, as the

qubits strongly hybridized due to resonances. The slight variations in the GRAPE fidelity in the wide detuning range are partially attributable to variations in higher-order two-qubit coupling amplitudes across, alongside the performance of gradient ascent for different optimization landscapes. This also constitutes evidence for the robustness of our algorithm, as excellent fidelities are obtained in a broad range of parameters.

V. FUTURE WORK AND CONCLUSION

We have demonstrated that the `Dysolve` algorithm provides a means to quickly and accurately simulate driven systems whilst accounting for all of the effects of the counter-rotating and off-resonant terms. Analysis of ultrafast quantum gates and the quantum speed limit [23], where counter-rotating effects are particularly important, would also be possible with our algorithm. Additionally, this method trivializes the calculation of the gradient, allowing for rapid optimization without the need for additional approximations, and can be modified to include dissipative effects. Indeed, a simple extension of the optimization scheme would allow for optimization of lossy quantum systems specifically, through an open GRAPE-like scheme [24], or the simulation of larger lossy systems through the use of trajectories. We also note that as the expressions depend explicitly on the drive amplitudes, second derivatives and the Hessian matrix can also be calculated exactly, opening the door to second order optimizers such as Newton's method.

We anticipate that future iterations of the `Dysolve` algorithm improve the efficiency of both the preparation and contraction stages of the algorithm. Given the parallelized nature of this solver, we also envision a direct extension to graphics processing units (GPU's), which could allow for fast simulation of significantly larger systems. We leave this for future work.

At the time of publication, the authors were made aware of a recent paper [25] which uses a different approach to obtain a result similar to that which we provide in Sec. II. Our derived weighting functions are equivalent to divided differences [26] utilised in their method.

ACKNOWLEDGMENTS

We thank Catherine Leroux for discussions and Éric Giguère for assistance with numerical calculations. This work was undertaken thanks in part to funding from NSERC, the Canada First Research Excellence Fund, and the U.S. Army Research Office Grant No. W911NF-18-1-0411.

[1] J. C. Tremblay and T. Carrington, *The Journal of Chemical Physics* **121**, 11535 (2004),

<https://doi.org/10.1063/1.1814103>.

- [2] C. Moler and C. Loan, SIAMREV **20**, 801 (1978).
- [3] F. Bloch and A. Siegert, Phys. Rev. **57**, 522 (1940).
- [4] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, Journal of Magnetic Resonance **172**, 296 (2005).
- [5] E. Barnes and S. Das Sarma, Phys. Rev. Lett. **109**, 060401 (2012).
- [6] Y. Wu and X. Yang, Phys. Rev. Lett. **98**, 013601 (2007).
- [7] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm, Phys. Rev. Lett. **103**, 110501 (2009).
- [8] F. Motzoi, J. M. Gambetta, S. T. Merkel, and F. K. Wilhelm, Phys. Rev. A **84**, 022307 (2011).
- [9] J. Johansson, P. Nation, and F. Nori, Computer Physics Communications **184**, 1234 (2013).
- [10] J. Koch, T. M. Yu, J. Gambetta, A. A. Houck, D. I. Schuster, J. Majer, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf, Phys. Rev. A **76**, 042319 (2007).
- [11] L. Armijo, Pacific J. Math. **16**, 1 (1966).
- [12] P. A. Absil, R. Mahony, and B. Andrews, SIAM Journal on Optimization **16**, 531 (2005), <https://doi.org/10.1137/040605266>.
- [13] C. Rigetti and M. Devoret, Phys. Rev. B **81**, 134507 (2010).
- [14] J. M. Chow, A. Córcoles, J. M. Gambetta, C. Rigetti, B. Johnson, J. A. Smolin, J. Rozen, G. A. Keefe, M. B. Rothwell, M. B. Ketchen, *et al.*, Physical review letters **107**, 080502 (2011).
- [15] J. L. Allen, R. Kosut, J. Joo, P. Leek, and E. Ginossar, Physical Review A **95**, 042325 (2017).
- [16] S. Kirchhoff, T. Keßler, P. J. Liebermann, E. Assémat, S. Machnes, F. Motzoi, and F. K. Wilhelm, Physical Review A **97**, 042348 (2018).
- [17] M. H. Devoret *et al.*, Les Houches, Session LXIII **7** (1995).
- [18] S. Sheldon, E. Magesan, J. M. Chow, and J. M. Gambetta, Physical Review A **93**, 060302 (2016).
- [19] A. Patterson, J. Rahamim, T. Tsunoda, P. Spring, S. Jebari, K. Ratter, M. Mergenthaler, G. Tancredi, B. Vlastakis, M. Esposito, *et al.*, Physical Review Applied **12**, 064013 (2019).
- [20] J. M. Chow, J. M. Gambetta, A. D. Corcoles, S. T. Merkel, J. A. Smolin, C. Rigetti, S. Poletto, G. A. Keefe, M. B. Rothwell, J. R. Rozen, *et al.*, Physical review letters **109**, 060501 (2012).
- [21] A. D. Córcoles, J. M. Gambetta, J. M. Chow, J. A. Smolin, M. Ware, J. Strand, B. L. Plourde, and M. Steffen, Physical Review A **87**, 030301 (2013).
- [22] M. Malekakhlagh, E. Magesan, and D. C. McKay, Phys. Rev. A **102**, 042605 (2020).
- [23] Y. Shao, B. Liu, M. Zhang, H. Yuan, and J. Liu, Physical Review Research **2**, 10.1103/physrevresearch.2.023299 (2020).
- [24] M. Abdelhafez, D. I. Schuster, and J. Koch, Phys. Rev. A **99**, 052327 (2019).
- [25] A. Kalev and I. Hen, An integral-free representation of the dyson series using divided differences (2020), arXiv:2010.09888 [quant-ph].
- [26] C. de Boor, Surveys in Approximation Theory **1**, 46–69 (2005).

Appendix A: Derivation of n-th order weighting functions

In this Appendix, we derive the n -th order weighting function and the form of the Dyson series operators. We begin by defining a frequency-dependent set of path operators

$$\hat{R}_{\{m_n, \dots, m_0\}}(\mathbf{v}_n, \delta t) = (-i)^n \int_0^{\delta t} dt_M \cdots \int_0^{t_2} dt_1 (-i(\hat{H}_0 - \mathbf{v}_n[n]))^{m_n} \hat{X} \cdots (-i(\hat{H}_0 - \mathbf{v}_n[1]))^{m_1} \hat{X} (-i(\hat{H}_0 - i\mathbf{v}_n[0]))^{m_0}, \quad (\text{A1})$$

where, for now, \mathbf{v}_n is an arbitrary vector and $M = \sum_{i=0}^n m_i + n$. Since there is no explicit time dependence in Eq. (A1), we can evaluate its M integrals simultaneously,

$$\begin{aligned} \hat{R}_{\{m_n, \dots, m_0\}}(\mathbf{v}_n, \delta t) &= \frac{(-i)^n (-i(\hat{H}_0 - \mathbf{v}_n[n]))^{m_n} \hat{X} \cdots (-i(\hat{H}_0 - \mathbf{v}_n[1]))^{m_1} \hat{X} (-i(\hat{H}_0 - \mathbf{v}_n[0]))^{m_0} (\delta t)^M}{M!} \\ &= (-i\delta t)^n \frac{(-i\delta t(\hat{H}_0 - \mathbf{v}_n[n]))^{m_n} \hat{X} \cdots (-i\delta t(\hat{H}_0 - \mathbf{v}_n[0]))^{m_0}}{M!}. \end{aligned} \quad (\text{A2})$$

Inserting identity operators $I_N = \sum_k |k\rangle\langle k|$, with $\hat{H}_0|k\rangle = \lambda_k|k\rangle$ and N the size of the system's Hilbert space, between groups of \hat{H}_0 and each \hat{X} operator leads to

$$\begin{aligned} \hat{R}_{\{m_n, \dots, m_0\}}(\mathbf{v}_n, \delta t) &= (-i\delta t)^n \sum_{\mathbf{k}_n} \langle k^{(n)} | \hat{X} | k^{(n-1)} \rangle \langle k^{(n-1)} | \hat{X} \cdots | k^{(1)} \rangle \langle k^{(1)} | \hat{X} | k^{(0)} \rangle | k^{(n)} \rangle \langle k^{(0)} | \\ &\quad \times \frac{(-i(\boldsymbol{\lambda}_n(\mathbf{k}_n) - \mathbf{v}_n)[0] \delta t)^{m_0} \cdots (-i(\boldsymbol{\lambda}_n(\mathbf{k}_n) - \mathbf{v}_n)[n] \delta t)^{m_n}}{M!}. \end{aligned} \quad (\text{A3})$$

To obtain the Dyson operator, we sum Eq. (A3) over all $\mathbf{m} \in \mathbf{Z}_+^{n+1}$. To this end, we define the n -th order weighting function $f(\boldsymbol{\lambda}_n)$ as

$$\begin{aligned}
f(\boldsymbol{\lambda}_n) &= \sum_{m_0, \dots, m_n} \frac{(-i\boldsymbol{\lambda}_n[0])^{m_0} \dots (-i\boldsymbol{\lambda}_n[n])^{m_n}}{(\sum_{z=0}^n (m_z) + n)!} \\
&= \sum_{m_0, \dots, m_{n-2}} (-i\boldsymbol{\lambda}_n[0])^{m_0} \dots (-i\boldsymbol{\lambda}_n[n-2])^{m_{n-2}} \sum_{P=0}^{\infty} \sum_{p=0}^P \frac{(-i\boldsymbol{\lambda}_n[n-1])^{P-p} (-i\boldsymbol{\lambda}_n[n])^p}{(\sum_{n=0}^{n-2} (m_n) + P - p + p + n)!} \\
&= \sum_{m_0, \dots, m_{n-2}, P} \frac{(-i\boldsymbol{\lambda}_n[0])^{m_0} \dots (-i\boldsymbol{\lambda}_n[n-2])^{m_{n-2}} ((-i\boldsymbol{\lambda}_n[n-1])^{P+1} - (-i\boldsymbol{\lambda}_n[n])^{P+1})}{(\sum_{n=0}^{n-2} (m_n) + P + 1 + n - 1)! (-i\boldsymbol{\lambda}_n[n-1] + i\boldsymbol{\lambda}_n[n])} \\
&= i \left[\frac{f(\mathbf{g}(\boldsymbol{\lambda}_n)) - f(\mathbf{g}^2(\boldsymbol{\lambda}_n) \cup \boldsymbol{\lambda}_n[n])}{\boldsymbol{\lambda}_n[n-1] - \boldsymbol{\lambda}_n[n]} \right],
\end{aligned} \tag{A4}$$

where $f(\boldsymbol{\lambda}_0) = \exp(-i\boldsymbol{\lambda}_0[0])$. This gives us the form of $\hat{R}^{(n)}(\mathbf{v}_n, 0, \delta t)$:

$$\hat{R}^{(n)}(\mathbf{v}_n, \delta t) = \left(\frac{-i\delta t}{2} \right)^n \sum_{\mathbf{k}_n} \langle k^{(n)} | \hat{X} | k^{(n-1)} \rangle \langle k^{(n-1)} | \hat{X} \dots | k^{(1)} \rangle \langle k^{(1)} | \hat{X} | k^{(0)} \rangle | k^{(n)} \rangle \langle k^{(0)} | \times f((\boldsymbol{\lambda}_n(\mathbf{k}_n) - \mathbf{v}_n) \delta t). \tag{A5}$$

We then note that

$$e^{i\omega t_0} f(\boldsymbol{\lambda}_n) = f(\boldsymbol{\lambda}_n - \omega t_0). \tag{A6}$$

To prove this, first note that this is trivially true for $f(\boldsymbol{\lambda}_0)$. We then make the inductive step, assuming Eq. (A6) to be true for $n-1$. Then,

$$\begin{aligned}
e^{i\omega t_0} f(\boldsymbol{\lambda}_n) &= i \left[\frac{e^{i\omega t_0} f(\mathbf{g}(\boldsymbol{\lambda}_n)) - e^{i\omega t_0} f(\mathbf{g}^2(\boldsymbol{\lambda}_n) \cup \boldsymbol{\lambda}_n[n])}{\boldsymbol{\lambda}_n[n-1] - \boldsymbol{\lambda}_n[n]} \right] \\
&= i \left[\frac{f(\mathbf{g}(\boldsymbol{\lambda}_n) - \omega_0 t) f(\mathbf{g}^2(\boldsymbol{\lambda}_n) \cup \boldsymbol{\lambda}_n[n] - \omega_0 t)}{(\boldsymbol{\lambda}_n[n-1] - \omega_0 t) - (\boldsymbol{\lambda}_n[n] - \omega_0 t)} \right] \\
&= f(\boldsymbol{\lambda}_n - \omega_0 t).
\end{aligned} \tag{A7}$$

By linearity, this implies that $e^{ia\delta t} \hat{R}^{(n)}(\mathbf{v}_n, \delta t) = \hat{R}^{(n)}(\mathbf{v}_n + a, \delta t)$.

We now consider the effect of a set of oscillatory terms. To this end, we first note that the Dyson path operator in Eq. (7), as well as the Dyson operators themselves, can be defined recursively

$$\hat{S}_{[m_n, \dots, m_0]}^{(n)}(\boldsymbol{\omega}_n, \delta t) = (-i\hat{H}_0)^{m_n} \hat{X} \int_0^{\delta t} dt_M \dots \int_0^{t_{M-m_n+1}} dt_{M-m_n} e^{i\boldsymbol{\omega}_n[n]t_{M-m_n}} \hat{S}_{[m_{n-1}, \dots, m_0]}^{(n-1)}(\mathbf{g}(\boldsymbol{\omega}_n), t_{M-m_n}), \tag{A8}$$

$$\hat{S}^{(n)}(\boldsymbol{\omega}_n, \delta t) = \frac{1}{2} \sum_{m_n \in \mathbf{Z}_+} (-i\hat{H}_0)^{m_n} \hat{X} \int_0^{\delta t} dt_M \dots \int_0^{t_{M-m_n+1}} dt_{M-m_n} e^{i\boldsymbol{\omega}_n[n]t_{M-m_n}} \hat{S}^{(n-1)}(\mathbf{g}(\boldsymbol{\omega}_n), t_{M-m_n}), \tag{A9}$$

$$\hat{R}^{(n)}(\mathbf{v}_n, \delta t) = \frac{1}{2} \sum_{m_n \in \mathbf{Z}_+} (-i\hat{H}_0 - i\mathbf{v}_n[n])^{m_n} \hat{X} \int_0^{\delta t} dt_M \dots \int_0^{t_{M-m_n+1}} dt_{M-m_n} \hat{R}^{(n-1)}(\mathbf{g}(\mathbf{v}_n), t_{M-m_n}), \tag{A10}$$

We now wish to demonstrate that $\hat{R}^{(n)}(\mathbf{c}(\boldsymbol{\omega}_n), \delta t) = \hat{S}^{(n)}(\boldsymbol{\omega}_n, \delta t)$, where $\mathbf{c}(\boldsymbol{\omega}_n)$ is the cumulative vector introduced in

Eq. (16). To first order,

$$\begin{aligned}
\hat{S}^{(1)}(\boldsymbol{\omega}_1, \delta t) &= \frac{1}{2} \sum_{m_1 \in \mathbf{Z}_+} (-i\hat{H}_0)^{m_1} \hat{X} \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m_1+1}} dt_{M-m_1} e^{i\boldsymbol{\omega}_1[1]t_{M-m_1}} \hat{S}^{(0)}(\mathbf{0}, t_{M-m_1}) \\
&= \frac{1}{2} \sum_{m_1 \in \mathbf{Z}_+} (-i\hat{H}_0)^{m_1} \hat{X} \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m_1+1}} dt_{M-m_1} \left[\sum_i |i\rangle \langle i| e^{i\boldsymbol{\omega}_1[1]t_{M-m_1}} f(\lambda_i t_{M-m_1}) \right] \\
&= \frac{1}{2} \sum_{m_1 \in \mathbf{Z}_+} (-i\hat{H}_0)^{m_1} \hat{X} \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m_1+1}} dt_{M-m_1} \left[\sum_i |i\rangle \langle i| f((\lambda_i - \boldsymbol{\omega}_1[1])t_{M-m_1}) \right], \\
&= \frac{1}{2} \sum_{m_1 \in \mathbf{Z}_+} (-i\hat{H}_0)^{m_1} \hat{X} \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m_1+1}} dt_{M-m_1} \hat{R}^{(0)}([\boldsymbol{\omega}_1[1]], t_{M-m_1}), \\
&= \frac{1}{2} \sum_{m_1 \in \mathbf{Z}_+} (-i(\hat{H}_0 - 0))^{m_1} \hat{X} \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m_1+1}} dt_{M-m_1} \hat{R}^{(0)}(\mathbf{g}([\boldsymbol{\omega}_1[1], 0]), t_{M-m_1}), \\
&= \hat{R}^{(1)}([\boldsymbol{\omega}_1[1], 0], \delta t) \\
&= \hat{R}^{(1)}(\mathbf{c}(\boldsymbol{\omega}_1), \delta t).
\end{aligned} \tag{A11}$$

We again make the inductive step with the assumption that $\hat{R}^{(n-1)}(\mathbf{c}(\boldsymbol{\omega}_n), \delta t) = \hat{S}^{(n-1)}(\boldsymbol{\omega}_n, \delta t)$. To proceed, we first show the following result:

$$\begin{aligned}
\mathbf{c}(\mathbf{g}(\mathbf{v}_n)) &= \left(\left[\sum_{p=1}^{n-1} \mathbf{v}_n[n-p] \right], \left[\sum_{p=1}^{n-2} \mathbf{v}_n[n-p] \right], \dots, \mathbf{v}_n[n-1], 0 \right). \\
&= \left(\left[\sum_{p=0}^{n-1} \mathbf{v}_n[n-p] \right], \left[\sum_{p=0}^{n-2} \mathbf{v}_n[n-p] \right], \dots, \mathbf{v}_n[n-1] + \mathbf{v}_n[n], \mathbf{v}_n[n] \right) - \mathbf{v}_n[n] \\
&= \mathbf{g} \left(\left[\sum_{p=0}^{n-1} \mathbf{v}_n[n-p] \right], \left[\sum_{p=0}^{n-2} \mathbf{v}_n[n-p] \right], \dots, \mathbf{v}_n[n-1] + \mathbf{v}_n[n], \mathbf{v}_n[n], 0 \right) - \mathbf{v}_n[n] \\
&= \mathbf{g}(\mathbf{c}(\mathbf{v}_n)) - \mathbf{v}_n[n]
\end{aligned} \tag{A12}$$

Then,

$$\begin{aligned}
\hat{S}^{(n)}(\boldsymbol{\omega}_n, \delta t) &= \frac{1}{2} \sum_{m_n \in \mathbf{Z}_+} (-i\hat{H}_0)^{m_n} \hat{X} \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m_n+1}} dt_{M-m_n} e^{i\boldsymbol{\omega}_n[n]t_{M-m_n}} \hat{S}^{(n-1)}(\mathbf{g}(\boldsymbol{\omega}_n), t_{M-m_n}) \\
&= \frac{1}{2} \sum_{m_n \in \mathbf{Z}_+} (-i\hat{H}_0)^{m_n} \hat{X} \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m_n+1}} dt_{M-m_n} e^{i\boldsymbol{\omega}_n[n]t_{M-m_n}} \hat{R}^{(n-1)}(\mathbf{c}(\mathbf{g}(\boldsymbol{\omega}_n)), t_{M-m_n}) \\
&= \frac{1}{2} \sum_{m_n \in \mathbf{Z}_+} (-i\hat{H}_0)^{m_n} \hat{X} \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m_n+1}} dt_{M-m_n} \hat{R}^{(n-1)}(\mathbf{c}(\mathbf{g}(\boldsymbol{\omega}_n)) + \boldsymbol{\omega}_n[n], t_{M-m_n}) \\
&= \frac{1}{2} \sum_{m_n \in \mathbf{Z}_+} (-i\hat{H}_0 - i0)^{m_n} \hat{X} \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m_n+1}} dt_{M-m_n} \hat{R}^{(n-1)}(\mathbf{g}(\mathbf{c}(\boldsymbol{\omega}_n)), t_{M-m_n}) \\
&= \hat{R}^{(n)}(\mathbf{c}(\boldsymbol{\omega}_n), \delta t).
\end{aligned} \tag{A13}$$

finishing the induction and giving the final desired form in Eq. (14).

Appendix B: Multiple Drive Inputs

Suppose there are q -independent drive inputs each with own drive frequency and operator. For an n -th order Dyson expansion, we define an n -dimensional vector $\boldsymbol{\beta}$ with values ranging from 1 to q , each value referring to one of the

drive inputs. To simplify the notation, we drop all subscripts n , which are implied. The new Dyson series operators corresponding to a particular vector $\boldsymbol{\beta}$ is

$$\hat{S}_{\boldsymbol{\beta}}^{(n)}(\boldsymbol{\omega}_{\boldsymbol{\beta}}, \delta t) = \sum_{\mathbf{k}_n} \langle k^{(n)} | \hat{X}_{\boldsymbol{\beta}[n]} | k^{(n-1)} \rangle \langle k^{(n-1)} | \dots \langle k^{(1)} | \hat{X}_{\boldsymbol{\beta}[1]} | k^{(0)} \rangle | k^{(n)} \rangle \langle k^{(0)} | f(\boldsymbol{\lambda} - \delta t \mathbf{v}_{\boldsymbol{\beta}}), \quad (\text{B1})$$

with $\boldsymbol{\omega}_{\boldsymbol{\beta}}$ an n -vector where the p -th element of this vector is $\pm \omega_{\boldsymbol{\beta}[p]}$, or more simply, plus or minus the drive frequency corresponding to the $\boldsymbol{\beta}[p]$ -th input. Similarly, $\hat{X}_{\boldsymbol{\beta}[p]}$ refers to the $\boldsymbol{\beta}[p]$ -th input operator. The definition of $\mathbf{v}_{\boldsymbol{\beta}}$ remains unchanged from Eq. (16), albeit with the new drive frequency vector $\boldsymbol{\omega}_{\boldsymbol{\beta}}$. The drive function is also modified and now reads

$$\Omega(\boldsymbol{\omega}_{\boldsymbol{\beta}}) = \prod_{p=1}^n \Omega_{\boldsymbol{\beta}[p]}^{\mu} \Omega_{\boldsymbol{\beta}[p]}^{*(1-\mu)}, \quad (\text{B2})$$

where

$$\mu = \frac{1}{2} (1 + \text{sign}\{\omega_{\boldsymbol{\beta}[p]}\}), \quad (\text{B3})$$

thus allowing us to define our generic n -th order Dyson series,

$$\hat{U}^{(n)}(t, t + \delta t) = \sum_{\boldsymbol{\beta}} \sum_{\boldsymbol{\omega}_{\boldsymbol{\beta}}} \exp\left(i \sum_{p=1}^n \omega_{\boldsymbol{\beta}[p]} t\right) \Omega(\boldsymbol{\omega}_{\boldsymbol{\beta}}) \hat{S}_{\boldsymbol{\beta}}^{(n)}(\boldsymbol{\omega}_{\boldsymbol{\beta}}, \delta t), \quad (\text{B4})$$

which reads as a version of Eq. (22) with multiple drive operators and frequencies.

Appendix C: Linear interpolation of subpixels

The accuracy of the Dyson expansion is insensitive to the drive frequency ω – rather, it depends only on the number of subpixels and the drive amplitudes. However, if each subpixel assumes a constant amplitude, a leading error can be caused by the change in amplitude of the drive over a single subpixel. In Fig. 7, we illustrate this issue, where the subpixel amplitudes (black bars) will over or underestimate the true pulse amplitude, depending on the gradient of the envelope function. To circumvent this issue, we consider a new linear interpolation of the drive envelope over a single subpixel. We define $y_l(t)$ as the new time dependent amplitude for the l -th subpixel:

$$y_l(t) = s'_l + \frac{s_{l+1} - s_l}{\delta t} (t - l\delta t), \quad l\delta t < t < (l+1)\delta t, \quad (\text{C1})$$

where s'_l is calculated from a modified filter function to ensure that the integral of the subpixel matches that of the continuous pulse. The $y_l(t)$ time-dependent subpixels are shown in orange, and well approximate the true pulse even for relatively few subpixels.

To determine the impact of the linear time term t_0 on the time ordered integral, we consider the $(n-j)$ -th iteration of a Dyson Series operator,

$$\begin{aligned} & \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m} - t_{M-m_j}} dt_{M-m_j} \exp(i\boldsymbol{\omega}_n[j]t_{M-m_j}) \hat{S}^{(j-1)}(\mathbf{g}^{(j-1)}(\boldsymbol{\omega}_n), t_{M-m_j}), \\ &= \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m} - t_{M-m_j}} dt_{M-m_j} (-i\partial_{\boldsymbol{\omega}_n, j}) \exp(i\boldsymbol{\omega}_n[j]t_{M-m_j}) \hat{S}^{(j-1)}(\mathbf{g}^{(j-1)}(\boldsymbol{\omega}_n), t_{M-m_j}), \\ &= (-i\partial_{\boldsymbol{\omega}_n, j}) \int_0^{\delta t} dt_M \cdots \int_0^{t_{M-m} - t_{M-m_j}} dt_{M-m_j} \exp(i\boldsymbol{\omega}_n[j]t_{M-m_j}) \hat{S}^{(j-1)}(\mathbf{g}^{(j-1)}(\boldsymbol{\omega}_n), t_{M-m_j}), \\ &= (-i\partial_{\boldsymbol{\omega}_n, j}) \hat{S}^{(j)}(\mathbf{g}^{(j)}(\boldsymbol{\omega}_n), t_{M-m_{j+1}}), \end{aligned} \quad (\text{C2})$$

where we define $\partial_{\boldsymbol{\omega}_n, j}$ as the derivative with respect to the j -th component of the vector $\boldsymbol{\omega}_n$. For example

$$\partial_{\boldsymbol{\omega}_n, j}(\boldsymbol{\omega}_n) = [0, 0, \dots, \underbrace{1}_{j\text{-th}}, \dots, 0]. \quad (\text{C3})$$

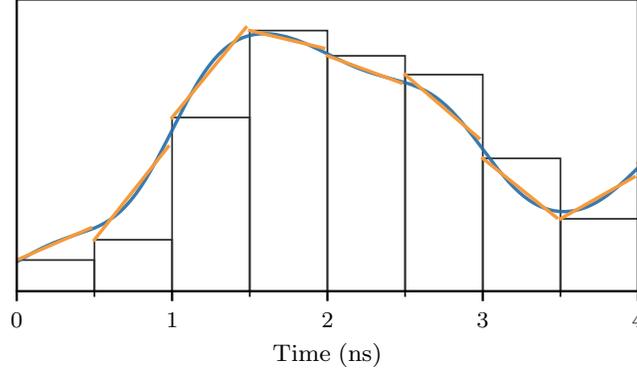


Figure 7. Linear interpolation of the pulse sequences (yellow) with two subpixels per pixel. The black bars indicate the original Gaussian filtering, with the blue line the ‘true’ filtered pulse.

This result allows us to modify Eq. (29) to the following result, replacing the drive amplitude function by a drive amplitude operator,

$$\hat{\Omega}_l(\omega_n) = \prod_{p=1}^n (s_l - is'_l \partial_{\omega_{n,p}})^\mu (s_l^* + is_l'^* \partial_{\omega_{n,p}})^{1-\mu}, \quad \text{where } \mu = \frac{1}{2}(1 + \text{sign}\{\omega[p]\}), \quad (\text{C4})$$

where the derivatives are to act upon the Dyson series operators. It is important to note that $s'_l \ll s_l$ in the majority of cases – as such, it is possible to consider a partial truncation of the series, where only up to a certain power of derivative terms s'_l are included in the set of Dyson series operators.

To make use of the drive amplitude function, we must determine the derivatives of the weighting function $f(\lambda_n)$. We begin with the base case with $\lambda_0 = [\lambda_0]$,

$$\begin{aligned} \frac{d}{d\lambda_0} f(\lambda_0) &= \frac{d}{d\lambda_0} e^{-i\lambda_0} \\ &= -ie^{-i\lambda_0} \\ &= \lim_{\epsilon \rightarrow 0} \frac{e^{-i(\lambda_0+\epsilon)} - e^{-i\lambda_0}}{\epsilon} \\ &= if(\lambda_0 \cup \lambda_0). \end{aligned} \quad (\text{C5})$$

Now, we assume that $\partial_{\lambda_{n,j}} f(\lambda_n) = if(\lambda_n[j] \cup \lambda_n)$. Then,

$$\begin{aligned} \partial_{\lambda_{n,j}} f(\lambda_n) &= i\partial_{\lambda_{n,j}} \frac{f(g(\lambda_n)) - f(g^2(\lambda_n) \cup \lambda_n[n])}{\lambda_n[n-1] - \lambda_n[n]} \\ &= \frac{i}{\lambda_n[n-1] - \lambda_n[n]} [\partial_{\lambda_{n,j}} f(g(\lambda_n)) - \partial_{\lambda_{n,j}} f(g^2(\lambda_n) \cup \lambda_n[n])] \\ &= \frac{i}{\lambda_n[n-1] - \lambda_n[n]} [if(g(\lambda_n[j] \cup \lambda_n)) - if(g^2(\lambda_n[j] \cup \lambda_n) \cup \lambda_n[n])] \\ &= if(\lambda_n[j] \cup \lambda_n). \end{aligned} \quad (\text{C6})$$

Appendix D: Benchmark calculations

In order to calculate the Frobenius norm error used as a metric in our benchmarks, we required an excellent approximation to the reference propagator operator $\hat{U}_{\text{ref}}(0, T)$. We first considered an alternative numerical solver, `propagator` from the `python` package `QuTiP`. We selected highly accurate settings for this computation, namely absolute and relative tolerances of 10^{-16} , 2 different numerical methods with their maximal order (12 for `adams` and 5 for `bdf`), discretizing each pixel into 10^4 subpixels and allowing for 10^{17} internal sub-steps. After averaging over many simulations of $T = 500$ ns with a single drive, it became apparent that the `propagator` function was converging

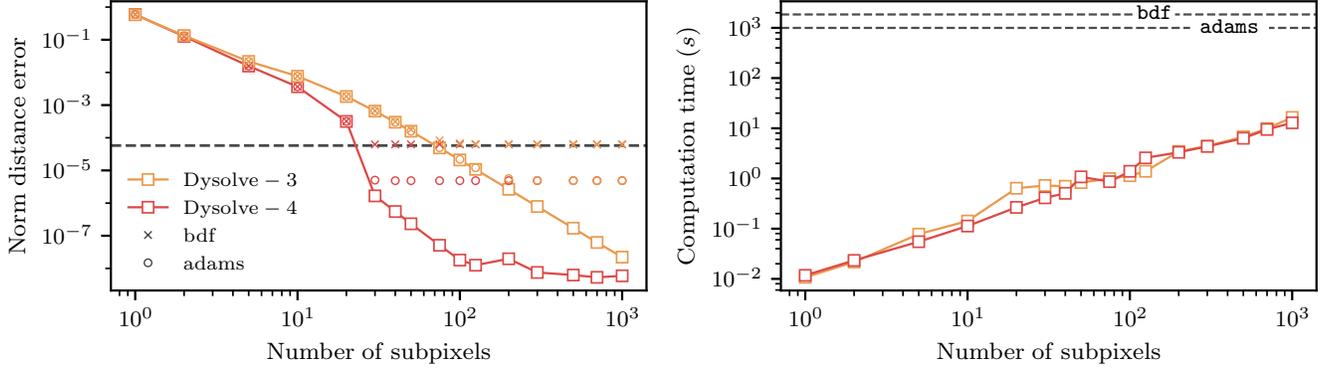


Figure 8. Comparison of the `Dysolve` algorithm and QuTiP’s `propagator` over an evolution of 20 Rabi oscillations, with the same parameters as in Fig 3 of the main text. **Left panel:** Frobenius norm distance between the `Dysolve` algorithm with different subpixel numbers and QuTiP’s `propagator` with `adams` (circles) and `bdf` (x’s) methods. The squares are the norm distance between `Dysolve-n` and `Dysolve-4` with 10^4 subpixels. The dashed line indicates the error between `propagator` with the `adams` and `bdf` methods. **Right panel:** Computational time of `Dysolve` and `propagator` for results of the left panel. The dashed lines refer to the computational time for `propagator` with `adams` and `bdf` with the highly accurate settings described in the text.

and unable to return solutions with a Frobenius norm error less than 5×10^{-6} , as seen in Fig. 8. The shortcoming of this numerical method was confirmed by the fact that the norm distance between `adams` and `bdf` was found to be significantly greater than the norm distance between `adams` and the `Dysolve` algorithm for a sufficient subpixel number. This is in addition to a computational time at least 2 order of magnitudes larger with the standard QuTiP approach as seen in the right panel of Fig. 8.