





TEMPO: A Python Package for Time Evolution of Pulse Sequences in QuTiP

Jner Tzern Oon ^{1,2*}, Sonja A. Hakala^{1*}, George A. Witt¹, and Ronald Walsworth^{1,2,3}

¹ Department of Physics, University of Maryland, College Park, Maryland 20742, USA ² Quantum Technology Center, University of Maryland, College Park, Maryland 20742, USA ³ Department of Electrical and Computer Engineering, University of Maryland, College Park, Maryland 20742, USA  Corresponding author * These authors contributed equally.

DOI: [N/A](#)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: 01 January 1970

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

TEMPO (Time-dependent Evolution of Multiple Pulse Operations) offers accessible and efficient simulations of pulse sequences in Python, using the suite of master equation solvers available in the Quantum Toolbox in Python (QuTiP). It enables straightforward definition of pulse sequence structures, including any underlying time-dependent Hamiltonians and pulse timing information, and faster simulations of pulse sequence dynamics (compared to naive implementations using QuTiP) while remaining compatible with the existing collection of QuTiP subpackages. Given the ubiquitous use of pulse sequences throughout quantum information/computing sciences, magnetic resonance studies, and quantum metrology, this work has immediate relevance to a wide array of research applications.

Statement of Need

Pulse sequences typically contain a series of discrete operations (pulses) using radio frequency, microwave, or optical fields. Their application for quantum control has an extensive history across atomic physics ([Vitanov et al., 2001](#)); nuclear magnetic resonance ([Levitt, 2013](#)); solid-state spin systems for quantum sensing ([Barry et al., 2020](#)); and a broad range of other platforms. In recent years, research into quantum technologies has driven the development of advanced software tools for numerical simulations of quantum systems ([Fingerhuth et al., 2018](#)). In particular, the QuTiP (Quantum Toolbox in Python) framework provides open-source tools for simulations of open quantum systems, and has received prolific use across numerous quantum applications ([Johansson et al., 2012, 2013](#)). Utilizing the master equation solvers that are native to QuTiP, TEMPO provides two key advantages for numerical simulations of pulse sequence dynamics.

Ease of Use: By incorporating both the characteristics of a Hamiltonian and its time constraints as necessary properties, pulses are first constructed individually, then collated to form a 'pulse sequence'. Time evolution is performed without the need to manually deactivate each pulse Hamiltonian outside its given time interval. Using pulse 'recipes' in TEMPO, the creation of pulses with overlapping functional forms is streamlined, with parameters that can be tuned for individual pulses.

Faster Executions of Time Evolution: TEMPO organizes each pulse sequence as a series of time segments, preserving only the pulses that are active within each segment. This avoids overheads incurred by repeated inspections of inactive pulse(s), significantly speeding up evaluation of system evolution.

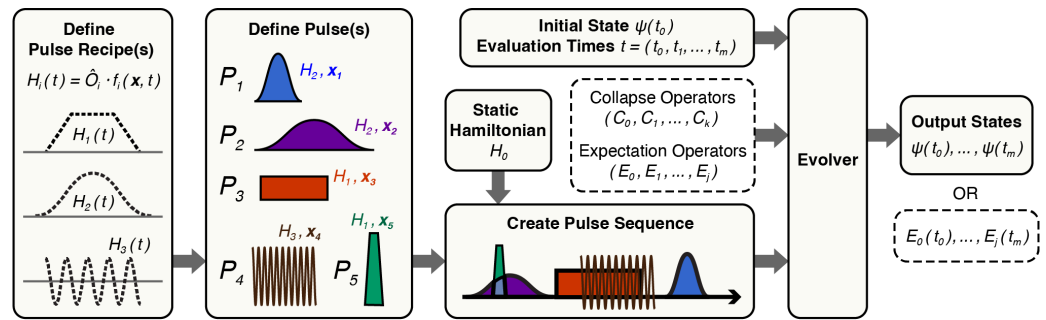


Figure 1: Outline of steps in TEMPO used to perform time evolution due to a pulse sequence. First, each pulse recipe contains the functional form of a time-dependent Hamiltonian $H_i(t) = \hat{O}_i \cdot f_i(\mathbf{x}, t)$. Pulse recipe(s) are used to create individual pulse(s) P_1, \dots, P_n , with individual parameter settings $\mathbf{x}_1, \dots, \mathbf{x}_n$, respectively. The pulses are organized into a sequence along with an optional static Hamiltonian. Next, the pulse sequence is provided to the Evolver, with an initial system state $\psi(t_0)$ and an array of evaluation times (t_0, t_1, \dots, t_m) . Time evolution returns the state $\psi(t_0), \dots, \psi(t_m)$ at these times; if operators E_0, \dots, E_j are provided, the operator expectation values $E_0(t_0), \dots, E_j(t_0)$ are calculated instead.

Usage

There are five main classes that make up a simulation in TEMPO: Pulse_Recipe, Pulse, Hamiltonian, Pulse_Sequence, and Evolver. In general, simulations can be easily executed by following the steps outlined below, which are illustrated in Figure 1.

1. Create a pulse recipe by defining the functional form of the time-dependent Hamiltonian

$$\hat{H}_i(t) = \hat{O}_i \cdot f_i(\mathbf{x}, t).$$

The user provides an operator/matrix \hat{O} and scalar function $f(\mathbf{x}, t)$ that depends on input parameters \mathbf{x} and time t .

2. Create individual pulse(s) P_1, \dots, P_n by providing a pulse recipe, individual parameter settings $\mathbf{x}_1, \dots, \mathbf{x}_n$, and pulse timing information.
3. Generate a pulse sequence by inputting the pulses P_1, \dots, P_n , and optionally a time-independent (static) Hamiltonian.
4. To evolve the system in time, the pulse sequence is provided to the Evolver class along with the initial system state $\psi(t_0)$ (state vector or density matrix) and an array of time points t_0, t_1, \dots, t_m . The system state is returned at these times. It is also possible to provide collapse operators for evolution using the Lindblad (open) master equation, along with operators for calculation of expectation values at these times.
5. The Evolver returns the system state $\psi(t_0), \dots, \psi(t_m)$ or operator expectation values at the times provided.

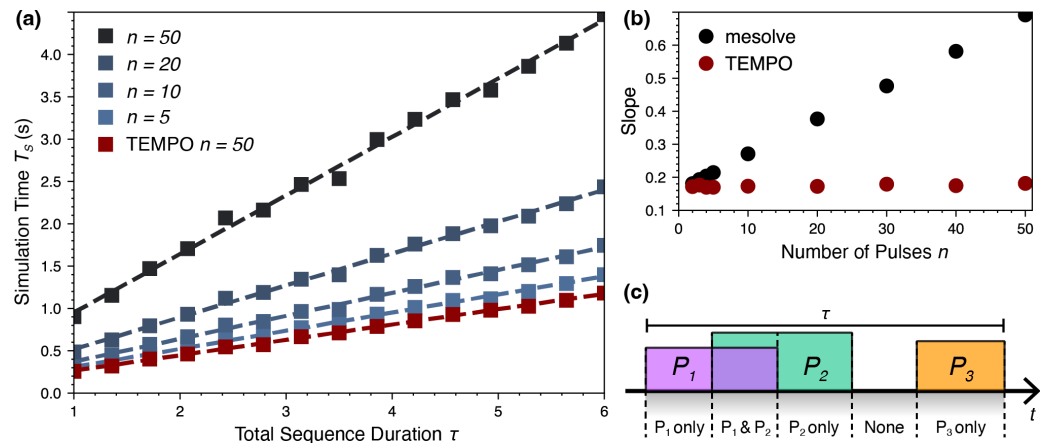


Figure 2: (a) Average simulation wall-clock time T_S as a function of pulse sequence duration τ without using TEMPO, for number of pulses $n = 5, 10, 20, 50$, shown in shades of blue. Average simulation times using TEMPO are shown in red at high pulse number count ($n = 50$). Best-fit curves are displayed as dashed lines, with slope values plotted in (b) for measurements with TEMPO in red and without TEMPO (via QuTiP's mesolve function) in black. (c) Example visualization of pulse sequence segmentation performed by TEMPO, where $n = 3$ pulses result in $2n - 1 = 5$ time segments for solver evaluation. All measurements of T_S are average values over 20 repeated runs.

Efficient Simulation Times:

Using the existing master equation solvers available in QuTiP, TEMPO provides significant speedups in simulations of multi-pulse sequences. Without TEMPO, the simulation (wall-clock) time T_S generally follow a linear trend with respect to 1) the total duration of a pulse sequence τ , and 2) the number of pulses n , roughly obeying

$$T_S \propto n \cdot \tau.$$

This dependence is highlighted in Figure 2(a), which shows measurements of the average simulation time with varying τ . Each set of measurements, shown as points of the same color, is performed for a fixed number of pulses n . Without TEMPO (shades of blue), the fitted slope for each set of measurements increases with n , exhibiting a linear dependence as shown in Figure 2(b). In contrast, the use of TEMPO (red) results in simulation times that are nearly independent of the number of pulses n , instead following $T_S \propto \tau$. As a result, the slope of each best-fit line using TEMPO stays roughly constant with respect to n , as shown in Figure 2(b).

This advantage can be understood by considering how the solvers are constructed natively (without TEMPO). At each timestep during the pulse sequence, the solver typically inspects if each pulse is active at this time. This results in n checks at each timestep and thus the rough linear dependence of T_S with respect to n .

Instead, TEMPO first divides the pulse sequence into consecutive time segments for efficient time evolution, illustrated by an example in Figure 2(c). By creating segment breaks at the start and end of each pulse, TEMPO preserves only the active pulses in each of the $2n - 1$ segments. The solver is then executed consecutively across each segment, without any redundant checks of inactive pulses. As a result, simulation times are largely independent of n , besides minor overheads from repeated use of the solver.

Related Pulse Sequence Simulation Software

Existing open-source software that offer compatibility with pulse sequence simulations include QuTiP-based packages such as SeQuencing (Horn, 2022) and PULSEE [Candoli2023]. Other software suitable for open quantum simulations of pulse sequence dynamics include Qiskit

Dynamics in Python (Alexander et al., 2020; Puzzuoli et al., 2023), QuantumOptics.jl in Julia for speed and scalability to large system sizes (Krämer et al., 2018), and Spinach in MATLAB for nuclear magnetic resonance systems (Hogben et al., 2011).

Acknowledgements

We thank Saipriya Satyajit, Katrijn Everaert, Declan Daly, Kevin Olsson and John Blanchard for testing the package and providing feedback during development. This work is supported by, or in part by, the DEVCOM Army Research Laboratory under Contract Numbers W911NF1920181 and W911NF2420143; the DEVCOM ARL Army Research Office under Grant Number W911NF2120110; the U.S. Air Force Office of Scientific Research under Grant Number FA95502210312; and the University of Maryland Quantum Technology Center.

Alexander, T., Kanazawa, N., Egger, D. J., Capelluto, L., Wood, C. J., Javadi-Abhari, A., & C McKay, D. (2020). Qiskit pulse: Programming quantum computers through the cloud with pulses. *Quantum Science and Technology*, 5(4), 044006. <https://doi.org/10.1088/2058-9565/aba404>

Barry, J. F., Schloss, J. M., Bauch, E., Turner, M. J., Hart, C. A., Pham, L. M., & Walsworth, R. L. (2020). Sensitivity optimization for NV-diamond magnetometry. *Reviews of Modern Physics*, 92(1). <https://doi.org/10.1103/revmodphys.92.015004>

Fingerhuth, M., Babej, T., & Wittek, P. (2018). Open source software in quantum computing. *PLOS ONE*, 13(12), e0208561. <https://doi.org/10.1371/journal.pone.0208561>

Hogben, H. J., Krzystyniak, M., Charnock, G. T. P., Hore, P. J., & Kuprov, I. (2011). Spinach – a software library for simulation of spin dynamics in large spin systems. *Journal of Magnetic Resonance*, 208(2), 179–194. <https://doi.org/10.1016/j.jmr.2010.11.008>

Horn, L. B.-V. (2022). *Sequencing-dev/sequencing: v1.2.0*. Zenodo. <https://doi.org/10.5281/ZENODO.4515634>

Johansson, J. R., Nation, P. D., & Nori, F. (2012). QuTiP: An open-source python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 183(8), 1760–1772. <https://doi.org/10.1016/j.cpc.2012.02.021>

Johansson, J. R., Nation, P. D., & Nori, F. (2013). QuTiP 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4), 1234–1240. <https://doi.org/10.1016/j.cpc.2012.11.019>

Krämer, S., Plankensteiner, D., Ostermann, L., & Ritsch, H. (2018). QuantumOptics. JI: A julia framework for simulating open quantum systems. *Computer Physics Communications*, 227, 109–116.

Levitt, M. H. (2013). *Spin dynamics* (2nd ed.). John Wiley & Sons.

Puzzuoli, D., Wood, C. J., Egger, D. J., Rosand, B., & Ueda, K. (2023). Qiskit dynamics: A python package for simulating the time dynamics of quantum systems. *Journal of Open Source Software*, 8(90), 5853. <https://doi.org/10.21105/joss.05853>

Vitanov, N. V., Fleischhauer, M., Shore, B. W., & Bergmann, K. (2001). *Coherent manipulation of atoms molecules by sequential laser pulses* (pp. 55–190). Elsevier. [https://doi.org/10.1016/S1049-250X\(01\)80063-X](https://doi.org/10.1016/S1049-250X(01)80063-X)