3.4.1 Hadamard Test

In a previous section we discussed the *Swap Test* to measure the similarity between two unknown states $|\psi\rangle$ and $|\phi\rangle$ without having to measure these state directly. Note that we use the words *similarity* and *overlap* interchangeably. We derived that for the swap test circuit, the probability $Pr(|0\rangle)$ of measuring state $|0\rangle$ was:

$$Pr(|0\rangle) = \frac{1}{2} + \frac{1}{2}\langle\psi|\phi\rangle^2.$$

We can invert this and express the *overlap* as:

$$\langle\psi|\phi\rangle^2 = 1 - Pr(|0\rangle).$$

In this section we present another test of this nature, the *Hadamard Test*.

Both the Swap test and the Hadamard test can be visualized with an analogy using real-valued vectors. The numbers come out differently, but the principle is the same. Think about how we compute the inner product of the sum $(\vec{a} + \vec{b})$ of two *normalized*, real-valued vectors $\vec{a}$ and $\vec{b}$ (they have to be normalized, else the math doesn't work out):

$$\begin{aligned}
(\vec{a}+\vec{b})^T(\vec{a}+\vec{b}) &= \sum_i (a_i + b_i)^2 \\
&= \underbrace{\sum_i a_i^2}_{=1} + \underbrace{\sum_i b_i^2}_{=1} + 2\sum_i a_i b_i \\
&= 2 + 2\vec{a}^T\vec{b}. \qquad (3.1)
\end{aligned}$$

Note the three extreme cases where $\vec{a}$ and $\vec{b}$ point in the same direction, are orthogonal, or are anti-parallel. In these three cases, Equation 3.1 yields:

$$\begin{aligned}
\text{parallel:} \quad & \vec{a} = \begin{bmatrix}1\\0\end{bmatrix} \quad \text{and} \quad \vec{b} = \begin{bmatrix}1\\0\end{bmatrix}, \quad \text{then} \quad \vec{a}^T\vec{b} = 1. \\
\text{orthogonal:} \quad & \vec{a} = \begin{bmatrix}0\\1\end{bmatrix} \quad \text{and} \quad \vec{b} = \begin{bmatrix}1\\0\end{bmatrix}, \quad \text{then} \quad \vec{a}^T\vec{b} = 0. \\
\text{anit-parallel:} \quad & \vec{a} = \begin{bmatrix}-1\\0\end{bmatrix} \quad \text{and} \quad \vec{b} = \begin{bmatrix}1\\0\end{bmatrix}, \quad \text{then} \quad \vec{a}^T\vec{b} = -1.
\end{aligned}$$

Now let's apply this principle for the Hadamard test. Remember how the Swap test used two quantum registers to hold the states ψ and ϕ? The Hadamard test is different, it uses only one quantum register which will hold the superposition of the two states $|a\rangle$ and $|b\rangle$ for which we want to determine overlap. Hence, as a precondition, we need to prepare this initial state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle|a\rangle + |1\rangle|b\rangle\big). \qquad (3.2)$$

How can we generate such a state? First, let's see how the partial expressions look as state vectors:

$$|0\rangle|a\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}a_0\\a_1\end{bmatrix} = \begin{bmatrix}a_0\\a_1\\0\\0\end{bmatrix},$$

and

$$|1\rangle|b\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}b_0\\b_1\end{bmatrix} = \begin{bmatrix}0\\0\\b_0\\b_1\end{bmatrix}.$$

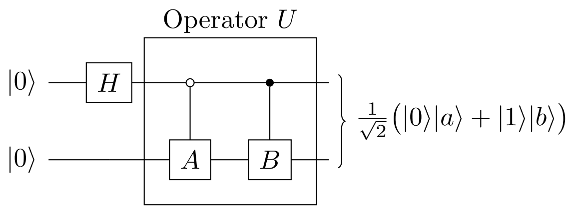
As a vector, state $|\psi\rangle$ in Equation 3.2 would be:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle|a\rangle + |1\rangle|b\rangle\big) = \frac{1}{\sqrt{2}}\begin{bmatrix}a_0\\a_1\\b_0\\b_1\end{bmatrix}.$$

We need operators A and B that produce the states $|a\rangle$ and $|b\rangle$ when applied to state $|0\rangle$. Note that A and B have to be a unitary and that we name the matrix element in a way to produce the desired output vectors:

$$\begin{aligned} A\,|0\rangle = A\begin{bmatrix}1\\0\end{bmatrix} &= \begin{bmatrix}a_0 & a_2\\a_1 & a_3\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}a_0\\a_1\end{bmatrix} = |a\rangle\,,\\ B\,|0\rangle = B\begin{bmatrix}1\\0\end{bmatrix} &= \begin{bmatrix}b_0 & b_2\\b_1 & b_3`\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}b_0\\b_1\end{bmatrix} = |b\rangle\,. \end{aligned}$$

To construct the circuit, we use qubit 0 as an ancilla and qubit 1 should hold the superposition of states a and b. We can use an initial Hadamard gate to produce an equal superposition of $|0\rangle$ and $|1\rangle$ with which we control the gates A and B on qubit 1, as shown in this circuit:



Note that in the literature the two controlled operators A and B are often referred to as a combined single operator U. Let's verify this in code! First we create random unitary operators A and B and apply them to state $|0\rangle$ to extract the relevant state components a_0, a_1, b_0, and b_1:

```
def make_rand_operator():
  """Make a unitary operator U, derive u0, u1."""

  U = ops.Operator(unitary_group.rvs(2))
  if not U.is_unitary():
    raise AssertionError('Error: Generated non-unitary operator')
  psi = U(state.bitstring(0))
  u0 = psi[0]
  u1 = psi[1]
  return (U, u0, u1)

def hadamard_test():
  """Perform Hadamard Test."""

  A, a0, a1 = make_rand_operator()
  B, b0, b1 = make_rand_operator()
```

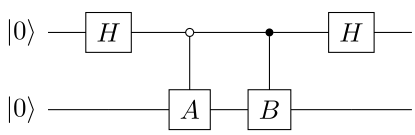
With these parameters, we can construct the state in two different ways. First we compute it explicitly, following Equation 3.2:

```
  # Construct the desired end state psi as an explicit expression.
  #    psi = 1/sqrt(2)(|0>|a> + |1>|b>)
  psi = (1 / cmath.sqrt(2) *
         (state.bitstring(0) * state.State([a0, a1]) +
          state.bitstring(1) * state.State([b0, b1])))
```

To compare, we construct the state with a circuit and confirm that the result matches the closed form above:

```
  # Let's see how to make this state with a circuit.
  qc = circuit.qc('Hadamard test - initial state construction.')
  qc.reg(2, 0)
  qc.h(0)
  qc.applyc(A, [0], 1)  # Controlled-by-0
  qc.applyc(B, 0, 1)    # Controlled-by-1

  # The two states should be identical!
  if not np.allclose(qc.psi, psi):
      raise AssertionError('Incorrect result')
```

Now let's add another Hadamard gate to the ancilla qubit 0:



This changes the state to:

$$\begin{aligned}\frac{1}{\sqrt{2}}H\big(|0\rangle|a\rangle + |1\rangle|b\rangle\big) &= \frac{1}{\sqrt{2}}\big(H|0\rangle|a\rangle + H|1\rangle|b\rangle\big)\\ &= \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\big(|0\rangle|a\rangle + |1\rangle|a\rangle + |0\rangle|b\rangle - |1\rangle|b\rangle\big)\\ &= \frac{1}{2}\,|0\rangle\,\big(|a\rangle + |b\rangle\big) + \frac{1}{2}\,|1\rangle\,\big(|a\rangle - |b\rangle\big).\end{aligned}$$

The probability of measuring state $|0\rangle$ in the first qubit is the norm squared of the probability amplitude:

$$\begin{aligned}\Big|\frac{1}{2}\big(|a\rangle + |b\rangle\big)\Big|^2 &= \frac{1}{2}\big(\langle a| + \langle b|\big)\frac{1}{2}\big(|a\rangle + |b\rangle\big)\\ &= \frac{1}{4}\big(\underbrace{\langle a|a\rangle}_{=1} + \langle a|b\rangle + \langle b|a\rangle + \underbrace{\langle b|b\rangle}_{=1}\big)\\ &= \frac{1}{4}\big(2 + \langle a|b\rangle + \langle a|b\rangle^*\big). \qquad (3.3)\end{aligned}$$

The two inner products are complex numbers. For a given complex number z, adding $z + z^* = a + ib + a - ib = 2a$. Hence we can write Equation 3.3, the probability of measuring $|0\rangle$ on qubit 0, as:

$$Pr\big(|0\rangle\big) = \frac{1}{2} + \frac{1}{2}\mathrm{Re}\big(\langle a|b\rangle\big)$$

or, alternatively:

$$2Pr\big(|0\rangle\big) - 1 = \mathrm{Re}\big(\langle a|b\rangle\big)$$

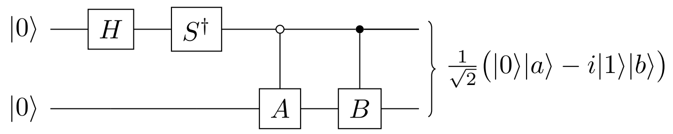
We can quickly verify this in code as well:

```
# Now let's apply a final Hadamard to ancilla.
qc.h(0)

# At this point, this inner product estimation should hold:
#   P(|0>) = 1/2 + 1/2 Re(<a|b>)
# Or
#   2 * P(|0>) - 1 = Re(<a|b>)
dot = np.dot(np.array([a0, a1]).conj(), np.array([b0, b1]))
p0 = qc.psi.prob(0, 0) + qc.psi.prob(0, 1)
if not np.allclose(2 * p0 - 1, dot.real, atol = 1e-6):
  raise AssertionError('Incorrect inner product estimation')
```

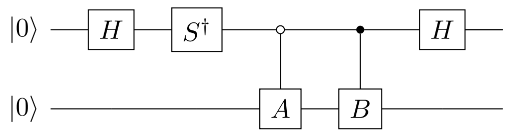
Can we obtain an estimate for the imaginary part of the inner product as well? Yes we can. For this, we start with a slightly modified initial state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle|a\rangle - i|1\rangle|b\rangle\big).$$

The construction is similar to the one above, but we have to apply a factor of $-i$ to the $|1\rangle$ part of the state by adding an $S^\dagger$ gate right after the initial Hadamard gate:



Similar to above, we add a final Hadamard gate to the ancilla:



which changes the state to:

$$\begin{aligned}
\frac{1}{\sqrt{2}} H\big(|0\rangle|a\rangle - i|1\rangle|b\rangle\big) &= \frac{1}{\sqrt{2}}\big(H|0\rangle|a\rangle - Hi|1\rangle|b\rangle\big) \\
&= \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\big(|0\rangle|a\rangle + |1\rangle|a\rangle - i|0\rangle|b\rangle + i|1\rangle|b\rangle\big) \\
&= \frac{1}{2}\,|0\rangle\,\big(|a\rangle - i\,|b\rangle\big) + \frac{1}{2}\,|1\rangle\,\big(|a\rangle + i\,|b\rangle\big).
\end{aligned}$$

The probability of measuring state $|0\rangle$ on the ancilla is, again, the norm squared of the probability amplitude:

$$\begin{aligned}
\Big|\frac{1}{2}\big(|a\rangle - i\,|b\rangle\big)\Big|^2 &= \frac{1}{2}\big(\langle a| + i\,\langle b|\big)\frac{1}{2}\big(|a\rangle - i\,|b\rangle\big) \\
&= \frac{1}{4}\big(\underbrace{\langle a|a\rangle}_{=1} - i\langle a|b\rangle + i\langle b|a\rangle + \underbrace{\langle b|b\rangle}_{=1}\big) \\
&= \frac{1}{4}\big(2 - i\langle a|b\rangle + i\langle a|b\rangle^*\big). \qquad (3.4)
\end{aligned}$$

The inner products are complex numbers. For a complex number $z = a + ib$, $z^* = a - ib$ and these relations hold:

$$\begin{aligned}
-iz &= -i(a+ib) = -ia + b, \\
iz^* &= i(a-ib) = ia + b, \\
\Rightarrow -iz + iz^* &= -ia + b + ia + b \\
&= 2b \\
&= 2\,\mathrm{Im}(z).
\end{aligned}$$

Using this in Equation 3.4, we obtain the final result:

$$Pr\big(|0\rangle\big) = \frac{1}{2} + \frac{1}{2}\,\mathrm{Im}\big(\langle a|b\rangle\big).$$

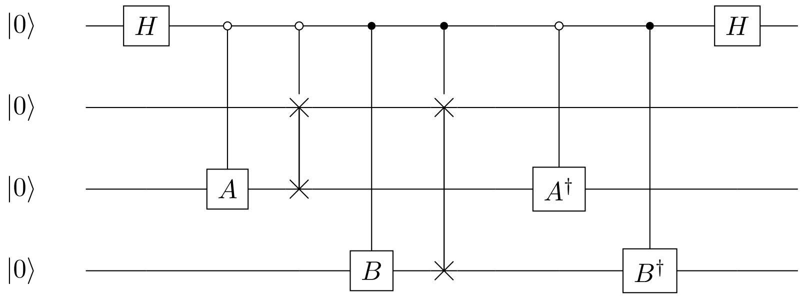
or, alternatively:

$$2Pr\big(|0\rangle\big) - 1 = \mathrm{Im}\big(\langle a|b\rangle\big).$$

We can confirm this in code as well:

```
# Now let's try the same to get to the imaginary parts.
#    psi = 1/sqrt(2)(|0>|a> - i|1>|b>)
psi = (1 / cmath.sqrt(2) *
       (state.bitstring(0) * state.State([a0, a1]) -
        1.0j * state.bitstring(1) * state.State([b0, b1])))

# Let's see how to make this state with a circuit.
qc = circuit.qc('Hadamard test - initial state construction.')
qc.reg(2, 0)
qc.h(0)
qc.sdag(0)             # <- this gate is new.
qc.applyc(A, [0], 1)  # Controlled-by-0
qc.applyc(B, 0, 1)    # Controlled-by-1

# The two states should be identical!
if not np.allclose(qc.psi, psi):
    raise AssertionError('Incorrect result')

# Now let's apply a final Hadamard to ancilla.
qc.h(0)

# At this point, this inner product estimation should hold:
#  P(|0>) = 1/2 + 1/2 Im(<a|b>)
# Or
#  2 * P(|0>) - 1 = Im(<a|b>)
dot = np.dot(np.array([a0, a1]).conj(), np.array([b0, b1]))
p0 = qc.psi.prob(0, 0) + qc.psi.prob(0, 1)
if not np.allclose(2 * p0 - 1, dot.imag, atol = 1e-6):
  raise AssertionError('Incorrect inner product estimation')
```

In cases where the construction of A and B is more complex there is the potential that the end state is entangled with the construction of A and B. In such cases, we should introduce an explicit result quantum register. After computation of A and B we superimpose the results onto this register, before uncomputing the construction of A and B, as described in the section on uncomputation. For example:



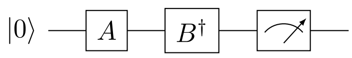
3.4.2 Inversion Test

In this chapter we will discuss the *inversion test*, a third way to approximate the similarity between states via estimating their scalar product. So far we have learned about the swap test, which utilized a register for each input $|a\rangle$ and $|b\rangle$ together with an ancilla. We also learned about the Hadamard test, which uses the ancilla but just one register, assuming there are operators A and B to construct the states $|a\rangle$ and $|b\rangle$.

The inversion test takes this one step further. It no longer needs an ancilla, just one quantum register, but it needs the ability to construct $B^\dagger$. We again assume operators A and B produce states $|a\rangle$ and $|b\rangle$, with:

$$
\begin{aligned}
A\,|0\rangle = A\begin{bmatrix}1\\0\end{bmatrix} &= \begin{bmatrix}a_0 & a_2\\ a_1 & a_3\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}a_0\\a_1\end{bmatrix} = |a\rangle\,,\\
B\,|0\rangle = B\begin{bmatrix}1\\0\end{bmatrix} &= \begin{bmatrix}b_0 & b_2\\ b_1 & b_3`\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}b_0\\b_1\end{bmatrix} = |b\rangle\,,
\end{aligned}
$$

and construct this simple circuit:



The expectation value of the projective measurement $M = |0\rangle\langle 0|$ is given by

$$
\begin{aligned}
\big(\langle 0|A^\dagger B|0\rangle\big)\big(\langle 0|B^\dagger A|0\rangle\big) &= \langle 0|A^\dagger B|0\rangle\langle 0|B^\dagger A|0\rangle\\
&= |\langle 0|B^\dagger A|0\rangle|^2\\
&= |\underbrace{\langle 0|B^\dagger}_{=\langle b|}\underbrace{A|0\rangle}_{=|a\rangle}|^2\\
&= |\langle b|a\rangle|^2\\
&= \langle a|b\rangle\langle b|a\rangle = \langle b|a\rangle\langle a|b\rangle\\
&= |\langle a|b\rangle|^2.
\end{aligned}
$$

Note that the norm of the inner product is symmetric. In code, we reuse the mechanism introduced in the section on the Hadamard test to construct random unitaries A and B with function `make_rand_operator()`. The inversion test itself is then straightforward:

```
def inversion_test():
  """Perform Inversion Test."""

  # The inversion test allows keeping the number of qubits to a minimum
  # when trying to determine the overlap between two states. However, it
  # requires a precise way to generate states a and b, as well as the
  # adjoint for one of them.
  #
  # If we have operators A and B (similar to the Hadamard Test),
  # to determine the overlap between a and be (<a|b>), we run:
  #     B_adjoint A |0>
  # and determine the probability p0 of measuring |0>. p0 is an
  # a precise estimate for <a|b>.

  A, a0, a1 = make_rand_operator()
  B, b0, b1 = make_rand_operator()

  # For the inversion test, we will need B^\dagger:
  Bdag = B.adjoint()

  # Compute the dot product <a|b>:
  dot = np.dot(np.array([a0, a1]).conj(), np.array([b0, b1]))

  # Here is the inversion test. We run B^\dagger A |0> and find
  # the probability of measuring |0>:
  qc = circuit.qc('Hadamard test - initial state construction.')
  qc.reg(1, 0)
  qc.apply1(A, 0)
  qc.apply1(Bdag, 0)

  # The probability amplitude of measuring |0> should be the
  # same value as the dot product.
  p0, _ = qc.measure_bit(0, 0)
  if not np.allclose(dot.conj() * dot, p0):
    raise AssertionError('Incorrect inner product estimation')
```

Bibliography