Branch: master ▾

Find file | Copy path

krotov / tests / **test_mu.py**

goerz Fix `derivative_wrt_pulse` (#47, #48)

cb0d333 Jul 9, 2019

1 contributor

Raw | Blame | History

131 lines (118 sloc) | 4.03 KB

```python
"""Tests for krotov.mu"""
import pytest
from qutip import ket, sigmam, sigmap, sigmax, sigmaz

import krotov


@pytest.fixture
def tls_control_system():
    """Non-trivial control system defined on a TLS"""
    eps1 = lambda t, args: 0.5
    eps2 = lambda t, args: 1
    H1 = [0.5 * sigmaz(), [sigmap(), eps1], [sigmam(), eps1]]
    H2 = [0.5 * sigmaz(), [sigmaz(), eps2]]
    c_ops = [0.1 * sigmap()]
    objectives = [
        krotov.Objective(
            initial_state=ket('0'), target=ket('1'), H=H1, c_ops=c_ops
        ),
        krotov.Objective(
            initial_state=ket('0'), target=ket('1'), H=H2, c_ops=c_ops
        ),
    ]
    controls = [eps1, eps2]
    controls_mapping = krotov.structural_conversions.extract_controls_mapping(
        objectives, controls
    )
    return objectives, controls, controls_mapping


@pytest.fixture
def tls_control_system_tdcops(tls_control_system):
    """Control system with time-dependent collapse operators"""
    objectives, controls, _ = tls_control_system
    c_op = [[0.1 * sigmap(), controls[0]]]
    c_ops = [c_op]
    H1 = objectives[0].H
    H2 = objectives[1].H
    objectives = [
        krotov.Objective(
            initial_state=ket('0'), target=ket('1'), H=H1, c_ops=c_ops
        ),
        krotov.Objective(
            initial_state=ket('0'), target=ket('1'), H=H2, c_ops=c_ops
        ),
    ]
    controls_mapping = krotov.structural_conversions.extract_controls_mapping(
        objectives, controls
    )
    return objectives, controls, controls_mapping


def test_derivative_wrt_pulse_multiple_terms(tls_control_system):
    """Test the calculation of μ if the same control appears more than once"""
    objectives, pulses, pulses_mapping = tls_control_system
    # distinction between controls and pulses doesn't matter here, we're only
    # considering linear controls and don't plug in any time_index
    i_objective = 0
    mu = krotov.mu.derivative_wrt_pulse(
        objectives,
        i_objective,
        pulses,
        pulses_mapping,
        i_pulse=0,
        time_index=0,
    )
    # 0.5 * (σ₊ + σ₋) = σₓ
    for state in (ket('0'), ket('1')):
        assert (mu(state) - (sigmax())(state)).norm('max') == 0
        assert (mu(state)).dims == state.dims


def test_derivative_wrt_pulse_zero(tls_control_system):
    """Test that μ=0 if taking derivative wrt pulse not in objective"""
    objectives, pulses, pulses_mapping = tls_control_system
    # distinction between controls and pulses doesn't matter here, we're only
    # considering linear controls and don't plug in any time_index
    i_objective = 0
    mu = krotov.mu.derivative_wrt_pulse(
        objectives,
        i_objective,
        pulses,
        pulses_mapping,
        i_pulse=1,
        time_index=0,
    )
    for state in (ket('0'), ket('1')):
        assert mu(state).norm('max') == 0
        assert (mu(state)).dims == state.dims

    i_objective = 1
    mu = krotov.mu.derivative_wrt_pulse(
        objectives,
        i_objective,
        pulses,
        pulses_mapping,
        i_pulse=0,
        time_index=0,
    )
    for state in (ket('0'), ket('1')):
        assert mu(state).norm('max') == 0
        assert (mu(state)).dims == state.dims


def test_derivative_wrt_pulse_no_timedependent_cops(tls_control_system_tdcops):
    """Test that time-dependent collapse operators are no allowed"""
    objectives, pulses, pulses_mapping = tls_control_system_tdcops
    i_objective = 0
    with pytest.raises(NotImplementedError):
        krotov.mu.derivative_wrt_pulse(
            objectives,
            i_objective,
            pulses,
            pulses_mapping,
            i_pulse=0,
            time_index=0,
        )
    # however, we do allow the c_ops to be time-dependent with controls we're
    # not taking the derivative with respect to
    mu = krotov.mu.derivative_wrt_pulse(
        objectives,
        i_objective,
        pulses,
        pulses_mapping,
        i_pulse=1,
        time_index=0,
    )
    for state in (ket('0'), ket('1')):
        assert mu(state).norm('max') == 0
        assert (mu(state)).dims == state.dims
```