Practical Pulse Engineering: Gradient Ascent Without Matrix Exponentiation

Gaurav Bhole[1] and Jonathan A. Jones[1,*]

[1] *Centre for Quantum Computation, Clarendon Laboratory, University of Oxford, Parks Road, OX1 3PU, United Kingdom*

(Dated: April 23, 2018)

Since 2005 there has been a huge growth in the use of engineered control pulses to perform desired quantum operations in systems such as NMR quantum information processors. These approaches, which build on the original gradient ascent pulse engineering (GRAPE) algorithm, remain computationally intensive because of the need to calculate matrix exponentials for each time step in the control pulse. Here we discuss how the propagators for each time step can be approximated using the Trotter–Suzuki formula, and a further speed up achieved by avoiding unnecessary operations. The resulting procedure can give a substantial speed gain with negligible cost in propagator error, providing a more practical approach to pulse engineering.

PACS numbers:

Quantum information processors encode information in two-level quantum systems (qubits) and manipulate this through a series of elementary unitary transformations (quantum logic gates) [1, 2]. Quantum control seeks to implement some target unitary propagator U in a quantum system with background Hamiltonian $\mathcal{H}_0$ by applying some time-dependent Hamiltonian $\mathcal{H}_1(t)$. The resulting operator can be written as

$$V = \mathcal{T}\left(\exp\left[-\mathrm{i}\int \mathcal{H}_0 + \mathcal{H}_1(t)\,\mathrm{d}t\right]\right) \tag{1}$$

where $\mathcal{T}$ is the Dyson time-ordering operator. To make progress beyond this formal solution it is usually necessary to replace this continuously varying Hamiltonian by a piecewise constant form, so that

$$V = V_n V_{n-1} \ldots V_1, \quad V_j = \exp\left[-\mathrm{i}(\mathcal{H}_0 + \mathcal{H}_j)\delta t\right], \tag{2}$$

and to write the time-varying portion of the Hamiltonian as a weighted sum of a set of p distinct control fields

$$\mathcal{H}_j = \sum_{k=1}^{p} a_j^k \mathcal{H}^k. \tag{3}$$

Any particular control pulse can then be described by the corresponding set of amplitudes, a_j^k, and the time step δt, here taken as fixed. The quality of a control pulse can be measured by its fidelity with the desired operation U

$$\Phi = |\langle U|V\rangle|^2 \tag{4}$$

where the Hilbert-Schmidt inner product is defined by $\langle U|V\rangle = \mathrm{tr}(U^\dagger V)$, possibly normalised by the dimension of the operators [2]. The optimal control problem is then to find the set of amplitudes which maximises this fidelity, usually in the presence of practical constraints on the magnitudes of the amplitudes and the total length of the sequence. This is computationally challenging as the dimension of the underlying Hilbert space rises exponentially with the number of qubits to be controlled, although this difficulty can in some cases be reduced by using subsystems to simplify the calculations [3]. One recent approach [4] is to use subsystem methods to find approximate control pulses and then optimise these directly using the quantum system itself. Whatever approach is adopted, it is important to perform any computations as efficiently as possible.

I. GRADIENT ASCENT

Practical algorithms to maximize a function usually rely on the calculation of gradients, built from values of $\partial\Phi/\partial a_j^k$, but early work on optimal control was hampered by the belief that calculation of each of the np distinct elements of the gradient vector would require the full evaluation of V, and thus a total of n^2p sub-propagators. The design of control pulses was a major topic within Nuclear Magnetic Resonance (NMR) [5], but most NMR sequences were paramaterized by a small number of variables to keep the size of the computation manageable.

A breakthrough in 2005 was the realisation that it is not necessary to recalculate these sub-propagators, which can instead be stored and reused. The resulting algorithm, known as GRadient Ascent Pulse Engineering (GRAPE) [6], is now widely used, as are many variants, such as second order GRAPE [7], Newton–Raphson GRAPE [8], and GRadient Ascent in Functional Space (GRAFS) [9]. However, although this algorithm reduces the number of sub-propagator calculations from n^2p to n, the calculation of each of the n sub-propagators still requires a matrix exponential, which must be re-evaluated at each round of the search algorithm.

A variety of algorithms exist for calculating numerical matrix exponentials [10], but the most widely used methods combine scaling and squaring with Padé approximants [11, 12]. Whatever approach is used, this large number of matrix exponentials remains a key computa-

*Electronic address: jonathan.jones@physics.ox.ac.uk

tional bottleneck, and reducing the number of such operations would greatly speed up the entire process. Similar issues are encountered in other optimal control algorithms such as Krotov [13], Lyapunov [14], and Chopped Random Basis Optimization (CRAB) [15]. As we shall show below, however, when control pulses are made up from a large number of small rotations then the matrix exponentials can be efficiently approximated with negligible error.

To make further progress we will initially specialise to the case of a homonuclear NMR system of q distinct spin-1/2 nuclei, that is a homonuclear NMR implementation of q qubits, with a Hilbert space of dimension $N = 2^q$. Our efficient algorithm is not confined to this case, but it is useful to begin with a concrete example. The background Hamiltonian can then be written as

$$\mathcal{H}_0 = \sum_{r=1}^{q} \omega_r I_r^z + \sum_{s<r} \omega_{rs} \mathbf{I}_r \cdot \mathbf{I}_s \tag{5}$$

where I_r^z indicates $\frac{1}{2}\sigma_z$ on the r^{th} spin [5], and we are working in natural units such that $\hbar = 1$. In the case of weak coupling, that is $|\omega_{rs}| \ll |\omega_r - \omega_s|$, this can be approximated by

$$\mathcal{H}_0 = \sum_{r} \omega_r I_r^z + \sum_{s<r} \omega_{rs} I_r^z I_s^z. \tag{6}$$

The control Hamiltonian is provide by an RF oscillating magnetic field, with controllable amplitude and phase, but a fixed frequency close to resonance with the spins, which all have similar resonance frequencies in a homonuclear system. Transforming into a rotating frame, and making the rotating wave approximation, the control Hamiltonian is now the sum of two terms, each of which is applied identically to all the spins,

$$\mathcal{H}_j = x_j F^x + y_j F^y, \tag{7}$$

where $F^x = \sum_r I_r^x$ is the total x-operator on all spins, and similarly for F^y. The amplitudes of these control Hamiltonians are $x_j = \alpha_j \cos\phi_j$ and $y_j = \alpha_j \sin\phi_j$, where α_j is the magnitude of the applied field and ϕ_j is its phase. The background Hamiltonian $\mathcal{H}_0$ is unaffected, except that the individual spin frequencies are replaced by their offsets from the rotating frame frequency [5].

It appears necessary to use a full matrix exponential to evaluate each sub-propagator V_j in Eqn. 2, because $\mathcal{H}_j$ is the sum of two non-commuting terms, neither of which commutes with $\mathcal{H}_0$. The matrix exponential is, of course, simple to evaluate in the eigenbasis of $\mathcal{H}_0 + \mathcal{H}_j$, but finding this eigenbasis is as difficult as evaluating the matrix exponential directly. Progress can be made with an appropriate phase transformation [16], into a frame where $\mathcal{H}_j$ is aligned with the x-axis. This transformation is diagonal in the computational basis, and so easy to evaluate and perform. In particular we write

$$\mathrm{e}^{-\mathrm{i}(\mathcal{H}_0+\mathcal{H}_j)\delta t} = \mathrm{e}^{-\mathrm{i}\phi_j F^z} \mathrm{e}^{-\mathrm{i}(\mathcal{H}_0+\mathcal{H}_j^x)\delta t} \mathrm{e}^{\mathrm{i}\phi_j F^z} \tag{8}$$

where $\mathcal{H}_j^x = \alpha_j F^x$. However the inner matrix exponential still involves the sum of two non-commuting terms, and so appears to require a computationally intensive full matrix exponential.

II. APPROXIMATING UNITARIES

It is, however, possible to approximate the sub-propagator as long as δt is small enough, as all small-angle unitary evolutions approximately commute. If the terms are very small then it suffices to write

$$\mathrm{e}^{-\mathrm{i}(\mathcal{H}_0+\mathcal{H}_j^x)\delta t} \approx \mathrm{e}^{-\mathrm{i}\mathcal{H}_0\delta t} \mathrm{e}^{-\mathrm{i}\mathcal{H}_j^x\delta t} \approx \mathrm{e}^{-\mathrm{i}\mathcal{H}_j^x\delta t} \mathrm{e}^{-\mathrm{i}\mathcal{H}_0\delta t}, \tag{9}$$

but this Trotter approximation, implicitly used by Bhole and Mahesh [17], is only accurate to second order in δt. It is better to use the Trotter–Suzuki form [2, 18]

$$\mathrm{e}^{-\mathrm{i}(\mathcal{H}_0+\mathcal{H}_j^x)\delta t} \approx \mathrm{e}^{-\mathrm{i}\mathcal{H}_0\delta t/2} \mathrm{e}^{-\mathrm{i}\mathcal{H}_j^x\delta t} \mathrm{e}^{-\mathrm{i}\mathcal{H}_0\delta t/2}, \tag{10}$$

which is accurate to third order in δt. As the propagator fidelity depends quadratically on the size of the error, this approximation will be accurate to sixth order in δt when calculating fidelities and their associated gradients. A more thorough treatment of these errors is given below, but in our experience these errors are essentially negligible for pulse engineering calculations.

It might seem that approximating a matrix exponential by the product of three exponentials does not constitute progress. However, the two outer terms are constant, and so only need to be evaluated once for the entire GRAPE calculation, while the central Hamiltonian is a simple scalar multiple of F^x, and so will always be diagonal in the Hadamard basis. The basis transformation can be combined with the fixed outer terms, giving

$$V_j \approx \mathrm{e}^{-\mathrm{i}\phi_j F^z} W_1 \mathrm{e}^{-\mathrm{i}\alpha_j F^z \delta t} W_2 \mathrm{e}^{\mathrm{i}\phi_j F^z}, \tag{11}$$

with all explicit matrix exponentials now diagonal in the computational basis. The basis transformations

$$W_1 = \mathrm{e}^{-\mathrm{i}\mathcal{H}_0\delta t/2} \mathrm{H}^{(q)}, \quad W_2 = \mathrm{H}^{(q)} \mathrm{e}^{-\mathrm{i}\mathcal{H}_0\delta t/2}, \tag{12}$$

where $\mathrm{H}^{(q)}$ is the q-qubit Hadamard gate, are the same for every propagator and so are evaluated only once, requiring only a single full matrix exponential.

This expression does not depend on the weak coupling approximation, as strong couplings are unaffected by phase transformations. Similarly dipolar couplings are axially symmetric, and so once again unaffected. It can be trivially extended to heteronuclear spin systems, as control fields applied to one nuclear species do not affect spins of other species, and so all commute with one another. The resulting approximate expression is *identical* to Eqn. 11, except that the three diagonal matrix exponentials are now calculated over weighted sums of operators corresponding to each control field.

III. ERRORS AND SPEED GAINS

The errors in this approach are most simply considered for a one-spin system with $\mathcal{H}_0 = \omega_0 I^z$ and $\mathcal{H}_j = \alpha_j I^x$, which permits analytical calculations. The fidelity between the exact value of V_j, Eqn. 2, and its approximation, Eqn. 11, is

$$\Phi = 1 - \frac{\omega_0^2 \alpha_j^2 (\omega_0^2 + 4\alpha_j^2)}{2304} \delta t^6 + O(\delta t^8). \qquad (13)$$

For realistic frequencies and time steps in NMR systems, the infidelity of the approximate approach will be very small. Offset frequencies rarely exceed 15 kHz, and for the low RF powers used during long control pulses the nutation rate is usually below 5 kHz. Even for these extreme values, the error for a 10 μs time step is below 5×10^{-5}, and for the lower frequencies and smaller time steps normally used the error will far smaller.

Evaluating the error for pulse engineering in a real system is more complicated, reflecting the larger matrices involved, the large number of relevant frequencies, and the need to evaluate the error in the entire combined propagator, and not just the individual sub-propagators. The worst case occurs when each sub-propagator is identical: in this case the errors grow linearly with the number of steps, and so the infidelity, which depends on the square of the error, grows quadratically with the number of steps. However such a case is quite unrealistic in practice, as variations in the amplitude and phase of control fields mean that errors will partly cancel. Our simulations indicate that the infidelities remain small in realistic cases, typically around 10^{-4}.

The speed gains achieved by this approximate approach arise from avoiding full matrix exponentials by performing all calculations in an appropriate eigenbasis where the operators are diagonal. The new computational bottleneck is the calculation of matrix products, and since both matrix multiplication and numerical matrix exponentiation have a computational complexity of $O(N^3)$ for N-by-N matrices, these gains are approximately constant, and independent of the dimension of the Hamiltonian. (Much larger speed gains have been demonstrated in open system GRAPE [21], but only for state-to-state tasks.)

As three of the matrices in Eqn. 11 are diagonal most steps can be carried out using efficient partial matrix multiplication, and only one full matrix product is required (see the Appendix). The exact speed up achieved can be quite complex, due to implicit parallel computation on larger matrices. While this reduces the wall clock time, the CPU time will be increased to handle the overheads of parallelisation. This will affect the observed speed-up in a way that depends upon matrix size [19] as matrix exponentiation is more easily parallelised than our efficient algorithm. The speed gain achievable will depend on both the spin system chosen and the precise code used, but our MATLAB simulations for the 3-spin homonuclear system corresponding to the three ^{13}C nuclei in alanine [20] indicate that the calculation of sub-propagators can be sped up by a factor of around 18, while full propagators (which require an additional full matrix multiplication for each sub-propagator) are sped up by a factor of around 13. This speed gain means that a four-spin system can be simulated more rapidly with our approximate approach than an exact simulation of a three-spin system.

IV. REDUCING ERRORS

The infidelity in Eqn. 11 arises from the fact that $\mathcal{H}_j^x$ does not commute with $\mathcal{H}_0$, and can be minimised by making $\mathcal{H}_j^x$ as small as possible. This can be achieved by moving part of $\mathcal{H}_j^x$ into $\mathcal{H}_0$, writing

$$\mathcal{H}_0' = \mathcal{H}_0 + \Omega F^x, \quad \mathcal{H}_j' = \alpha_j' F^x = (\alpha_j - \Omega) F^x, \qquad (14)$$

where the offset frequency Ω is chosen to make the values α_j' as close to zero as possible, for example by setting it to half the maximum amplitude expected. While this will increase the infidelity of some individual sub-propagators, on average the infidelity will be decreased, and the fidelity of the total propagator will improve. The basis transformation operators, Eqn. 12, have to be calculated for the new value of $\mathcal{H}_0'$, but as this only needs to be done once for the entire calculation this gain in fidelity comes at no cost in time.

A more accurate approach is to choose Ω as the mean amplitude for the particular propagator being calculated: in this case the basis transformation operators have to be recalculated each time, but this still only requires one full matrix exponential for each propagator. If even higher accuracy is required then it is possible to use two different offset values, one for small values of α_j and one for large values, choosing the appropriate basis transformation in each case. Our simulations suggest that using a single value of Ω can improve the infidelity of the propagator by a factor around 15, while using two values can give an improvement of about 200, leading to propagator infidelities around 10^{-6}.

This could be improved still further by using a larger number of offsets, effectively trading memory for time [17], but these later gains are smaller than the early ones. When the numbers of offsets is large then all values of α_j' are small in comparison with all other frequencies, and in this limit the infidelity falls quadratically with the number of offsets, as expected from Eqn. 13. If very precise pulse engineering is required then it is simpler to use approximate techniques in the early stages of optimization, and switch to exact calculations using full matrix exponentials once the pulse fidelity is high enough to justify this.

V. CONCLUSIONS

We have applied our efficient algorithm with a single offset frequency to calculate GRAPE control pulses in a variety of NMR systems, observing a speed increase of at least a factor of 6. We have checked each pulse against a conventional full matrix exponential calculation, and the error in the fidelity of the final propagator has always been below 10^{-4}. As typical experiments in this field seek a fidelity of around 0.999 [22] this error is effectively negligible, and approximate propagators provide an entirely practical approach to pulse engineering.

Although developed, like GRAPE, for use in NMR, our approach could be used in other fields where GRAPE has been applied such as ESR [23], NV centres [24], ion traps [25], and circuit QED [26]. The key requirement is that control Hamiltonians either commute with one another or can be converted to commuting forms using phase transformations which commute with the background Hamiltonian, and that the individual sub-propagators remain small enough to use Trotter–Suzuki approximations.

Acknowledgments

G. Bhole is supported by a Felix Scholarship.

APPENDIX: EFFICIENT MATRIX MULTIPLICATION

As matrix multiplication is now the computational bottleneck, it is vital that this is carried out as efficiently as possible. Full matrix multiplication, with computational complexity of $O(N^3)$, should only be used when actually necessary: when multiplying a full matrix by a diagonal matrix only $O(N^2)$ steps are required.

If the algorithm is coded in a low level language then it is easy to ensure this, but in higher level languages, such as MATLAB, it is necessary to code carefully. Diagonal matrices should be stored not as matrices but as vectors, to avoid unnecessary operations. In particular the exponentials of diagonal matrices must be calculated using direct exponentiation of the individual elements. To combine the individual matrices, Eqn. 11 can be written as

$$\{\phi\}W_1\{\alpha\}W_2\{\phi^*\} \tag{A.1}$$

where braces indicate diagonal matrices stored as vectors. The multiplications by the two outer diagonal matrices can be combined, and the overall process written as

$$\Phi .* (W_1 * (\alpha .* W_2)), \quad \text{with} \quad \Phi = \phi .* \phi', \tag{A.2}$$

where $*$ is the MATLAB operator for a full matrix multiplication, $.*$ is the operator for an element-by-element multiplication, and ϕ' indicates the adjoint of the vector ϕ. Note that only a single full matrix multiplication is required.

Combining the phase multiplications into a single matrix Φ is particularly useful when developing pulses which are robust to variations in the RF coupling strength. These are simulated by evaluating propagators with the control amplitude set to a range of values [22] (e.g., 95%, 100% and 105% of the nominal value). In such cases the matrix Φ is the same for all the different coupling strengths, and need only be calculated once.

When programming in MATLAB it is also important to think carefully about memory handling. In particular it is quicker to evaluate Eqn. A.2 one multiplication at a time, storing intermediate results in explicit variables. If the multiplications are carried out in one line then temporary variables are created to hold intermediate results, and subsequently destroyed. If equivalent calculations are carried out many times, as happens when evaluating propagators, it is quicker to reuse previously allocated variables. These minor issues are almost irrelevant in conventional GRAPE calculations, where the time needed for matrix exponentiation dominates over everything else, but become important once all these slow stages have been removed.

It might appear possible to speed up calculations further, for example by using the structure in F^z to avoid repeatedly calculating the same exponential terms. This would certainly be sensible when programming in a low level language, but in our experience such tricks actually slow MATLAB down. It can be difficult to predict precisely what will give the fastest MATLAB code, and experimentation is the best approach.

[1] C. H. Bennett and D. P. DiVincenzo, Nature **404**, 247 (2000).
[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (CUP, 2000).
[3] C. A. Ryan, C. Negrevergne, M. Laforest, E. Knill, and R. Laflamme, Phys. Rev. A **78**, 012328 (2008).
[4] D. Lu, K. Li, J. Li, H. Katiyar, A. J. Park, G. Feng, T. Xin, H. Li, G. Long, A. Brodutch, et al., npj Quantum Information **3**, 45 (2017).
[5] R. R. Ernst, G. Bodenhausen, and A. Wokaun, *Principles of Nuclear Magnetic Resonance in One and Two Dimensions* (Oxford University Press, 1987).
[6] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, J. Magn. Reson. **172**, 296 (2005).
[7] P. De Fouquieres, S. G. Schirmer, S. J. Glaser, and I. Kuprov, J. Magn. Reson. **212**, 412 (2011).
[8] D. L. Goodwin and I. Kuprov, J. Chem. Phys. **144**, 204107 (2016).
[9] D. G. Lucarelli, preprint arXiv:1611.00188 (2016).
[10] C. Moler and C. V. Loan, SIAM Review **45**, 3 (2003).

[11] N. J. Higham, SIAM J. Matrix Anal. Appl. **26**, 1179 (2005).
[12] A. H. Al-Mohy and N. J. Higham, SIAM J. Matrix Anal. Appl. **31**, 970 (2009).
[13] I. I. Maximov, Z. Tošner, and N. C. Nielsen, J. Chem. Phys. **128**, 05B609 (2008).
[14] S. C. Hou, L. C. Wang, and X. X. Yi, Phys. Lett. A **378**, 699 (2014).
[15] T. Caneva, T. Calarco, and S. Montangero, Phys. Rev. A **84**, 022326 (2011).
[16] G. Bhole, V. S. Anjusha, and T. S. Mahesh, Phys. Rev. A **93**, 042339 (2016).
[17] G. Bhole and T. S. Mahesh, preprint arXiv:1707.02162 (2017).
[18] M. Suzuki, J. Stat. Phys. **43**, 883 (1986).
[19] K. Waldherr, T. Huckle, T. Auckenthaler, U. Sander, and T. Schulte-Herbrüggen, *High Performance Computing in Science and Engineering* (Springer, 2010), chap. Fast 3D Block Parallelisation for the Matrix Multiplication Prefix Problem, pp. 39–50.
[20] D. G. Cory, M. D. Price, W. Maas, E. Knill, R. Laflamme, W. H. Zurek, T. F. Havel, and S. S. Somaroo, Phys. Rev. Lett. **81**, 2152 (1998).
[21] S. Boutin, C. K. Andersen, J. Venkatraman, A. J. Ferris, and A. Blais, Phys. Rev. A **96**, 042315 (2017).
[22] T. Xin, S. Huang, S. Lu, K. Li, Z. Luo, Z. Yin, J. Li, D. Lu, G. Long, and B. Zeng, Sci. Bull. **63**, 17 (2018).
[23] Y. Zhang, C. A. Ryan, R. Laflamme, and J. Baugh, Phys. Rev. Lett. **107**, 170503 (2011).
[24] F. Dolde, V. Bergholm, Y. Wang, I. Jakobi, B. Naydenov, S. Pezzagna, J. Meijer, F. Jelezko, P. Neumann, T. Schulte-Herbrggen, et al., Nat. Commun. **5**, 3371 (2014).
[25] V. Nebendahl, H. Häffner, and C. F. Roos, Phys. Rev. A **79**, 012312 (2009).
[26] R. Fisher, F. Helmer, S. J. Glaser, F. Marquardt, and T. Schulte-Herbrüggen, Phys. Rev. B **81**, 085328 (2010).