5.a-d) SO(2) Mapping Functions

Code Snippet 1: SO(2) Mapping Functions

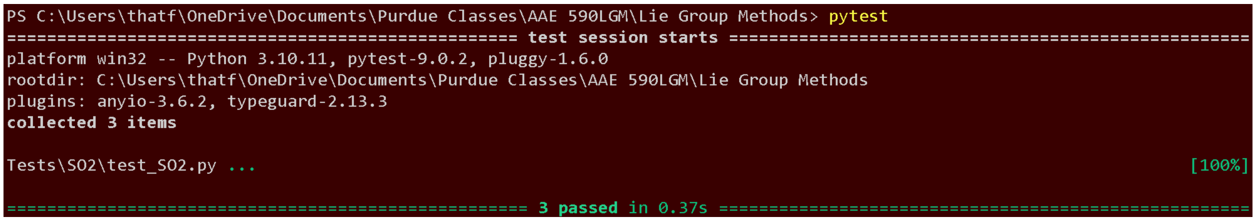
```
import numpy as np

def so2_wedge(theta):
    """
    Maps from so2 real number parametrization to so2 Lie algebra

    :param theta: angle in R
    """
    Omega = np.array([[0, -theta],
                      [theta,  0]])

    return Omega

def so2_vee(Omega):
    """
    Maps from so2 Lie algebra to its real number parametrization

    :param Omega: 2x2 skew symmetric matrix (in so2 Lie algebra)
    """

    theta = Omega[1, 0]

    return theta

def so2_exp(theta):
    """
    Maps from parametrization of so2 Lie algebra to SO2 Lie group

    :param theta: angle in R
    """

    R = np.array([[np.cos(theta), -np.sin(theta)],
                  [np.sin(theta),  np.cos(theta)]])

    return R

def so2_log(R):
   """
   Maps from SO2 Lie group to the parametrization of its Lie algebra so2

   :param R: 2D rotation matrix
   """
   theta = np.arctan2(R[1, 0], R[0, 0])

   return theta
```

5.e) Unit Tests

Code Snippet 2: Unit Tests Using Pytest

```
import pytest as pt
import numpy as np
from Code.SO2.SO2_maps import so2_wedge, so2_vee, so2_exp, so2_log

def test_so2_exp_log():
    """
    Test that exp(log(R)) = R for random rotation matrices
    """
    rng = np.random.default_rng()
    theta_random = rng.uniform(-10, 10, 100)

    checks = np.zeros([theta_random.size, 1])
    for i in range(theta_random.size):
        R_random = so2_exp(theta_random[i])
        checks[i] = np.all(so2_exp(so2_log(R_random)) == pt.approx(R_random))

    assert np.all(checks)

def test_so2_log_exp():
    """
    Test that log(exp(theta)) = theta for all theta in (-pi, pi]
    """
    theta_array = -np.linspace(-np.pi, np.pi, 100)

    checks = np.zeros([theta_array.size, 1])
    for i in range(theta_array.size):
        checks[i] = np.all(so2_log(so2_exp(theta_array[i])) == pt.approx(theta_array[i]))

    assert np.all(checks)

def test_so2_commutativity():
    """
    Test that R(theta_1)R(theta_2) = R(theta_1 + theta_2)
    """
    iter = 100

    rng = np.random.default_rng()
    theta_1 = rng.uniform(-10, 10, iter)
    theta_2 = rng.uniform(-10, 10, iter)

    checks = np.zeros([iter, 1])
    for i in range(iter):
        checks[i] = so2_exp(theta_1[i]) @ so2_exp(theta_2[i]) == pt.approx(so2_exp(theta_1[i] + theta_2[i]))

    assert np.all(checks)
```

Figure 1: Pytest unit test output showing all 3 tests passing



```
PS C:\Users\thatf\OneDrive\Documents\Purdue Classes\AAE 590LGM\Lie Group Methods> pytest
================================================= test session starts ==================================================
platform win32 -- Python 3.10.11, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\thatf\OneDrive\Documents\Purdue Classes\AAE 590LGM\Lie Group Methods
plugins: anyio-3.6.2, typeguard-2.13.3
collected 3 items

Tests\SO2\test_SO2.py ...                                                                                         [100%]

================================================== 3 passed in 0.37s ===================================================
```