Z-X decomposition for a single-qubit operation

Suppose U is a unitary operation on a single-qubit. A fundamental theorem in quantum information science states that there exist real numbers α, γ, θ and ϕ such that

$$\begin{aligned} U(\alpha,\gamma,\theta,\phi) &= e^{i\alpha} R_z(\gamma) R_x(\theta) R_z(\phi) \\ &= e^{i\alpha} \begin{bmatrix} e^{-i\gamma/2} & 0 \\ 0 & e^{i\gamma/2} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix} \\ U(\alpha,\gamma,\theta,\phi) &= e^{i\alpha} \begin{bmatrix} e^{-i(\gamma+\phi)/2}\cos\frac{\theta}{2} & -ie^{i(\phi-\gamma)/2}\sin\frac{\theta}{2} \\ -ie^{i(\gamma-\phi)/2}\sin\frac{\theta}{2} & e^{i(\gamma+\phi)/2}\cos\frac{\theta}{2} \end{bmatrix} \end{aligned}$$

The α parameter being a global phase, we can thus without loss of generality take

$$\alpha = \frac{\gamma+\phi}{2},$$

leading to a much simpler version of the unitary transformation:

$$U(\gamma,\theta,\phi) = \begin{bmatrix} \cos\frac{\theta}{2} & -ie^{i\phi}\sin\frac{\theta}{2} \\ -ie^{i\gamma}\sin\frac{\theta}{2} & e^{i(\gamma+\phi)}\cos\frac{\theta}{2} \end{bmatrix}$$

From the last definition, we see that a general single-qubit gate is parameterized by three variables.

Example (H-gate): It comes from the previous definition that

$$U\left(\gamma=\frac{\pi}{2},\theta=\frac{\pi}{2},\phi=\frac{\pi}{2}\right) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H$$

Following the same approach, it also comes that:

- $X = X^\dagger = U\left(\gamma=\frac{\pi}{2},\theta=\pi,\phi=\frac{\pi}{2}\right)$
- $Y = Y^\dagger = U\left(\gamma=\pi,\theta=\pi,\phi=0\right)$
- $Z = Z^\dagger = U\left(\gamma=0,\theta=0,\phi=\pi\right)$
- $R_x(\lambda) = R_x^\dagger(-\lambda) = U\left(\gamma=0,\theta=\lambda,\phi=0\right)$
- $R_y(\lambda) = R_y^\dagger(-\lambda) = U\left(\gamma=\frac{\pi}{2},\theta=\lambda,\phi=-\frac{\pi}{2}\right)$
- $R_z(\lambda) = R_z^\dagger(-\lambda) = U\left(\gamma=0,\theta=0,\phi=\lambda\right)$
- $H = H^\dagger = U\left(\gamma=\frac{\pi}{2},\theta=\frac{\pi}{2},\phi=\frac{\pi}{2}\right)$
- $S = R_z\left(\frac{\pi}{2}\right) = U\left(\gamma=0,\theta=0,\phi=\frac{\pi}{2}\right)$
- $S^\dagger = R_z\left(-\frac{\pi}{2}\right) = U\left(\gamma=0,\theta=0,\phi=-\frac{\pi}{2}\right)$

- $T = R_z\left(\frac{\pi}{4}\right) = U\left(\gamma = 0, \theta = 0, \phi = \frac{\pi}{4}\right)$

- $T^\dagger = R_z\left(-\frac{\pi}{4}\right) = U\left(\gamma = 0, \theta = 0, \phi = -\frac{\pi}{4}\right)$

Single-qubit operations as pulses

A pulse is defined as the modulation of a signal's amplitude, detuning (i.e. frequency difference with a reference frequency) and phase over a finite duration. In atomic physics, we use the following notations to refer to the aforementioned quantities:

- $\Omega(t)$: amplitude at instant t - physicists sometimes name this parameter the Rabi frequency even though it's actually an amplitude;
- $\delta(t)$: detuning at instant t - here the difference between the pulse frequency and the actual frequency of the atomic transition;
- ϕ: phase;
- τ: pulse duration.

In quantum physics, a pulse-driven transition between two-energy levels of an atom can be mapped to a spin-1/2 system through the drive Hamiltonian

$$H^D(t) = \frac{\hbar}{2}\mathbf{\Omega}\cdot\sigma$$

where $\sigma = (X, Y, Z)^T$ is the Pauli vector and $\mathbf{\Omega}(t) = (\Omega(t)\cos(\phi), -\Omega(t)\sin(\phi), -\delta(t))^T$ the rotation vector.

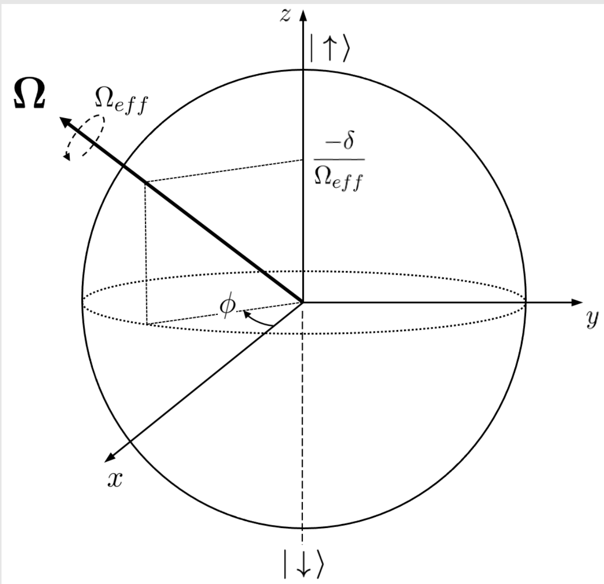

Figure 1: Representation of the drive Hamiltonian's dynamics as a rotation in the Bloch sphere.

In the Bloch sphere representation, for each instant t, this Hamiltonian describes a rotation around the axis $\mathbf{\Omega}$ with angular velocity $\Omega_{eff} = |\mathbf{\Omega}| = \sqrt{\Omega^2 + \delta^2}$, as illustrated on the figure above.

In terms of unitary evolution, the time evolution of this system is dictated by the operator

$$U(\mathbf{\Omega}, \tau) = T \exp\left[-\frac{i}{2}\int_0^\tau \mathbf{\Omega}\cdot\sigma\, dt\right],$$

where T denotes the time-ordering operator. The unitary U describes a rotation around the time-dependend axis $\mathbf{\Omega}(t)$. For a resonant pulse (i.e. $\delta = 0$) of phase ϕ, we get a rotation angle

$$\theta = \int_0^\tau \Omega(t)\, dt$$

around the fixed axis

$$\hat{\mathbf{n}}(\phi) = (\cos(\phi), -\sin(\phi), 0)^T,$$

situated on the equator of the Bloch sphere.

In linear algebra, if $\hat{\mathbf{n}} = (n_x, n_y, n_z)^T$ is a real unit vector in three dimensions, then rotating around $\hat{\mathbf{n}}(\phi)$ of angle θ actually corresponds to implementing the following rotation matrix

$$R_{\hat{\mathbf{n}}(\phi)}(\theta) \equiv \exp\left(-i\theta\, \hat{\mathbf{n}}\cdot\sigma/2\right) = \cos\frac{\theta}{2} I - i\sin\frac{\theta}{2}\, \hat{\mathbf{n}}\cdot\sigma$$

Thus, if $\hat{\mathbf{n}}(\phi) = (\cos(\phi), -\sin(\phi), 0)^T$ we get

$$\begin{aligned} R_{\hat{\mathbf{n}}(\phi)}(\theta) &= e^{-i\frac{\theta}{2}(\cos(\phi)X - \sin(\phi)Y)} \\ &= e^{i\frac{\phi}{2}Z} e^{-i\frac{\theta}{2}X} e^{-i\frac{\phi}{2}Z} \\ R_{\hat{\mathbf{n}}(\phi)}(\theta) &= R_z(-\phi)R_x(\theta)R_z(\phi) \end{aligned}$$

which, as its decomposition shows, can be thought of as a rotation around the Bloch sphere's x-axis, conjugated by z-rotations (i.e. phase gates). By following this gate with another z-rotation (which can be achieved virtually through a shift in the phase reference frame), we can then construct any arbitrary single-qubit gate,

$$U(\gamma, \theta, \phi) = R_z(\gamma + \phi)R_{\hat{\mathbf{n}}(\phi)}(\theta) = R_z(\gamma)R_x(\theta)R_z(\phi)$$

which relies solely on resonant pulses and phase reference frame changes.

Single-qubit operations with Pulser

Merging together considerations from sections 1 and 2, common single-qubit operations can be decomposed the following way:

- $X = X^\dagger = R_z(\pi)R_{\hat{\mathbf{n}}\left(\frac{\pi}{2}\right)}(\pi)$
- $Y = Y^\dagger = R_z(\pi)R_{\hat{\mathbf{n}}(0)}(\pi)$
- $Z = Z^\dagger = R_z(\pi)$
- $R_x(\lambda) = R_x^\dagger(-\lambda) = R_{\hat{\mathbf{n}}(0)}(\lambda)$
- $R_y(\lambda) = R_y^\dagger(-\lambda) = R_{\hat{\mathbf{n}}\left(-\frac{\pi}{2}\right)}(\lambda)$
- $R_z(\lambda) = R_z^\dagger(-\lambda) = R_z(\lambda)$

- $H = H^\dagger = R_z(\pi) R_{\hat{\mathbf{n}}\left(\frac{\pi}{2}\right)}\left(\frac{\pi}{2}\right)$

- $S = R_z\left(\frac{\pi}{2}\right)$

- $S^\dagger = R_z\left(-\frac{\pi}{2}\right)$

- $T = R_z\left(\frac{\pi}{4}\right)$

- $T^\dagger = R_z\left(-\frac{\pi}{4}\right)$

Here it is important to distinguish gates with non-zero and zero θ (pulse area). As the later (Z, R_z, S, $S^\dagger$, T, $T^\dagger$) would correspond to pulses with no waveform, in practice they are achieved virtually through the following command line:

```
# Simulating a Rz(rot_angle) gate
Sequence.phase_shift(rot_angle, *targets, basis='digital')
```

In this case, no pulse is sent to the atomic ensemble. Instead, Pulser makes a reference frame change (i.e. a passive rotation) to simulate the action of these gates.

Example (T-gate):

```
# Intializing a qubit named "q0" at location (0, 0)
import numpy as np
from pulser import Sequence, Register
from pulser.devices import MockDevice
reg = Register({"q0": (0, 0)})
device = MockDevice
seq = Sequence(reg, device)
seq.declare_channel("ch0", "raman_local", initial_target="q0")

# Then applying a T-gate on it
seq.phase_shift(np.pi / 4, basis='digital')
```

Other gates (X, Y, R_x, R_y, H) - operations with non zero θ - can be implemented by first defining a waveform with input parameters τ (pulse duration) and θ (pulse area), then use this waveform to define a pulse object with zero detuning. Parameters of the pulse have to be declared the following order:

1. $\Omega(\tau, \theta)$ (amplitude waveform)
2. δ (detuning)
3. ϕ (phase)
4. $\gamma + \phi$ (post phase-shift)

Example (H-gate):

```python
# Intializing a qubit named "q0" at location (0, 0)
import numpy as np
from pulser import Pulse, Sequence, Register
from pulser.devices import MockDevice
reg = Register({"q0": (0, 0)})
device = MockDevice
seq = Sequence(reg, device)
seq.declare_channel("ch0", "raman_local", initial_target="q0")

# Defining a pulse reproducing the Hadamard gate with Blackman waveform
from pulser.waveforms import BlackmanWaveform

def h_pulse(pulse_duration=1000):
  half_pi_wf = BlackmanWaveform(pulse_duration, np.pi / 2)
  h_pulse = Pulse.ConstantDetuning(
      amplitude=half_pi_wf, detuning=0, phase=np.pi / 2, post_phase_shift=np.pi
      )
  return h_pulse

# Applying the Hadamard gate on qubit "q0"
seq.add(h_pulse(), "ch0")
```

Other pulses can be defined as well:

```python
# Defining a pulse reproducing the X gate
def x_pulse(pulse_duration=1000):
  pi_wf = BlackmanWaveform(pulse_duration, np.pi)
  x_pulse = Pulse.ConstantDetuning(
      pi_wf, detuning=0, phase=np.pi / 2, post_phase_shift= np.pi
      )
  return x_pulse

# Defining a pulse reproducing the Y gate
def y_pulse(pulse_duration=1000):
  pi_wf = BlackmanWaveform(pulse_duration, np.pi)
  y_pulse = Pulse.ConstantDetuning(
      pi_wf, detuning=0, phase=0, post_phase_shift=np.pi
      )
  return y_pulse

def h_pulse(pulse_duration=1000):
  half_pi_wf = BlackmanWaveform(pulse_duration, np.pi / 2)
  h_pulse = Pulse.ConstantDetuning(
      amplitude=half_pi_wf, detuning=0, phase=np.pi / 2, post_phase_shift=np.pi
      )
  return h_pulse

# Defining a pulse reproducing the Rx(rot_angle) gate
def rx_pulse(rot_angle, pulse_duration=1000):
  rot_angle_wf = BlackmanWaveform(pulse_duration, rot_angle)
  rx_pulse = Pulse.ConstantDetuning(
      rot_angle_wf, detuning=0, phase=0, post_phase_shift=0
      )
```

```
  return rx_pulse

# Defining a pulse reproducing the Rx(-rot_angle) = Rx^{dagger}(rot_angle) gate
def rx_dag_pulse(rot_angle, pulse_duration=1000):
  rot_angle_wf = BlackmanWaveform(pulse_duration, rot_angle)
  rx_dag_pulse = Pulse.ConstantDetuning(
      rot_angle_wf, detuning=0, phase=np.pi, post_phase_shift=0
      )
  return rx_dag_pulse

# Defining a pulse reproducing the Ry(rot_angle) gate
def ry_pulse(rot_angle, pulse_duration=1000):
  rot_angle_wf = BlackmanWaveform(pulse_duration, rot_angle)
  ry_pulse = Pulse.ConstantDetuning(
      rot_angle_wf, detuning=0, phase=-np.pi / 2, post_phase_shift=0
      )
  return ry_pulse

# Defining a pulse reproducing the Ry(-rot_angle) = Ry^{dagger}(rot_angle) gate
def ry_dag_pulse(rot_angle, pulse_duration=1000):
  rot_angle_wf = BlackmanWaveform(pulse_duration, rot_angle)
  ry_dag_pulse = Pulse.ConstantDetuning(
      rot_angle_wf, detuning=0, phase=np.pi / 2, post_phase_shift=0
      )
  return ry_dag_pulse
```

For you to easily access these functions, they have been reproduced in `gate_pulses.py` and are accessible through the following command line:

```
from gate_pulses import *
```

The file also contains a function to implement a custom $U(\gamma, \theta, \phi)$ gate:

```
def u3_pulse(gamma_angle, theta_angle, phi_angle, pulse_duration=1000):
  custom_wf = BlackmanWaveform(pulse_duration, theta_angle)
  u3_pulse = Pulse.ConstantDetuning(
      amplitude=custom_wf, detuning=0, phase=phi_angle,
      post_phase_shift=phi_angle + gamma_angle
      )
  return u3_pulse
```