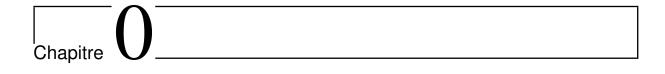
# Table des matières

| 5 | Experiment preparation |        |                                     |   |  |  |
|---|------------------------|--------|-------------------------------------|---|--|--|
|   | 5.1                    | Genera | al                                  | 1 |  |  |
|   |                        | 5.1.1  | Qubit and variable preparation      | 1 |  |  |
|   |                        | 5.1.2  | Transform spherical plan to 2d plan | 1 |  |  |
|   |                        | 5.1.3  | Data analyse for plan               | 2 |  |  |
|   |                        | 5.1.4  | Graph                               | 3 |  |  |



## Experiment preparation

On est trop souvent imprécis lorsqu'on fait une citation.

Quelqu'un, un jour.

Generative part of all experiments

#### 0.1 General

#### 0.1.1 Qubit and variable preparation

```
init_q = QuantumRegister(1, 'q')
  qc = QuantumCircuit(init_q)
  tab = [[], []]
  x = []
  y = []
7 \text{ x\_north} = []
s y_north = []
y = x_south = []
y_south = []
tab\_temp = [[], [], []]
z_0 = 0 + 0j
z1 = 0 + 0j
  z = 0 + 0j
z_north = 0*0j
 z_south = 0*0j
 shots = 100 # Never change -> Concidere it as 1 full circuit
_{19} max_shots = 100
                   # Change in fonction of how many full block we want
```

### 0.1.2 Transform spherical plan to 2d plan

```
def complex_cal(qc, statevector_sim):
       statevector_job = execute(qc, statevector_sim)
2
       statevector_result = statevector_job.result()
3
       psi = statevector_result.get_statevector()
4
       z0 = psi[0]
5
       z1 = psi[1]
6
       if z1.real != 0 or z1.imag != 0:
7
            z = z0/z1
8
            z = round(z.real, 2) + round(z.imag, 2) * 1j
9
            if np.abs(z0.real) >= 1.0 / np.sqrt(2):
10
                z_north = z0/z1
11
                z_north = round(z_north.real, 2) + round(z_north.imag, 2) * 1j
12
                z_south = 0
13
                return z, z_north, z_south
14
            if \operatorname{np.abs}(z0.real) \le 1.0 / \operatorname{np.sqrt}(2):
15
                z_south = z0/z1
16
                z_south = round(z_south.real, 2) + round(z_south.imag, 2) * 1j
17
                z 	 north = 0
18
                return z, z_north, z_south
19
       else:
20
            z = 0
21
            z 	 north = 0
22
            z \quad south = 0
23
       return z, z_north, z_south
24
```

#### 0.1.3 Data analyse for plan

```
for i in tab_temp[0]:
       iteration = tab_temp[0].count(i)
2
3
       if tab[0].count(i) < 1:
           tab [0]. append (i)
4
           tab[1].append(iteration)
5
6
  # the whole world
7
8
   for i in range (len(tab[0])):
       x.append(tab[0][i].real)
9
       y.append(tab[0][i].imag)
10
   # northern hemisphere
11
   for i in range (len (tab_temp[1])):
12
       x_north.append(tab_temp[1][i].real)
13
       y_north.append(tab_temp[1][i].imag)
14
   # southern hemisphere
15
   for i in range (len (tab_temp[2])):
16
       x_south.append(tab_temp[2][i].real)
17
18
       y_south.append(tab_temp[2][i].imag)
19
   print("Total of SV : ", len(tab_temp[0]))
20
```

#### 0.1.4 Graph

```
plt.scatter(x_cercle, y_cercle, c=tab_cercle[1], s=1, cmap="coolwarm")
plt.scatter(x, y, c=tab[1], s=1, cmap="coolwarm")
plt.colorbar()
print("Number of different value : ", len(x))
```

```
print("The whole world !")
   gen_tab = [[], [], []]
2
  {\tt zoom\_in} \, = \, 4
4
   for i in range (len(x)):
5
       if (x[i] > -1.05 \text{ and } x[i] < 2) and (y[i] > -1.05 \text{ and } y[i] < 1.05):
6
            gen_tab[0].append(x[i])
7
            gen_tab[1].append(y[i])
8
            gen_tab [2]. append (tab [1][i])
9
10
  #plt.scatter(x_cercle, y_cercle, c=tab_cercle[1], s=2, cmap="coolwarm")
11
   plt.scatter(gen_tab[0], gen_tab[1], c=gen_tab[2], s=2, cmap="coolwarm")
  #plt.colorbar()
   print("Number of different values in zoom : ", len(gen_tab[0]), "/", len(x)
      )
```