Implementation Details

Single Qubit

Overview of qubit class located in SingleQubit.py:

```
class Qubit:

def __init__(self, vec)
"""
input: vec = [rx, ry, rz]
Blochvector entries:
rho = 0.5*(I + vec*sigma_vector)
"""
def get_matrix(self)
# outputs current density matrix rho

def _set_Bloch_vector(self, vec)
# sets new state via new Bloch vector

def get_Bloch_vector(self)
# outputs current Bloch vector
```

Single Qubit Gates

Single qubit gates located in SingleQubitGates.py:

```
def operator_sum(blochvector, list)
# performs operator sum
# with operators in list
# on density matrix defined by the Bloch vector
```

Gate function	**Matrices E_k for $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$**
`def I_gate(blochvector)`	$E_0 = I$
`def X_gate(blochvector)`	$E_0 = \sigma_1$
`def Y_gate(blochvector)`	$E_0 = \sigma_2$
`def Z_gate(blochvector)`	$E_0 = \sigma_3$
`def H_gate(blochvector)`	$E_0 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
`def S_gate(blochvector)`	$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix}$
`def T_gate(blochvector)`	$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$
`def amplitude_damping(blochvector, Gamma, t)`	$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & e^{-0.5\cdot\mathtt{Gamma}\cdot\mathtt{t}} \end{pmatrix}$ $E_1 = \begin{pmatrix} 0 & \sqrt{1 - e^{-\mathtt{Gamma}\cdot\mathtt{t}}} \\ 0 & 0 \end{pmatrix}$
`def phase_damping(blochvector, Gamma, t)`	$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & e^{-\mathtt{Gamma}\cdot\mathtt{t}} \end{pmatrix}$ $E_1 = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{1 - e^{-2\cdot\mathtt{Gamma}\cdot\mathtt{t}}} \end{pmatrix}$
`def bit_flip(blochvector, p)`	$E_0 = \sqrt{1-\mathtt{p}} \cdot I$ $E_1 = \sqrt{\mathtt{p}} \cdot \sigma_1$
`def phase_flip(blochvector, p)`	$E_0 = \sqrt{1-\mathtt{p}} \cdot I$ $E_1 = \sqrt{\mathtt{p}} \cdot \sigma_3$
`def bit_phase_flip(blochvector, p)`	$E_0 = \sqrt{1-\mathtt{p}} \cdot I$ $E_1 = \sqrt{\mathtt{p}} \cdot \sigma_2$
`def depolarization(blochvector, p)`	$E_0 = \sqrt{1-\mathtt{p}} \cdot I$ $E_1 = \sqrt{\frac{\mathtt{p}}{3}} \cdot \sigma_1$ $E_2 = \sqrt{\frac{\mathtt{p}}{3}} \cdot \sigma_2$ $E_3 = \sqrt{\frac{\mathtt{p}}{3}} \cdot \sigma_3$

Table 1: Single qubit gate functions defined in file SingleQubitGates.py of Quantum Circuit Simulation project

Two Qubits

Overview of qubit class located in TwoQubits.py:

```
class Qubit:

def __init__(self, vec1, vec2):
"""
input:

r = [rx, ry, rz]
Blochvector entries:
rho = 0.5*(I + r*sigma_vector)

uses:
a00 ... a03
coef_matrix =   .   .     .
.     .  .
a30 ... a33

where
rho = a00*tensor(I, I)
+ a01*tensor(I, sigmaX)
+ ...
+ a33*tensor(sigmaZ, sigmaZ)

and
tensor(a,b) = tensorproduct of a and b
"""

def get_matrix(self):
# outputs current density matrix rho

def set_Pauli_basis_matrix(self, matrix):
# sets new state in Pauli basis
```

Two Qubit Gates

Two qubit gates located in file TwoQubitGates.py:

```
def operator_sum(rho, list)
# performs operator sum
# with operators in list
# on density matrix rho
```

Gate function	Matrices E_k for $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$
`def I2_gate(coef_matrix, qubitnumber)`	$E_0 = I \otimes I$
`def X2_gate(coef_matrix, qubitnumber)`	$E_0 = \begin{cases} \sigma_1 \otimes I & \texttt{qubitnumber} = 1 \\ I \otimes \sigma_1 & \texttt{qubitnumber} = 2 \end{cases}$
`def Y2_gate(coef_matrix, qubitnumber)`	$E_0 = \begin{cases} \sigma_2 \otimes I & \texttt{qubitnumber} = 1 \\ I \otimes \sigma_2 & \texttt{qubitnumber} = 2 \end{cases}$
`def Z2_gate(coef_matrix, qubitnumber)`	$E_0 = \begin{cases} \sigma_3 \otimes I & \texttt{qubitnumber} = 1 \\ I \otimes \sigma_3 & \texttt{qubitnumber} = 2 \end{cases}$
`def H2_gate(coef_matrix, qubitnumber)`	$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ $E_0 = \begin{cases} H \otimes I & \texttt{qubitnumber} = 1 \\ I \otimes H & \texttt{qubitnumber} = 2 \end{cases}$
`def S2_gate(coef_matrix, qubitnumber)`	$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix}$ $E_0 = \begin{cases} S \otimes I & \texttt{qubitnumber} = 1 \\ I \otimes S & \texttt{qubitnumber} = 2 \end{cases}$
`def T2_gate(blochvector)`	$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$ $E_0 = \begin{cases} T \otimes I & \texttt{qubitnumber} = 1 \\ I \otimes T & \texttt{qubitnumber} = 2 \end{cases}$
`def CNOTgate(coef_matrix)`	$E_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
`def CZgate(coef_matrix)`	$E_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$
`def SWAPgate(coef_matrix)`	$E_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Table 2: Two qubit gate functions defined in file TwoQubitGates.py of Quantum Circuit Simulation project

Gate function	**Matrices E_k for $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$**
`def individual_amplitude_damping (coef_matrix, Gamma, t)`	$E_{single} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & \sqrt{1 - e^{-\texttt{Gamma}\cdot\texttt{t}}} \\ 0 & 0 \end{pmatrix}$ $E_1 = E_{single} \otimes I$ $E_2 = I \otimes E_{single}$ $E_0 = \sqrt{I - \sum_i^2 E_i^\dagger E_i}$
`def fully_correlated_amplitude_damping (coef_matrix, Gamma, t)`	$E_{single} = \begin{pmatrix} 0 & \sqrt{1 - e^{-\texttt{Gamma}\cdot\texttt{t}}} \\ 0 & 0 \end{pmatrix}$ $E_1 = E_{single} \otimes E_{single}$ $E_0 = \sqrt{I - E_1^\dagger E_1}$
`def individual_phase_damping (coef_matrix, Gamma, t)`	$E_{single} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{1 - e^{-2\cdot\texttt{Gamma}\cdot\texttt{t}}} \end{pmatrix}$ $E_1 = E_{single} \otimes I$ $E_2 = I \otimes E_{single}$ $E_0 = \sqrt{I - \sum_i^2 E_i^\dagger E_i}$
`def fully_correlated_phase_damping (coef_matrix, Gamma, t)`	$E_{single} = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{1 - e^{-2\cdot\texttt{Gamma}\cdot\texttt{t}}} \end{pmatrix}$ $E_1 = E_{single} \otimes E_{single}$ $E_0 = \sqrt{I - E_1^\dagger E_1}$
`def individual_phase_flip (coef_matrix, Gamma, t)`	$E_1 = \sigma_3 \otimes I$ $E_2 = I \otimes \sigma_3$ $E_0 = \sqrt{I - \sum_i^2 E_i^\dagger E_i}$
`def fully_correlated_phase_phase (coef_matrix, Gamma, t)`	$E_1 = \sigma_3 \otimes \sigma_3$ $E_0 = \sqrt{I - E_1^\dagger E_1}$

Table 3: Two qubit noise gate functions defined in file SingleQubitGates.py of Quantum Circuit Simulation project