Christian B. Mendl, Martina Nibbi, Pedro Hack, Irene López Gutiérrez due: 26 Jun 2023, 08:00 on Moodle

Tutorial 9 (Quantum signal processing theorem[1])
We study the theoretical framework underlying the QSP theorem stated in the lecture. We study the possible unitary transformations achievable by a sequence of d primitive gates of the form

$$\hat{R}_\varphi(\theta) = \exp\left(-i\frac{\theta}{2}\hat{\sigma}_\varphi\right), \tag{1}$$

where $\sigma_\varphi = \cos(\varphi)X + \sin(\varphi)Y$ and X, Y, Z are the Pauli matrices. Specifically, we define

$$\hat{U}(\theta) = \hat{R}_{\varphi_d}(\theta)\hat{R}_{\varphi_{d-1}}(\theta)\cdots\hat{R}_{\varphi_1}(\theta), \quad \theta \in \mathbb{R}$$

and ask which matrix-valued functions $\hat{U}(\theta)$ can be achieved by varying phases $\vec{\varphi} = (\varphi_1, \dots, \varphi_d)$. $\hat{R}_\varphi(\theta)$ plays the role of the combined *signal* and *signal-processing* operation due to the identity $\hat{R}_\varphi(\theta) = \mathrm{e}^{-i\frac{\varphi}{2}Z}\,\mathrm{e}^{-i\frac{\theta}{2}X}\,\mathrm{e}^{i\frac{\varphi}{2}Z}$ together with $\mathrm{e}^{-i\frac{\theta}{2}X} = W(\cos(\frac{\theta}{2}))$ in case $\sin(\frac{\theta}{2}) \le 0$ and $\mathrm{e}^{-i\frac{\theta}{2}X} = Z\,W(\cos(\frac{\theta}{2}))\,Z$ in case $\sin(\frac{\theta}{2}) \ge 0$, where $W(a) = \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix}$ for $a \in [-1, 1]$ is the *signal* operator defined in the lecture.

Taking $x \equiv \cos(\theta/2)$ and $y \equiv \sin(\theta/2)$, one can show that

$$\hat{U}(\theta) = \begin{cases} A(x)I + iB(x)Z + iC(y)X + iD(y)Y, & d \text{ odd}, \\ A(x)I + iB(x)Z + ixC(y)X + ixD(y)Y, & d \text{ even}, \end{cases} \tag{2}$$

where $A(x), B(x), C(y), D(y)$ are real-valued polynomials of degree at most d.

We say that a tuple of polynomials (A, B, C, D) is *achievable* if there exists $\vec{\varphi} \in \mathbb{R}^d$ such that $\hat{U}(\theta) = \hat{R}_{\varphi_d}(\theta)\hat{R}_{\varphi_{d-1}}(\theta)\cdots\hat{R}_{\varphi_1}(\theta)$ has the form of (2). Given this definition, one can show that a tuple of polynomials (A, B, C, D) of degree at most d is achievable if and only if all the following are true:

- A, B, C, D are real.
- $A(1) = 1$ or $B(1) = 0$.
- $\begin{cases} A, B, C, D \text{ are odd}, & d \text{ odd}, \\ A, B \text{ are even and } C, D \text{ are odd}, & d \text{ even}. \end{cases}$
- $1 = \begin{cases} A(x)^2 + B(x)^2 + C(y)^2 + D(y)^2, & d \text{ odd}, \\ A(x)^2 + B(x)^2 + x^2C(y)^2 + x^2D(y)^2, & d \text{ even}. \end{cases}$

In the following, we derive some basic results that precede the proof of the previous statement:

(a) Verify that

$$\sigma_{\varphi_1}\sigma_{\varphi_2}\cdots\sigma_{\varphi_j} = \exp\left(iZ\sum_{k=1}^{j}(-1)^k\varphi_k\right)X^j$$

for all $\varphi_1, \dots, \varphi_j \in \mathbb{R}$.

(b) Show that

$$\hat{U}(\theta) = \sum_{j=0}^{d}(-i)^j \sin^j\left(\frac{\theta}{2}\right)\cos^{d-j}\left(\frac{\theta}{2}\right)\hat{\Phi}_d^j(\vec{\varphi}), \tag{3}$$

where

$$\hat{\Phi}_d^j(\vec{\varphi}) = \left(\mathrm{Re}[\Phi_d^j(\vec{\varphi})]I - i\,\mathrm{Im}[\Phi_d^j(\vec{\varphi})]Z\right)X^j,$$

$$\Phi_d^j(\vec{\varphi}) = \sum_{1\le h_1<h_2<\cdots<h_j\le d}\exp\left(-i\sum_{k=1}^{j}(-1)^k\varphi_{h_k}\right).$$

(c) Show that the recursive relation

$$\Phi_d^j = \Phi_{d-1}^j + \Phi_{d-1}^{j-1}\,\mathrm{e}^{i(-1)^{j+1}\varphi_d}$$

holds for all $d, j \ge 2$.

(d) Show that Eq. (2) holds.

[1] G. H. Low, T. J. Yoder, I. L. Chuang: *Methodology of resonant equiangular composite quantum gates.* PRX 6, 041067 (2016), and G. H. Low, T. J. Yoder, I. L. Chuang: *Optimal arbitrarily accurate composite pulse sequences.* Phys. Rev. A 89, 022341 (2014)

Exercise 9.1 (Polynomial representation of the quantum signal processing circuit)
Recall that the QSP sequence for $a \in [-1,1]$ and "signal-processing" phases $\vec{\varphi} = (\varphi_0, \dots, \varphi_d) \in \mathbb{R}^{d+1}$ is defined as

$$U_{\vec{\varphi}} = \mathrm{e}^{i\varphi_0 Z} \prod_{k=1}^{d} W(a) \mathrm{e}^{i\varphi_k Z} \quad \text{with} \quad W(a) = \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix}. \tag{4}$$

As stated in the QSP theorem, $U_{\vec{\varphi}}$ is of the form

$$U_{\vec{\varphi}} = \begin{pmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{pmatrix} \tag{5}$$

with P a polynomial of degree $\deg(P) \le d$ and parity $d \bmod 2$, and Q a polynomial of degree $\deg(Q) \le d-1$ and parity $(d-1) \bmod 2$. Here parity 0 (even) means that a polynomial $f(x)$ contains only even powers of x, and parity 1 (odd) that $f(x)$ contains only odd powers of x.

(a) Evaluate (with pen and paper) $U_{(\varphi_0,\varphi_1)}$ and $U_{(\varphi_0,\varphi_1,\varphi_2)}$, and identify the corresponding polynomials P and Q in both cases.

(b) Based on (a), verify that $\langle 0| U_{(0,0,0)} |0\rangle = T_2(a)$, with $T_2(x) = 2x^2 - 1$ the order-two Chebyshev polynomial of the first kind.

(c) Prove that $U_{\vec{\varphi}}$ is indeed of the form (5), with the respective degree and parity of P and Q as stated above.
Hint: Use induction, starting with $d = 0$.

(d) In many cases it turns out to be useful to consider $\langle +| U_{\vec{\varphi}} |+\rangle$ as "output" of the signal processing. Show that $\langle +| U_{\vec{\varphi}} |+\rangle = \mathrm{Re}(P(a)) + i\mathrm{Re}(Q(a))\sqrt{1-a^2}$.

Exercise 9.2 (Numerical simulation and examples of quantum signal processing)

(a) Write a Python/NumPy program which evaluates $U_{\vec{\varphi}}$ defined in Eq. (4) as function of a and $\vec{\varphi}$. To test your implementation, insert the BB1 angles $\vec{\varphi} = (\pi/2, -\eta, 2\eta, 0, -2\eta, \eta)$, $\eta = \frac{1}{2}\arccos(-1/4) \approx 0.911738$, and compare the top-left entry $\langle 0| U_{\vec{\varphi}} |0\rangle$ with the analytic formula

$$P_{\mathsf{BB1}}(a) = -\frac{1}{8} a \left(\left(\sqrt{15} - 15i\right) - 2\left(\sqrt{15} - 5i\right)a^2 + \left(\sqrt{15} - 3i\right)a^4 \right)$$

at several (randomly chosen) values of $a \in [-1,1]$.

Remark: Plotting $|P_{\mathsf{BB1}}(\cos(\frac{\theta}{2}))|^2$ as function of $\theta \in [-\pi, \pi]$ reproduces the curve shown in the lecture.

(b) A central feature of QSP is the capability to approximate quite general functions. As demonstration, we consider here cosine and sine, relevant for the quantum time evolution operator e^{-iHt}. The following phase angles are taken from Appendix D.4[2] for approximating $\frac{1}{2}\cos(at)$ and $\frac{1}{2}\sin(at)$ at $t = 10$:[3]

$\vec{\varphi}_{\cos} =$
```
(-1.70932079, -0.05312746,  2.12066859, -0.83307065,
 -0.50074601,  0.40728859,  0.32838472,  0.9142489,
 -2.81320793,  0.40728859, -0.50074601,  2.30852201,
 -1.02092406, -0.05312746,  3.00306819)
```

and

$\vec{\varphi}_{\sin} =$
```
(-1.63276817,  0.20550406, -0.84198335,  0.39732059,
 -0.26820613,  2.41324245,  0.04662674, -2.02847501,
  1.11311765,  0.04662674, -0.72835021, -0.26820613,
  0.39732059, -0.84198335,  0.20550406, -0.06197184)
```

Use your implementation from (a) to plot $\mathrm{Re}(\langle +| U_{\vec{\varphi}} |+\rangle)$ at these phases as function of a, and compare with the curves of $\frac{1}{2}\cos(at)$ and $\frac{1}{2}\sin(at)$. Visually, the functions should be indiscernible. Also plot $\mathrm{Im}(\langle +| U_{\vec{\varphi}} |+\rangle)$, which should be close to zero for all a.

[2] J. M. Martyn, Z. M. Rossi, A. K. Tan, I. L. Chuang: *Grand unification of quantum algorithms.* PRX Quantum 2, 040203 (2021)
[3] The description in the paper mentions $t = 5$, but this seems to be an error.

Exercise 9.1 (Polynomial representation of the quantum signal processing circuit)
Recall that the QSP sequence for $a \in [-1,1]$ and "signal-processing" phases $\vec{\varphi} = (\varphi_0, \ldots, \varphi_d) \in \mathbb{R}^{d+1}$ is defined as

$$U_{\vec{\varphi}} = e^{i\varphi_0 Z} \prod_{k=1}^{d} W(a) e^{i\varphi_k Z} \quad \text{with} \quad W(a) = \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix}. \tag{4}$$

As stated in the QSP theorem, $U_{\vec{\varphi}}$ is of the form

$$U_{\vec{\varphi}} = \begin{pmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{pmatrix} \tag{5}$$

with P a polynomial of degree $\deg(P) \le d$ and parity $d \bmod 2$, and Q a polynomial of degree $\deg(Q) \le d-1$ and parity $(d-1) \bmod 2$. Here parity 0 (even) means that a polynomial $f(x)$ contains only even powers of x, and parity 1 (odd) that $f(x)$ contains only odd powers of x.

(a) Evaluate (with pen and paper) $U_{(\varphi_0,\varphi_1)}$ and $U_{(\varphi_0,\varphi_1,\varphi_2)}$, and identify the corresponding polynomials P and Q in both cases.

(a)

$$U_{(\varphi_0,\varphi_1)} \Rightarrow U_{\vec{\varphi}} = e^{i\varphi_0 Z} \cdot \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix} e^{i\varphi_1 Z} \Rightarrow P = e^{i(\varphi_0+\varphi_1)\cdot Z} \cdot a$$
$$Q = e^{i(\varphi_0+\varphi_1)\cdot Z}$$

$$U_{\varphi_0\varphi_1\varphi_2} = e^{i\varphi_0 Z} \cdot e^{i\varphi_1 Z} \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix} e^{i\varphi_2 Z} \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix} =$$

$$= e^{i(\varphi_0+\varphi_1+\varphi_2)Z} \cdot \begin{pmatrix} a^2-1+a^2 & 2ai\sqrt{1-a^2} \\ 2i\sqrt{1-a^2} & -1+a^2+a^2 \end{pmatrix}$$

$P = 2e^{i(\varphi_0+\varphi_1+\varphi_2)Z} \cdot a^2 - 1$

$Q = 2a$

(b)

(b) Based on (a), verify that $\langle 0 | U_{(0,0,0)} | 0 \rangle = T_2(a)$, with $T_2(x) = \underline{2x^2 - 1}$ the order-two Chebyshev polynomial of the first kind.

$$U_{(0,0,0)} = \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix}\begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix} = \begin{pmatrix} 2a^2-1 & 2ai\sqrt{1-a^2} \\ 2ai\sqrt{1-a^2} & 2a^2-1 \end{pmatrix}$$

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \Rightarrow \langle 0 | U_{(000)} | 0 \rangle = (1\ 0)\begin{pmatrix} 2a^2-1 & 2ai\sqrt{1-a^2} \\ 2ai\sqrt{1-a^2} & 2a^2-1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 2a^2-1 & 2ai\sqrt{1-a^2} \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = 2a^2-1 \quad \square$$

(c) Prove that $U_{\vec{\varphi}}$ is indeed of the form (5), with the respective degree and parity of P and Q as stated above.
Hint: Use induction, starting with $d = 0$.

$$d = 0 \Rightarrow U_{\varphi_0} = e^{i\varphi_0 Z} = \cos(\varphi_0)\cdot\mathbb{1} + i\sin(\varphi_0)\cdot Z = \begin{pmatrix} \cos(\varphi_0)+i\sin(\varphi_0) & 0 \\ 0 & \cos(\varphi_0)-i\sin(\varphi_0) \end{pmatrix} = \begin{pmatrix} e^{i\varphi_0} & 0 \\ 0 & e^{-i\varphi_0} \end{pmatrix}$$

$\deg P = 0 \le 0$
$\deg Q = 0 \le -1$

$$d = k \Rightarrow U_{(\varphi_0,\ldots\varphi_k)} = e^{i(\varphi_0+\varphi_1+\ldots\varphi_k)Z} \cdot \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix}^k \rightarrow$$

$$d = k+1 = U_{(\varphi_0\ldots\varphi_k)} \cdot e^{i\varphi_k Z} \cdot \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix} = e^{i\varphi_k Z} \cdot \begin{pmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{pmatrix} \cdot \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix}$$

$$= e^{i\varphi_k Z} \cdot \begin{pmatrix} P(a)\cdot a - Q(a)\cdot + a^2 Q(a) & \ldots \\ iQ^*(a)\cdot a \cdot\sqrt{1-a^2} + P^*(a)\cdot i\sqrt{1-a^2} & \ldots \end{pmatrix}$$

$\underline{\text{degree}}$: $\Rightarrow \deg(\text{new } P(a)) = \max\text{-}\deg(P(a)\cdot a,\ Q(a),\ a^2\cdot Q(a))$ $\square$
$= \deg(P(a))\cdot a \le d+1$

$\deg(\text{new } Q(a)) = \deg(Q(a))\cdot \cancel{\deg} + 1 \le d$ $\square$

$k = d+1$

$\underline{\text{parity}}$: $\Rightarrow$ $P(a)\cdot a \rightarrow (d+1) \bmod 2 =$ $k \bmod 2$

$Q(a)\cdot a \rightarrow (d) \bmod 2 = (k-1) \bmod 2$ $\square$

true for $k+1$

(d) In many cases it turns out to be useful to consider $\langle +|U_{\vec{\varphi}}|+\rangle$ as "output" of the signal processing. Show that $\langle +|U_{\vec{\varphi}}|+\rangle = \mathrm{Re}(P(a)) + i\mathrm{Re}(Q(a))\sqrt{1-a^2}$.

$$|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$\langle + | U_{\vec{\varphi}} | + \rangle = \frac{1}{2} \cdot (\langle 0| + \langle 1|) \begin{pmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{pmatrix} (|0\rangle + |1\rangle) =$$

$$\langle 0|U_\varphi|0\rangle = (1 \quad 0) \begin{pmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ \ldots & \ldots \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = P(a)$$

$$\langle 0|U_\varphi|1\rangle = iQ(a) \cdot \sqrt{1-a^2}$$

$$\langle 1|U_\varphi|0\rangle = iQ^*(a)\sqrt{1-a^2}$$

$$\langle 1|U_\varphi|1\rangle = P^*(a)$$

since: $\frac{z + z^*}{2} = \mathrm{Re}(z) \Rightarrow \langle + U_\varphi | + \rangle = \mathrm{Re}(P(a)) + i\,\mathrm{Re}(Q(a)) \cdot \sqrt{1-a^2}$ $\square$

$z \in \mathbb{C}$

(c) Use the Python package pyqsp[4] to find the QSP phases corresponding to the polynomial $5x^3/4 - x/3$. Look at the "programmatic usage" section of the package documentation for instructions. In particular you should:

- Define the polynomial as follows (with adapted coefficients):

```
from numpy.polynomial.polynomial import Polynomial
# example for polynomial p(x) = -2 + 5x**2 + x**3
pcoefs = [-2., 0., 5., 1.]
poly = Polynomial(pcoefs)
```

- Compute the QSP phases via:

```
from pyqsp.angle_sequence import QuantumSignalProcessingPhases
phi = QuantumSignalProcessingPhases(poly, signal_operator="Wx", method="laurent")
```

- Compare the QSP circuit output with the actual polynomial, by using your implementation from (a). Alternatively, you can also use

```
pyqsp.response.PlotQSPResponse(phi, target=poly, signal_operator="Wx")
```

(d) Finally, we consider a QSP approximation of the sign function, defined as (for $x \in \mathbb{R}$)

$$\mathrm{sign}(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$$

The sign function plays an important role for a Grover-type unstructured search algorithm. Due to the discontinuity at 0, the sign function cannot be represented exactly as polynomial. A trick to circumvent this issue in practice (described in Ref. [2]) consists in approximating $\mathrm{erf}(kx)$ (error function) with fixed $k \in \mathbb{R}$ instead, which resembles the sign function but is continuous. A large coefficient k leads to a sharp transition from -1 to 1, i.e., a better approximation, but also a deeper QSP circuit. Choosing $k = \frac{1}{\Delta}\sqrt{2\log(\frac{2}{\pi\epsilon^2})}$ ensures that $\mathrm{erf}(kx)$ deviates from the sign function by at most ϵ in the region $[-1, -\frac{1}{2}\Delta] \cup [\frac{1}{2}\Delta, 1]$, see Fig. 6 in Ref. [2]. Since the error function is *holomorphic*, it can in turn be well approximated by polynomials.

Use the pyqsp package to find a degree 19 polynomial approximation of the sign function and corresponding QSP phase angles, using $k = 10$. Visualize the approximation as in (c). The following code generates the polynomial:

```
import pyqsp
pg = pyqsp.poly.PolySign()
# constructs polynomial approximation of 'erf(k x)'
poly_sign = pg.generate(degree=19, delta=k)
```

[4] https://github.com/ichuang/pyqsp

ex9_2

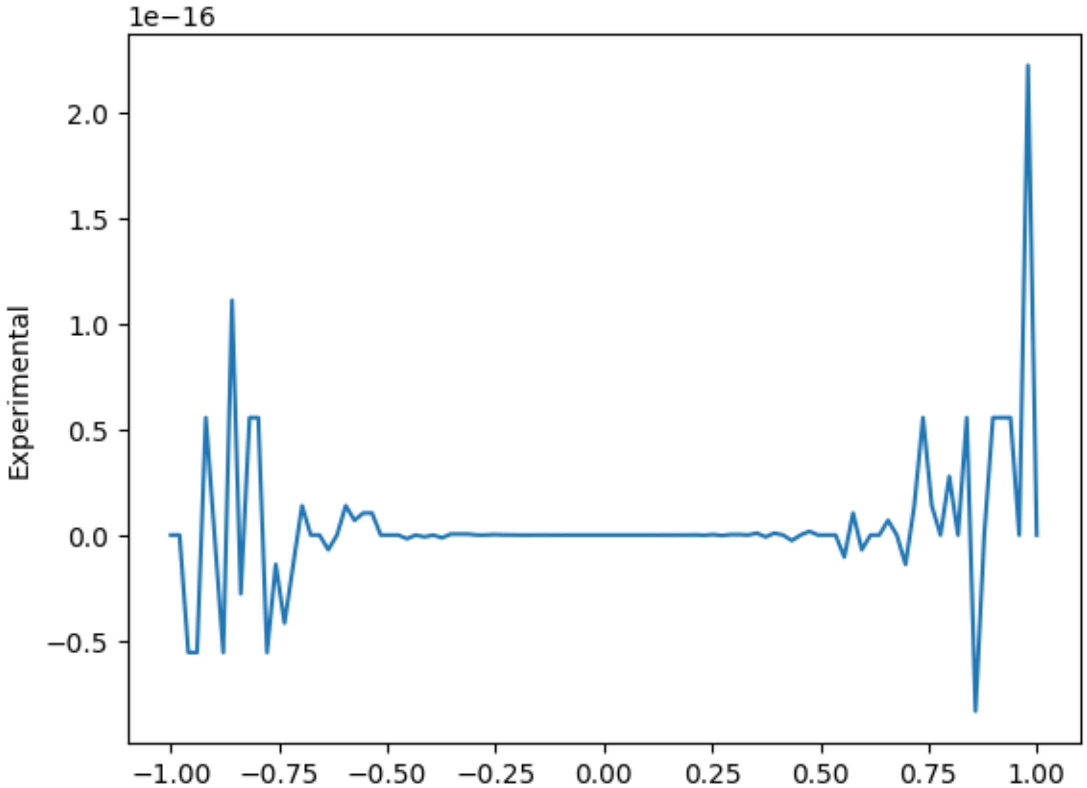
June 21, 2023

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

def W(a):
    return np.array([[a, 1j*np.sqrt(1-a**2)],[1j*np.sqrt(1-a**2), a]])

def exponential(phi):
    return np.array([[np.exp(1j*phi), 0],[0, np.exp(-1j*phi)]])

def U(a, phi):
    prod = exponential(phi[0])
    for i in phi[1:]:
        prod = prod*W(a)*exponential(i)
    return prod
```

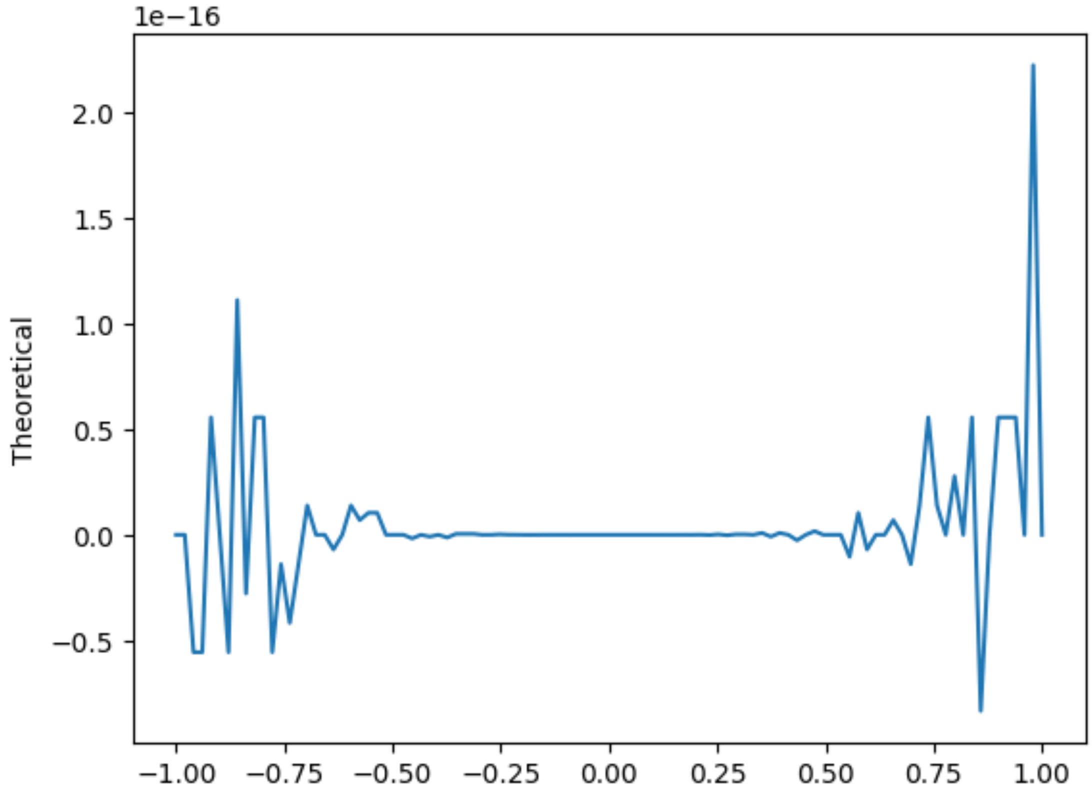
[ ]:
```
# Constants
eta = 1/2*np.arccos(-1/4)
BB1 = [np.pi/2, -eta, 2*eta, 0, -2*eta, eta]
a = np.linspace(-1,1,100)
```

[ ]:
```
# Experimental:
P_bb1 = []
for i in a:
    P_bb1.append(U(i, BB1)[0,0])
plt.figure()
plt.plot(a, P_bb1)
plt.ylabel('Experimental')
plt.show()



# U = np.exp(1j * )
```

```
/home/robi/.local/lib/python3.10/site-
packages/matplotlib/cbook/__init__.py:1335: ComplexWarning: Casting complex
values to real discards the imaginary part
  return np.asarray(x, float)
```

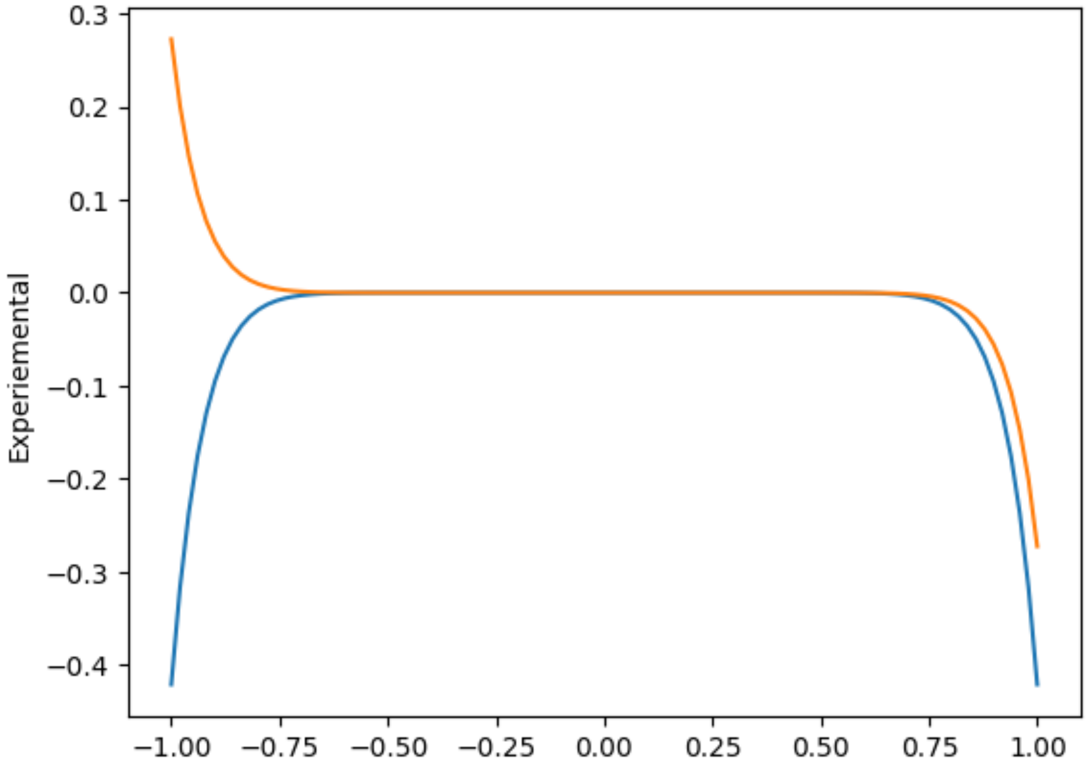


[ ]:
```
# Theoretical:
def P_bb1_teor(a):
    return -1/8*a*((np.sqrt(15) - 15j)-2*(np.sqrt(15), 5j)*a**2+(np.
↪sqrt(15)-3j)*a**4)
P_bb1_Teor=[]
for i in a:
    P_bb1_Teor.append(U(i, BB1)[0,0])
plt.figure()
plt.plot(a, P_bb1_Teor)
plt.ylabel('Theoretical')
plt.show()
```

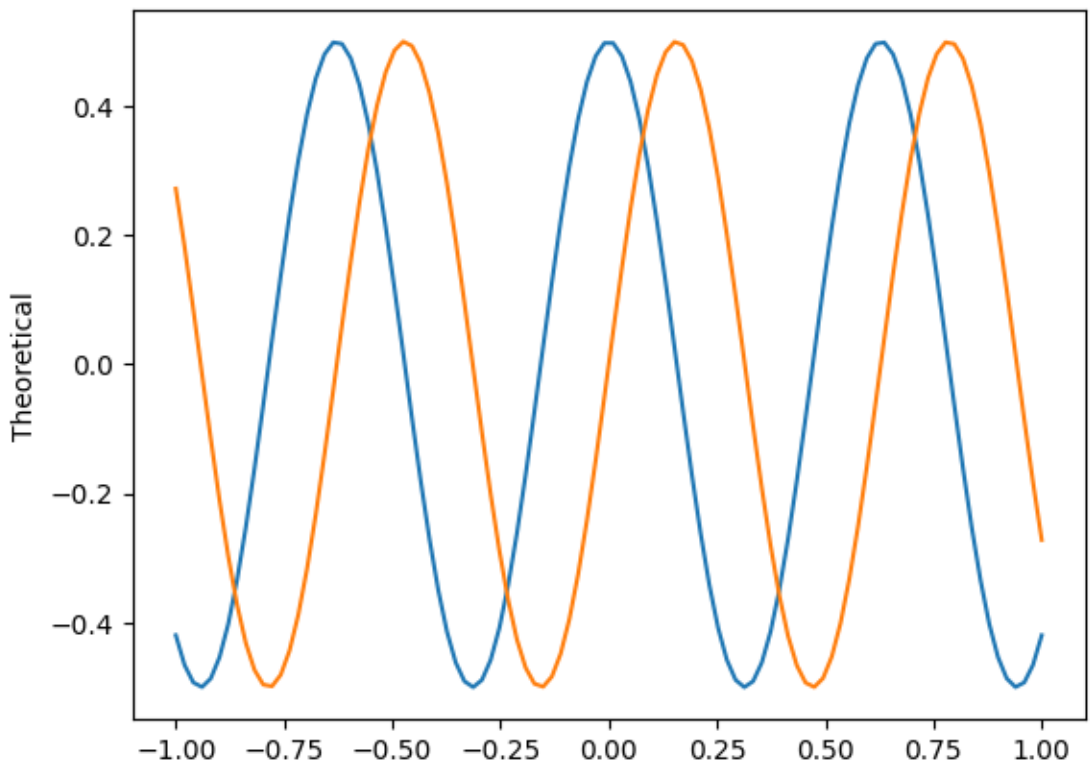


[ ]:
```
def ReU(U):
    return np.real(U)

phi_cos = (-1.70932079, -0.05312746, 2.12066859, -0.83307065, -0.50074601, 0.
 ↪40728859, 0.32838472, 0.9142489, -2.81320793, 0.40728859, -0.50074601, 2.
 ↪30852201, -1.02092406, -0.05312746, 3.00306819)
phi_sin = (-1.63276817, 0.20550406, -0.84198335, 0.39732059, -0.26820613, 2.
 ↪41324245, 0.04662674, -2.02847501, 1.11311765, 0.04662674, -0.72835021, -0.
 ↪26820613, 0.39732059, -0.84198335, 0.20550406, -0.06197184)

Re_U_cos_exp = []
Re_U_sin_exp = []

for i in a:
    Re_U_cos_exp.append(ReU(1/2 * (np.array([1.0, 1.0]) @ U(i,phi_cos)) @ np.
 ↪array([[1.0],[1.0]])))
    Re_U_sin_exp.append(ReU(1/2 * (np.array([1.0, 1.0]) @ U(i,phi_sin)) @ np.
 ↪array([[1.0],[1.0]])))

plt.figure()
plt.plot(a, Re_U_cos_exp)
```

```
plt.plot(a, Re_U_sin_exp)
plt.ylabel('Experiemental')
plt.show()
```



[ ]:
```
Re_U_cos_teor = []
Re_U_sin_teor = []

t = 10

for i in a:
    Re_U_cos_teor.append(1/2 * np.cos(i*t))
    Re_U_sin_teor.append(1/2 * np.sin(i*t))

plt.figure()
plt.plot(a, Re_U_cos_teor)
plt.plot(a, Re_U_sin_teor)
plt.ylabel('Theoretical')
plt.show()
```



[ ]:
```
from numpy.polynomial.polynomial import Polynomial
# example for polynomial p(x) = -2 + 5x**2 + x**3
pcoefs = [0., 1.0/3.0, 0., 5.0/4.0]
poly = Polynomial(pcoefs)

from pyqsp.angle_sequence import QuantumSignalProcessingPhases
phi = QuantumSignalProcessingPhases(poly, signal_operator="Wx",␣
↪method="laurent")
```

```
/home/robi/University/Adv_con_CQ/hw9/pyqsp/completion.py:172: RuntimeWarning:
invalid value encountered in sqrt
  G = LPoly(gcoefs * np.sqrt(norm / gcoefs[0]), -len(gcoefs) + 1)
```

```
---------------------------------------------------------------------------
CompletionError                           Traceback (most recent call last)
Cell In[1], line 7
      4 poly = Polynomial(pcoefs)
      6 from pyqsp.angle_sequence import QuantumSignalProcessingPhases
----> 7 phi = QuantumSignalProcessingPhases(poly, signal_operator="Wx",␣
↪method="laurent")
```

```
File ~/University/Adv_con_CQ/hw9/pyqsp/angle_sequence.py:155, in QuantumSignalProcessingPhases(poly, eps, suc, signal_operator, measurement, tolerance, method, **kwargs)
    151     poly = suc * \
    152         (poly + Polynomial([0, ] * poly.degree() + [eps / 2, ]))
    154     lcoefs = poly2laurent(poly.coef)
--> 155     lalg = completion_from_root_finding(lcoefs, coef_type="F")
    156 elif model == ("Wx", "z"):
    157     lalg = completion_from_root_finding(poly.coef, coef_type="P")

File ~/University/Adv_con_CQ/hw9/pyqsp/completion.py:244, in completion_from_root_finding(coefs, coef_type, seed, tol)
    241 success = np.max(np.abs(ncoefs - ncoefs_expected)) < tol
    243 if not success:
--> 244     raise CompletionError(
    245         "Completion Failed. Input {} = {} could not be completed".format(
    246             coef_type, coefs))
    248 return LAlg(ipoly, xpoly)

CompletionError: Completion Failed. Input F = [0.15624062 0.63537187 0.63537187 0.15624062] could not be completed
```