Simphony: an open source Python package to simulate point defect dynamics

Péter Boross[1], György Fehér[1], Győző Egri[1], and András Pályi[1,2,3]

[1]Qutility @ Faulhorn Labs, Budapest, Hungary (info@qutility.io)

[2]Department of Theoretical Physics, Budapest University of Technology and Economics, Budapest, Hungary

[3]HUN-REN-BME-BCE Quantum Technology Research Group, Budapest, Hungary

qutility.io

info@qutility.io

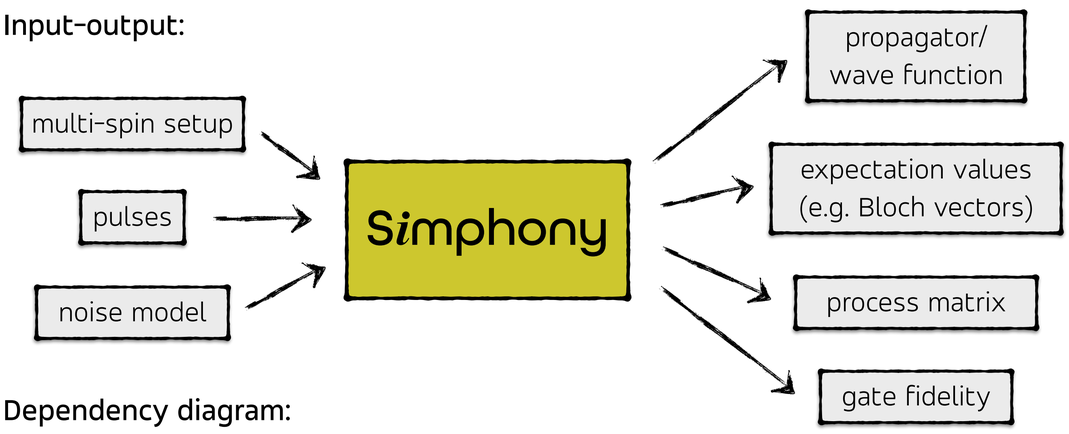
Abstract

Electronic and nuclear spins around point defects are key building blocks for quantum communication, quantum sensing, and quantum computing technologies. In this work, we present Simphony, a Python package designed to simulate the quantum dynamics of point defect spin ensembles efficiently and intuitively. Simphony enables users to define simulation setups with minimal, high-level Python code, describing the multi-spin configuration, noise model, and static or dynamic control parameters.

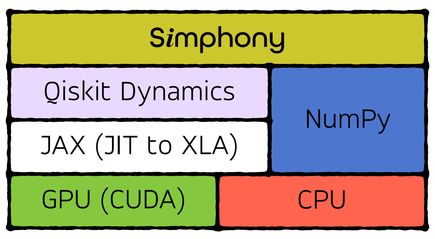
The package computes time-evolved wave functions and time-evolution operators, evaluates gate properties such as fidelity, and process matrices generated by user-defined pulse sequences. Simphony performs exact time evolution, capturing subtle effects such as Bloch–Siegert oscillations and leakage from the computational subspace—phenomena that are often neglected by standard approximations. Simphony supports GPU acceleration, automatic differentiation for gradient-based optimization, and the simulation of interacting spin defects. For NV centers, built-in hyperfine tensors allow for physical parametrization with minimal input, making it well-suited for both research and educational use. It also provides a flexible framework to define general point defects beyond the NV center. By making simulations simple, efficient, and noise-aware, Simphony facilitates the design and optimization of pulse sequences for applications in quantum technologies.

What Simphony does?

Input-output:

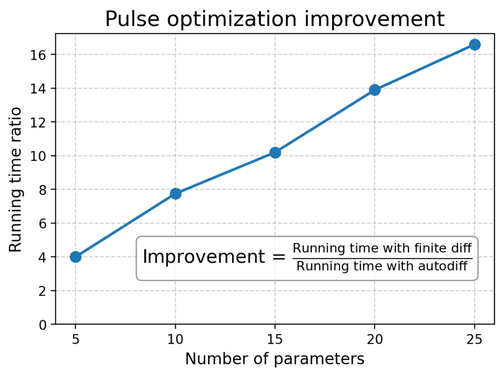


Dependency diagram:



Features:

- Predefined **NV-center** model with ^{13}C **hyperfine** interaction **database** by V. Ivády's group
- Local quasi-static noise model included
- Accelerated by Just-in-Time compilation
- Automatic differentiation for gradient-based **pulse optimization**



Roadmap:

- Scalable registers of coupled NV centers
- Lindblad-type noise models
- Non-Markovian system-bath interaction
- Photophysics dynamics: initialization and readout

Showcase

Multi-spin setup:

```python
spin_e = simphony.Spin(
    dimension = 3,
    name = 'e',
    qubit_subspace = (0, 1),
    gyromagnetic_ratio = 28033.1, # MHz/T
    zero_field_splitting = 2872., # MHz
)

spin_C = simphony.Spin(
    dimension = 2,
    name = 'C',
    qubit_subspace = (-1/2, 1/2),
    gyromagnetic_ratio = -10.7084, # MHz/T
    zero_field_splitting = 0
)

hyperfine = simphony.Interaction(spin_e, spin_C)
hyperfine.xx = -0.0672703 # MHz
hyperfine.yy = -0.0672701 # MHz
hyperfine.zz = 0.0958844 # MHz

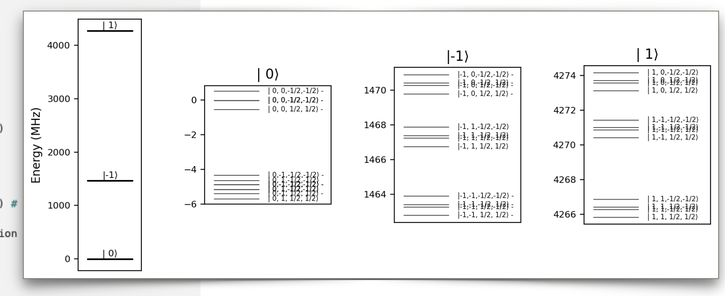
static_field = simphony.StaticField([0,0,0.015]) #

driving_field_MW = simphony.DrivingField(direction

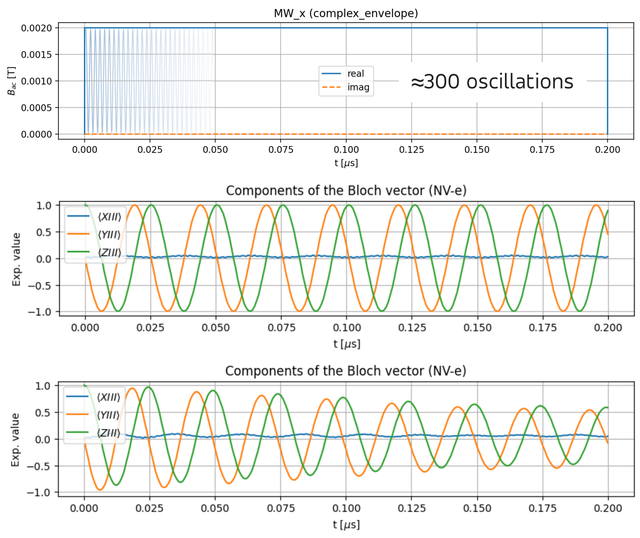
model = simphony.Model()

model.add_spin(spin_e)
model.add_spin(spin_C)
```

```python
model = simphony.default_nv_model(nitrogen_isotope = 14,
                                  static_field_strength = 0.05,
                                  carbon_atom_indices = [(1,1,1,0),
                                                         (1,2,-4,0)])
model.plot_levels()
```



Simulating electron spin resonance (ESR) pulse:

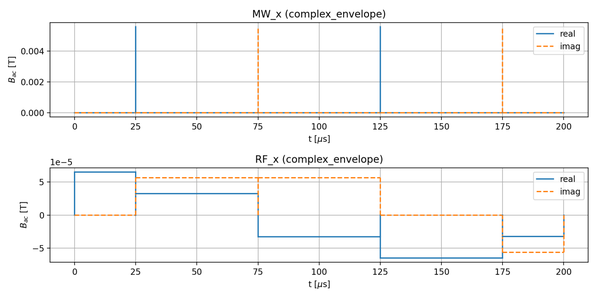


- Numerically exact simulation
- Without rotating wave approx.
- Single sine wave method

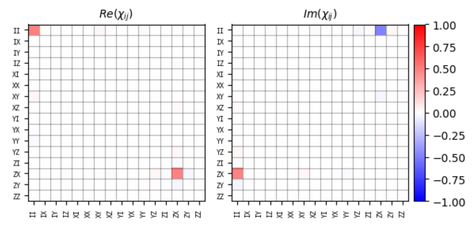
Noiseless simulation → Rabi oscillations

Quasi-static noise → Damped Rabi oscillations

Simulating dynamical decoupled radio-frequency (DDRF) pulse sequence:



Process matrix:



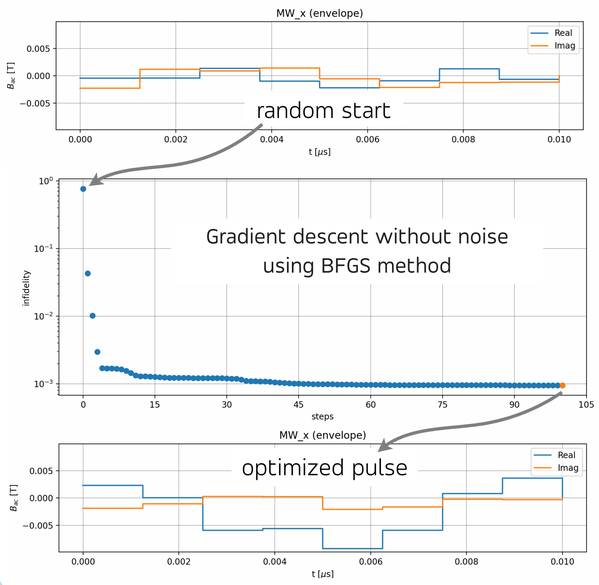
Ideal gate: $\mathrm{CROT} = \underbrace{|-1\rangle\langle -1|}_{\text{electron}} \otimes \overbrace{\mathrm{RX}(\pi/2)}^{\text{nuclear spin}} + \underbrace{|0\rangle\langle 0|}_{\text{electron}} \otimes \overbrace{\mathrm{RX}(-\pi/2)}^{\text{nuclear spin}}.$

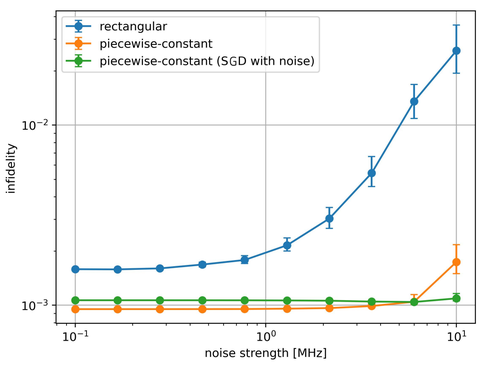
Fidelity: over 99%

Pulse optimization (partially in Quasar):

- Target gate: RX(π) gate on the electron
- Cost function: infidelity
- Ansatz: 8-step piecewise-constant pulse

Random parameters → Infidelity(parameters) → Gradient from autodiff → Update the parameters





Noise-aware optimization → Stochastic gradient descent (SGD) → 1 step ⟼ 1 noise realization

Acknowledgment

This research was supported by the National Research Development and Innovation Office of Hungary within the German-Hungarian EUREKA programme (Project No. 2020-1.2.3-EUREKA-2022-00029).

National Research, Development and Innovation Office Hungary / Project financed from the NRDI Fund

Quasar

Quasar is a Python package for compiling quantum circuits on star-topology architectures (QUantum Assembler for Star-topology ARchitectures). It provides optimized pulse sequences for NV center–based systems in quantum computing or quantum sensing contexts. Compilation is performed at two levels: into gate sequences, and into pulse sequences.