# oqcg

February 14, 2019

## 1 Imports

Need to have jate.py in your folder

```
In [47]: %run jate.py #will import everything
```

## 2 Next chapter

### 2.1 memory clear (uses regex, so be careful)

```
In [49]: %reset_selective -f var1, var2  # replace var1, var2 with your defined ones
```

### 2.2 Building parts

#### 2.2.1 Building the things to be calculated only once

```
In [51]: def maker(omega_1, H_0, H_1, T_s, Lin, d=2, gamma=0.1):
             r"""maker
             Makes all the things that remain constant throught the program, but are
             repeatedly used.


             Parameters
             ----------
             omega_1 : float
                       frequency corresponding to half of the difference between
                       energy levels of the qubit

             H_0     : Qobj
                       Bare Hamiltonian

             H_1     : Qobj
                       Interaction Hamiltonian

             T_s     : Qobj
                       Unitary to be implemented in the Hilbert space
```

```
    Lin     : Qobj
              Linbladian operators

    d       : int
              Dimension of the matrix. Defaults to 2

    gamma   : float
              Damping constantof the Linbladian


    Returns
    -------

    ih0     : Qobj
              $I\otimes H_{0}$

    ih1     : Qobj
              $I\otimes H_{1}$

    h0ci    : Qobj
              $H_{0}^{*}\otimes I $

    h1ci    : Qobj
              $H_{1}^{*}\otimes I $

    T       : Qobj
              Target unitary transformed to the Liouville space

    linbladian : Qobj
                 The full lindbladian term as it appears on transformation to
                 the Liouville space.

    """
    I = identity(d)
    L_I = tensor(I, I)
    ih0 = tensor(I, H_0)
    ih1 = tensor(I, H_1)
    h0ci = tensor(H_0.conj(), I)
    h1ci = tensor(H_1.conj(), I)
    x_k = ih1 - h1ci
    term1 = tensor(Lin.trans(), Lin)
    term2 = tensor(I, ((Lin.dag())*(Lin)))
    term3 = tensor(((Lin.trans())*(Lin.conj())), I)
    lindbladian = 1j*(gamma)*(term1 - 0.5*(term2 + term3))
    T = tensor(T_s.trans(), T_s) # Transforming $T_{s}$ to liouville space


    return ih0, ih1, h0ci, h1ci, x_k, lindbladian, T, L_I
```

```
In [53]: omega_1 = 0.5
         H_0 = omega_1*sigmaz()
         H_1 = sigmay()
         T_s = sigmax()
         Lin = sigmaz()
         ih0, ih1, h0ci, h1ci, x_k, lindbladian, T, L_I  = maker(omega_1,
                                                                 H_0, H_1, T_s,
                                                                 Lin, d=2, gamma=0.1)
```

```
In [55]: L_I
```

Out[55]:

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

### 2.2.2 Building $A(t)$

```
In [57]: def A(xi):
             r"""making $A(t)$"""
             A = ih0 - h0ci + xi*(ih1 - h1ci) + lindbladian
             return A
```

```
In [59]: A(0.5)
```

Out[59]:

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.0 & -0.500j & -0.500j & 0.0 \\ 0.500j & (-1.0-0.200j) & 0.0 & -0.500j \\ 0.500j & 0.0 & (1.0-0.200j) & -0.500j \\ 0.0 & 0.500j & 0.500j & 0.0 \end{pmatrix}$$

### 2.2.3 Building $L(t)$ and the Identity in the Liouville space

```
In [61]: def L(xi, dt):
             r"""Making $L(t) from $A(t)$"""
             L = (-1j*A(xi)*dt).expm()
             return L
```

```
In [63]: L(0.5, 0.001)
```

Out[63]:

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\left(\begin{array}{cccc} 1.000 & (-4.999\times10^{-04}-2.500\times10^{-07}j) & (-4.999\times10^{-04}+2.500\times10^{-07}j) & \\ (4.999\times10^{-04}+2.500\times10^{-07}j) & (1.000+9.998\times10^{-04}j) & -2.500\times10^{-07} & (- \\ (4.999\times10^{-04}-2.500\times10^{-07}j) & -2.500\times10^{-07} & (1.000-9.998\times10^{-04}j) & (- \\ 2.500\times10^{-07} & (4.999\times10^{-04}+2.500\times10^{-07}j) & (4.999\times10^{-04}-2.500\times10^{-07}j) & \end{array}\right.$$

## 2.3 Major functions

### 2.3.1 Major functions 1

```
In [64]: # building the function to optimize (optimizee)
         def L_vec(xi_vec, dt):
             r"""Building the vector of differential $L(t)$"""
             L_vec = [L(xi, dt) for xi in xi_vec]
             return L_vec
```

```
In [65]: def fidelity_calc(A, B):
             r"""Making a generalised fidelity function"""
             first_part = (A - B).dag()
             second_part = (A - B)
             f_int = (first_part* second_part)
             f = f_int.tr()
             return f
```

```
In [66]: def L_full_maker(xi_vec, dt):
             r"""Building the $L(t)$ for the total time $t$"""
             xi_vec_size = xi_vec.size # finding the size of xi
             L_full = L_I # Identity for the for loop of L
             L_v = L_vec(xi_vec, dt) # calling L_vec
             for i in range(xi_vec_size): # generating L_full
                 L_full = L_full*L_v[xi_vec_size - 1 - i]
             return L_full
```

```
In [67]: def F(xi_vec, dt):
             r"""Using the fidelity metric to find out the closeness between $T$
             and $L(t)$"""
             L_full = L_full_maker(xi_vec, dt)
             F = real(-fidelity_calc(T, L_full))
             return F
```

### 2.3.2 Testing major functions 1

```
In [68]: fidelity_calc(sigmax(), sigmay())

Out[68]: 4.0

In [69]: fidelity_calc(sigmay(), sigmay())

Out[69]: 0.0

In [70]: xi_vec_test = array([1.0, 2.0])
         xi_vec_test

Out[70]: array([1., 2.])

In [71]: xi_vec_test.size
```

```
Out[71]: 2

In [72]: w_vec = [xi**2 for xi in xi_vec_test]
         w_vec

Out[72]: [1.0, 4.0]

In [73]: # F(xi_vec, dt)
         F(xi_vec_test, 0.001)

Out[73]: -7.998400634493138

In [74]: L_v = L_vec(xi_vec_test, 0.001)

In [75]: L_v

Out[75]: [Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False
          Qobj data =
          [[ 9.99999000e-01+0.00000000e+00j -9.99899173e-04-4.99933130e-07j
            -9.99899173e-04+4.99933130e-07j  9.99932920e-07+0.00000000e+00j]
           [ 9.99899173e-04+4.99933130e-07j  9.99798520e-01+9.99799187e-04j
            -9.99866260e-07+0.00000000e+00j -9.99899173e-04-4.99933130e-07j]
           [ 9.99899173e-04-4.99933130e-07j -9.99866260e-07+0.00000000e+00j
             9.99798520e-01-9.99799187e-04j -9.99899173e-04+4.99933130e-07j]
           [ 9.99932920e-07+0.00000000e+00j  9.99899173e-04+4.99933130e-07j
             9.99899173e-04-4.99933130e-07j  9.99999000e-01+0.00000000e+00j]],
          Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False
          Qobj data =
          [[ 9.99996000e-01+0.00000000e+00j -1.99979435e-03-9.99865260e-07j
            -1.99979435e-03+9.99865260e-07j  3.99972768e-06+0.00000000e+00j]
           [ 1.99979435e-03+9.99865260e-07j  9.99795521e-01+9.99797187e-04j
            -3.99946104e-06+0.00000000e+00j -1.99979435e-03-9.99865260e-07j]
           [ 1.99979435e-03-9.99865260e-07j -3.99946104e-06+0.00000000e+00j
             9.99795521e-01-9.99797187e-04j -1.99979435e-03+9.99865260e-07j]
           [ 3.99972768e-06+0.00000000e+00j  1.99979435e-03+9.99865260e-07j
             1.99979435e-03-9.99865260e-07j  9.99996000e-01+0.00000000e+00j]]]
```

### 2.3.3 Major Functions 2

```
In [76]: def L_comma_k_maker(xi_vec, k, dt):
             r"""Making of the derivative of full $L(t)$ at time $t_{k}$"""
             N = xi_vec.size
             # Determining the size of xi, and thus the time_steps indirectly.
             L_v = L_vec(xi_vec, dt)# Making of the full $L(t)$
             inner_part = L_I # Beginner for the for loop
             for i in range(N):
                 if i == ( N - 1 - k ):
                     # The step at which $X_{k}(t)$ has to be inserted
                     inner_part = inner_part*x_k*L_v[k - 1]
```

```
            else:
                # Usual multiplications of $L_{k}$
                inner_part = inner_part*L_v[N - 1 - i]
        l_comma_k = inner_part
        return l_comma_k
```

```
In [77]: # L_comma_k_maker(xi_vec, k, dt)
         L_comma_k_maker(xi_vec_test, 2, 0.001)
```

Out[77]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\left(\begin{array}{cccc} 1.000 & (-0.003-3.499\times10^{-06}j) & (-0.003+3.499\times10^{-06}j) & 8.999 \\ (0.003+2.499\times10^{-06}j) & (1.000+0.002j) & (-8.998\times10^{-06}+2.999\times10^{-09}j) & (-0.003- 2 \\ (0.003-2.499\times10^{-06}j) & (-8.998\times10^{-06}-2.999\times10^{-09}j) & (1.000-0.002j) & (-0.003+ 2 \\ 8.999\times10^{-06} & (0.003+3.499\times10^{-06}j) & (0.003-3.499\times10^{-06}j) & 1 \end{array}\right.$$

```
In [78]: def updater(xi_vec, dt, epsilon):
             r"""Implementing the GRAPE update step"""
             xi_vec_size = xi_vec.size # finding the size of xi
             L_full = L_full_maker(xi_vec, dt)
             di = []
             for k in range(xi_vec_size):
                 # Building the thing to be added to the old function
                 L_comma_k = L_comma_k_maker(xi_vec, k, dt)
                 differentiated = T - L_comma_k
                 plain = T - L_full
                 c = -differentiated.dag()*plain
                 d = -plain.dag()*differentiated
                 inside = c.tr() + d.tr()
                 di.append(epsilon*inside)

             diff = array(di)
             xi_new_vec = xi_vec + diff
             return diff, xi_new_vec
```

```
In [79]: #  updater(xi_vec, dt, epsilon)
         updater(xi_vec_test, 0.001, 0.001)

Out[79]: (array([-0.008+0.j, -0.008+0.j]), array([0.992+0.j, 1.992+0.j]))

In [80]: import time

In [81]: total_time_evo = 2*pi # total time allowed for evolution

In [82]: times = linspace(0, total_time_evo, 500)

In [83]: # vector of times at which discretization
         # is carried out
```

```
In [84]: U = T_s
         U
```

Out[84]:

Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 1.0 \\ 1.0 & 0.0 \end{pmatrix}$$

```
In [85]: R = 500

In [86]: H_ops = [H_1]
         H_ops

Out[86]: [Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True
          Qobj data =
          [[0.+0.j 0.-1.j]
           [0.+1.j 0.+0.j]]]

In [87]: H_labels = [r'$g_{no diss}$']
         H_labels

Out[87]: ['$g_{no diss}$']

In [88]: H0 = H_0
         H0
```

Out[88]:

Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

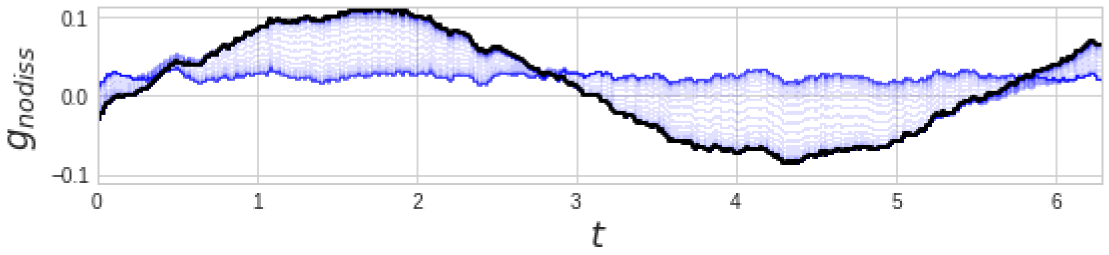
$$\begin{pmatrix} 0.500 & 0.0 \\ 0.0 & -0.500 \end{pmatrix}$$

```
In [89]: c_ops = []

In [90]: from qutip.control.grape import plot_grape_control_fields, _overlap
         from qutip.control.grape import grape_unitary_adaptive, cy_grape_unitary

In [91]: from scipy.interpolate import interp1d
         from qutip.ui.progressbar import TextProgressBar

In [92]: u0 = array([rand(len(times)) * 2 * pi * 0.05 for _ in range(len(H_ops))])

In [93]: from numpy import convolve
         u0 = [convolve(ones(10)/10, u0[idx,:], mode='same') for idx in range(len(H_ops))]

In [94]: u_limits = None #[0, 1 * 2 * pi]
         alpha = None

In [95]: result = cy_grape_unitary(U, H0, H_ops, R, times, u_start=u0, u_limits=u_limits,
                                   eps=2*np.pi*1, alpha=alpha, phase_sensitive=False,
                                   progress_bar=TextProgressBar())
```

```
10.0%. Run time:  33.00s. Est. time left: 00:00:04:57
20.0%. Run time:  65.56s. Est. time left: 00:00:04:22
30.0%. Run time:  98.20s. Est. time left: 00:00:03:49
40.0%. Run time: 130.76s. Est. time left: 00:00:03:16
50.0%. Run time: 163.29s. Est. time left: 00:00:02:43
60.0%. Run time: 196.05s. Est. time left: 00:00:02:10
70.0%. Run time: 228.23s. Est. time left: 00:00:01:37
80.0%. Run time: 260.41s. Est. time left: 00:00:01:05
90.0%. Run time: 292.53s. Est. time left: 00:00:00:32
Total run time: 324.17s
```

**Plot of optimized control field without dissipation**

```
In [96]: plot_grape_control_fields(times,
                                   result.u / (2 * np.pi), H_labels, uniform_axes=True);
```



```
In [97]: U
```

Out[97]:
Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 1.0 \\ 1.0 & 0.0 \end{pmatrix}$$

```
In [98]: result.U_f
```

Out[98]:
Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = False

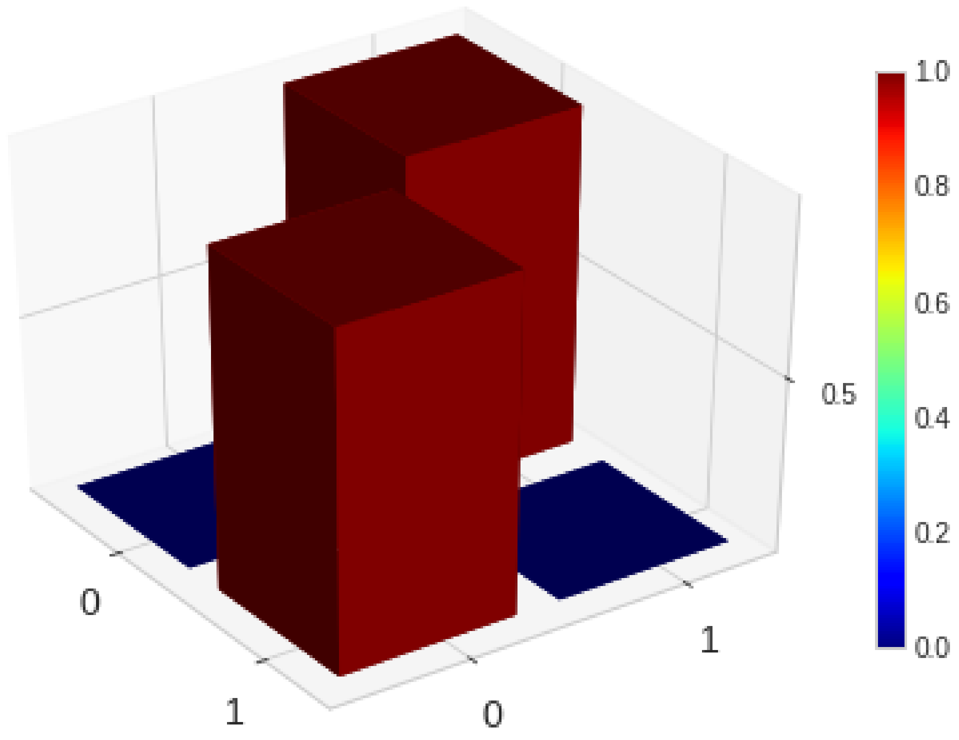
$$\begin{pmatrix} 4.195 \times 10^{-16} & 1.000j \\ 1.0j & -5.690 \times 10^{-16} \end{pmatrix}$$

```
In [99]: result.U_f/result.U_f[0,0]
```

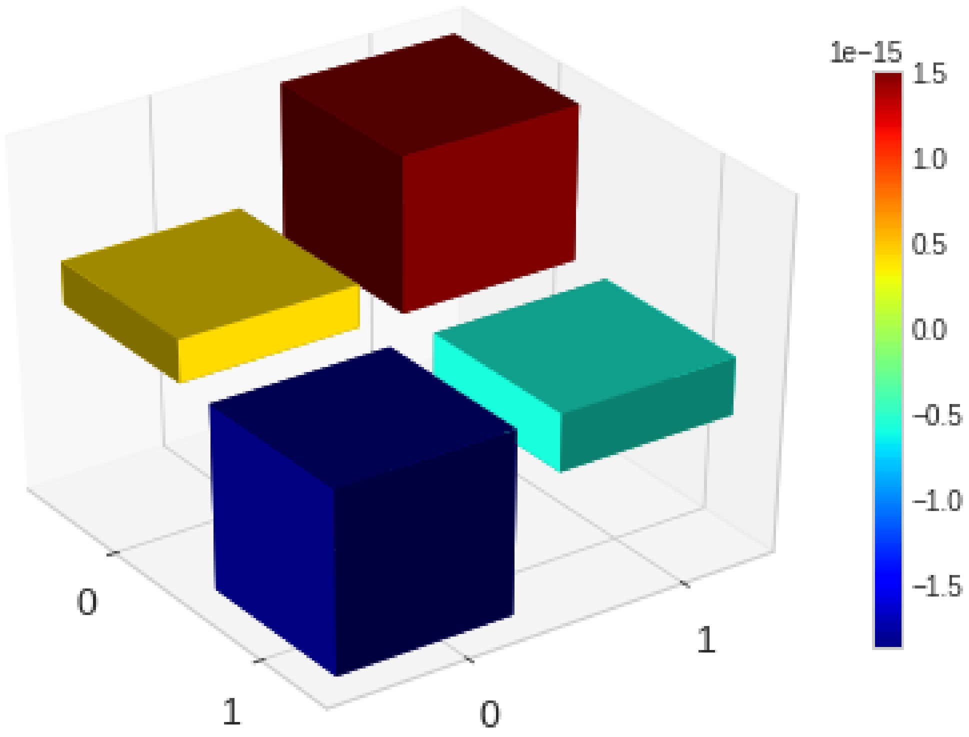
Out[99]:
Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = False

$$\begin{pmatrix} 1.0 & (1.179 \times 10^{+15} + 1.365 \times 10^{+15}j) \\ (1.179 \times 10^{+15} + 1.365 \times 10^{+15}j) & (-1.233 + 0.143j) \end{pmatrix}$$

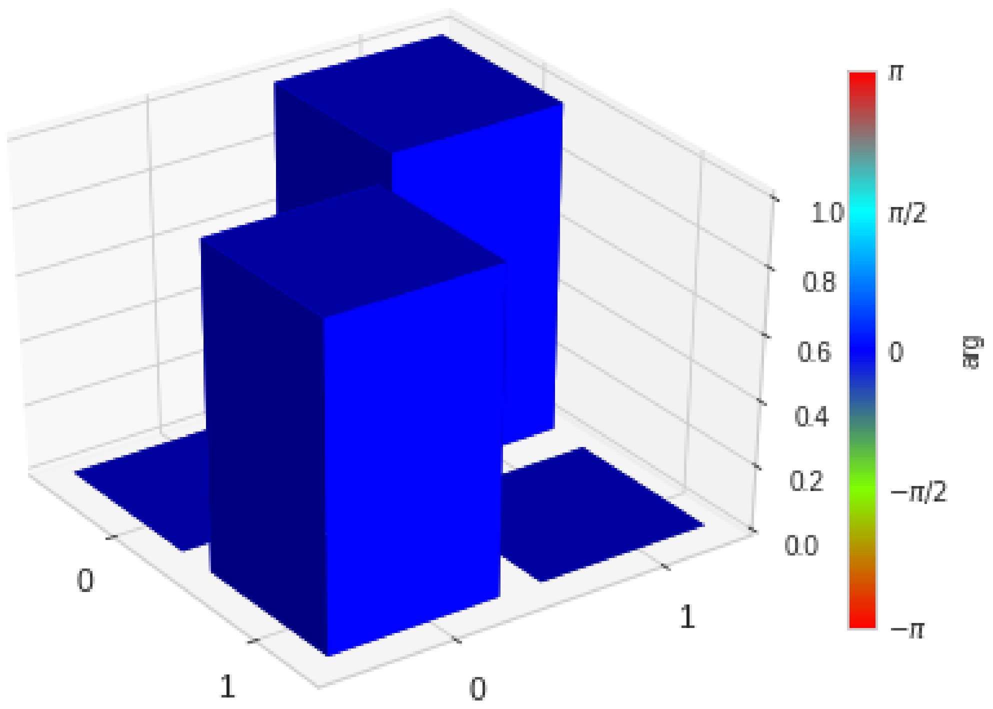
```
In [100]: matrix_histogram(U)

Out[100]: (<Figure size 432x288 with 2 Axes>,
           <mpl_toolkits.mplot3d.axes3d.Axes3D at 0x7fa12c63a518>)
```



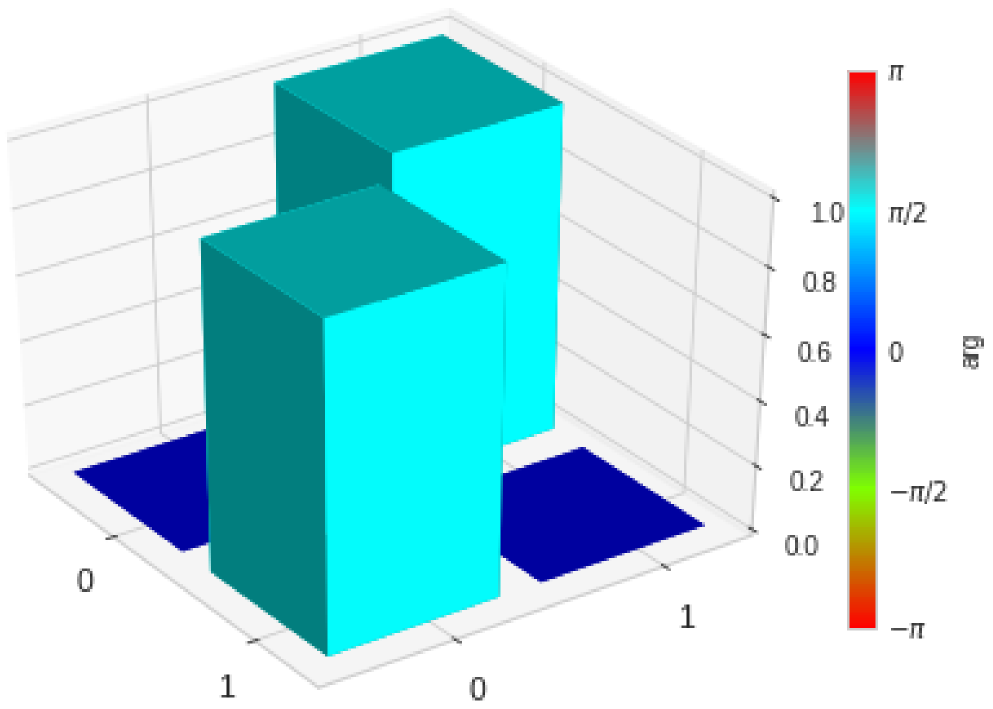
```
In [101]: matrix_histogram(result.U_f)

Out[101]: (<Figure size 432x288 with 2 Axes>,
           <mpl_toolkits.mplot3d.axes3d.Axes3D at 0x7fa12c5ccd68>)
```



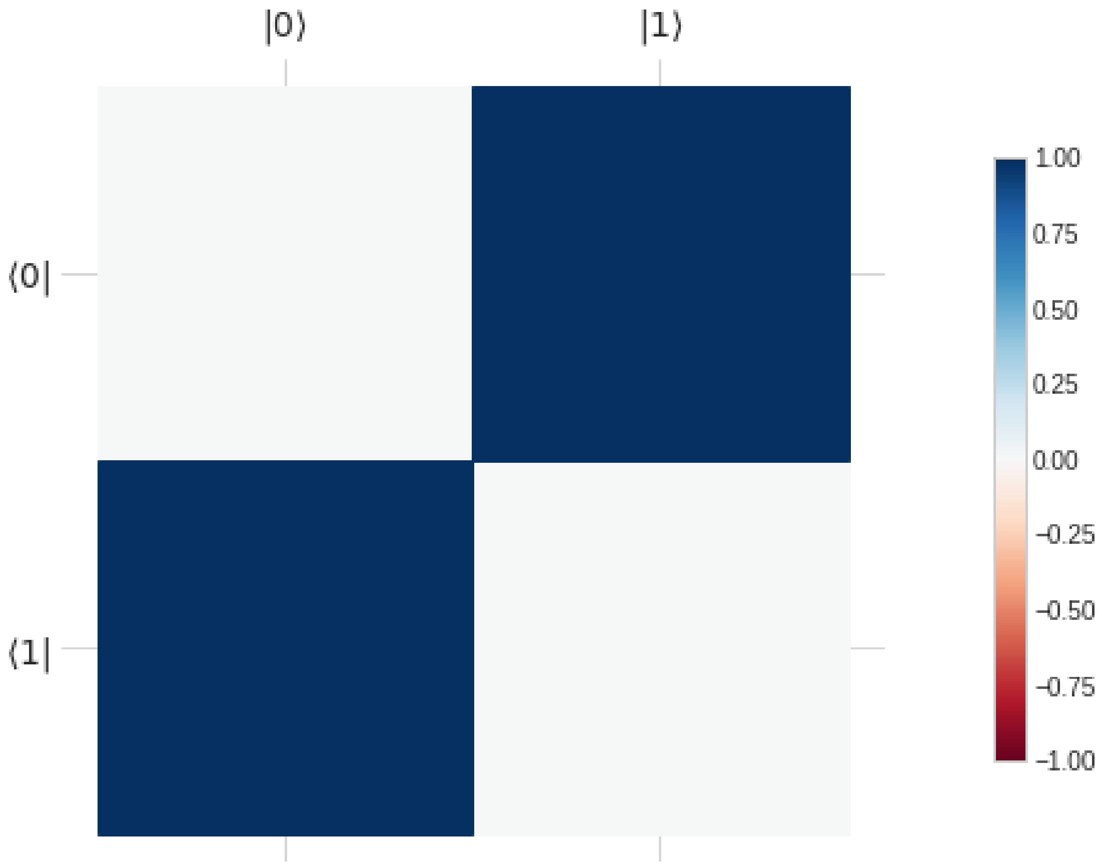
```
In [102]: matrix_histogram_complex(U)

Out[102]: (<Figure size 432x288 with 2 Axes>,
           <mpl_toolkits.mplot3d.axes3d.Axes3D at 0x7fa12c56dcf8>)
```



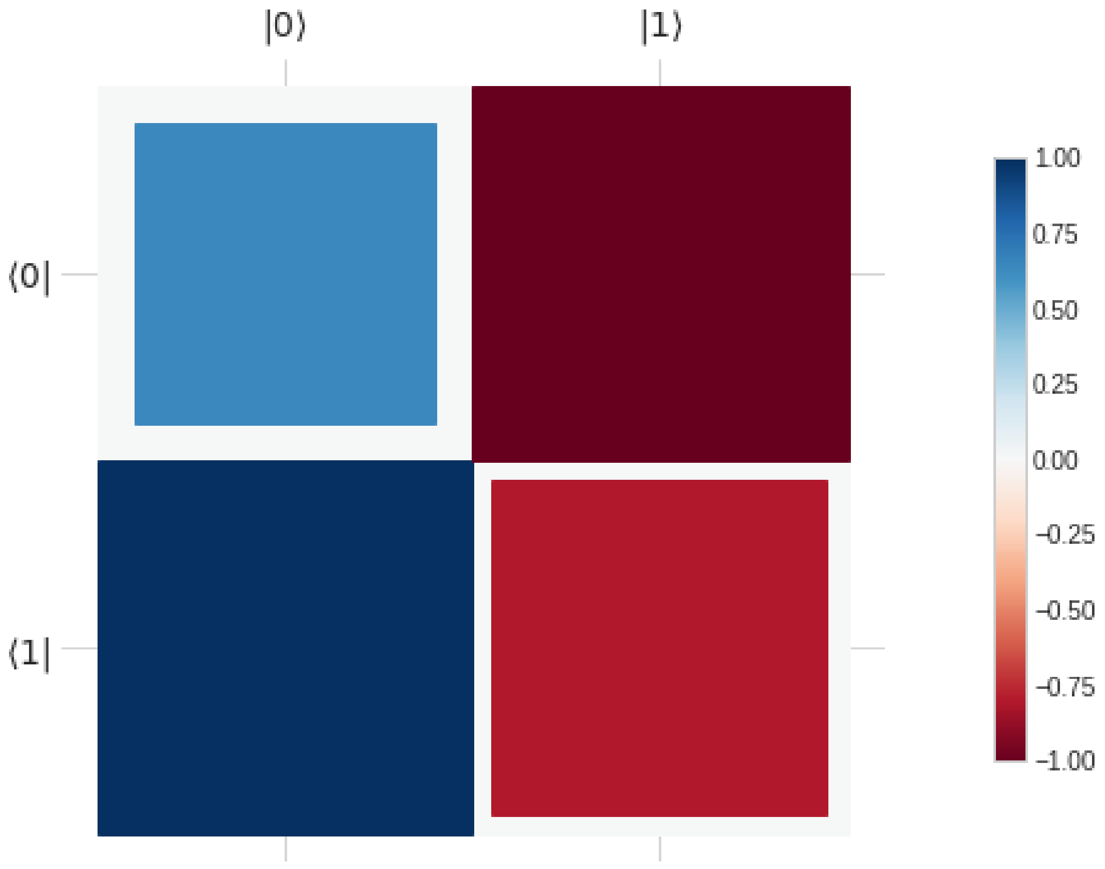
```
In [103]: matrix_histogram_complex(result.U_f)

Out[103]: (<Figure size 432x288 with 2 Axes>,
           <mpl_toolkits.mplot3d.axes3d.Axes3D at 0x7fa12c4a9d30>)
```



```
In [104]: hinton(U)

Out[104]: (<Figure size 576x432 with 2 Axes>,
           <matplotlib.axes._subplots.AxesSubplot at 0x7fa12c678d68>)
```



```
In [105]: hinton(result.U_f)

Out[105]: (<Figure size 576x432 with 2 Axes>,
            <matplotlib.axes._subplots.AxesSubplot at 0x7fa12c69c5f8>)
```



```
In [106]: times[-1]

Out[106]: 6.283185307179586

In [107]: total_time_evo

Out[107]: 6.283185307179586

In [108]: total_time
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-108-9bec417112c8> in <module>()
    ----> 1 total_time


        NameError: name 'total_time' is not defined
```

```
In [109]: len(times)

Out[109]: 500

In [110]: def terminator(max_iter, time_steps=len(times), total_time= total_time_evo,
                         epsilon= 2*pi*1):
              r"""Brief description of the function"""

              xi_initial = result.u[-1, 0, : ]
              #1000*random_sample((time_steps,))
              dt = (2*pi)/500  #total_time/time_steps
              xi_diff, xi_new_vec = updater(xi_initial, dt, epsilon)

              for i in range(max_iter):
                  if amax(xi_diff) < epsilon**2 :

                      xi_final = xi_new_vec
                      break
                  else :
                      xi_diff, xi_new_vec = updater(xi_new_vec, dt, epsilon)
                      print(i)
                      print(amax(xi_diff))


              xi_final = xi_new_vec
              return xi_final
```

## 2.4 sub topic 3

### 2.4.1 try

```
In [111]: xi_opt = terminator(10)

In [112]: time_steps=len(times)
          total_time= total_time_evo
          epsilon= 2*pi*1

In [113]: dt = (2*pi)/500
          F(xi_opt, dt)

Out[113]: -4.223783391928767

In [114]: L_full_maker(xi_opt, dt)
```

```
Out[114]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.345 & (0.217+0.007j) & (0.217-0.007j) & 0.655 \\ (-0.217-0.007j) & (-0.014+0.010j) & (-0.298+8.451\times10^{-05}j) & (0.217+0.007j) \\ (-0.217+0.007j) & (-0.298-8.451\times10^{-05}j) & (-0.014-0.010j) & (0.217-0.007j) \\ 0.655 & (-0.217-0.007j) & (-0.217+0.007j) & 0.345 \end{pmatrix}$$

### 2.4.2 try

```
In [115]: xi_opt = terminator(1000)
```

```
In [116]: time_steps=len(times)
          total_time= total_time_evo
          epsilon= 2*pi*1
```

```
In [117]: dt = (2*pi)/500
          F(xi_opt, dt)
```

```
Out[117]: -4.223783391928767
```

```
In [118]: L_full_maker(xi_opt, dt)
```

```
Out[118]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.345 & (0.217+0.007j) & (0.217-0.007j) & 0.655 \\ (-0.217-0.007j) & (-0.014+0.010j) & (-0.298+8.451\times10^{-05}j) & (0.217+0.007j) \\ (-0.217+0.007j) & (-0.298-8.451\times10^{-05}j) & (-0.014-0.010j) & (0.217-0.007j) \\ 0.655 & (-0.217-0.007j) & (-0.217+0.007j) & 0.345 \end{pmatrix}$$

### 2.4.3 try

```
In [121]: xi_opt = terminator(1000,time_steps=len(times), total_time= total_time_evo,
                              epsilon= ((0.1*2*pi)/(times[-1])))
```

```
In [123]: time_steps=len(times)
          total_time= total_time_evo
          epsilon = ((0.1*2*pi)/(times[-1]))
```

```
In [124]: dt = (2*pi)/500
          F(xi_opt, dt)
```

```
Out[124]: -2.239742398764373
```

```
In [125]: L_full_maker(xi_opt, dt)
```

```
Out[125]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.323 & (0.091-0.088j) & (0.091+0.088j) & 0.677 \\ (0.244+0.061j) & (-0.053+0.008j) & (0.175+0.262j) & (-0.244-0.061j) \\ (0.244-0.061j) & (0.175-0.262j) & (-0.053-0.008j) & (-0.244+0.061j) \\ 0.677 & (-0.091+0.088j) & (-0.091-0.088j) & 0.323 \end{pmatrix}$$

### 2.4.4 try

```
In [126]: xi_opt = terminator(10,time_steps=len(times), total_time= total_time_evo,
                             epsilon= ((0.1*2*pi)/(times[-1])))

In [127]: time_steps=len(times)
          total_time= total_time_evo
          epsilon = ((0.1*2*pi)/(times[-1]))

In [128]: dt = (2*pi)/500
          F(xi_opt, dt)

Out[128]: -2.239742398764373

In [129]: L_full_maker(xi_opt, dt)
```

Out[129]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.323 & (0.091-0.088j) & (0.091+0.088j) & 0.677 \\ (0.244+0.061j) & (-0.053+0.008j) & (0.175+0.262j) & (-0.244-0.061j) \\ (0.244-0.061j) & (0.175-0.262j) & (-0.053-0.008j) & (-0.244+0.061j) \\ 0.677 & (-0.091+0.088j) & (-0.091-0.088j) & 0.323 \end{pmatrix}$$

### 2.4.5 try

```
In [130]: xi_opt = terminator(1000,time_steps=len(times), total_time= total_time_evo,
                             epsilon= ((0.1*2*pi)/(10**3)))

In [131]: time_steps=len(times)
          total_time= total_time_evo
          epsilon = ((0.1*2*pi)/(times[-1]))

In [132]: dt = (2*pi)/500
          F(xi_opt, dt)

Out[132]: -0.9548953874356939

In [133]: L_full_maker(xi_opt, dt)
```

Out[133]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.220 & (0.033-0.026j) & (0.033+0.026j) & 0.780 \\ (-0.038+0.005j) & (-0.065-0.048j) & (0.394+0.015j) & (0.038-0.005j) \\ (-0.038-0.005j) & (0.394-0.015j) & (-0.065+0.048j) & (0.038+0.005j) \\ 0.780 & (-0.033+0.026j) & (-0.033-0.026j) & 0.220 \end{pmatrix}$$

### 2.4.6 try

```
In [134]: xi_opt = terminator(1000,time_steps=len(times), total_time= total_time_evo,
                              epsilon= ((0.1*2*pi)/(10**4)))
```

```
In [135]: dt = (2*pi)/500
          F(xi_opt, dt)
```

```
Out[135]: -0.9547563688866646
```

```
In [136]: L_full_maker(xi_opt, dt)
```

```
Out[136]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.220 & (0.035-0.027j) & (0.035+0.027j) & 0.780 \\ (-0.038+0.005j) & (-0.064-0.048j) & (0.394+0.012j) & (0.038-0.005j) \\ (-0.038-0.005j) & (0.394-0.012j) & (-0.064+0.048j) & (0.038+0.005j) \\ 0.780 & (-0.035+0.027j) & (-0.035-0.027j) & 0.220 \end{pmatrix}$$

### 2.4.7 try

```
In [137]: xi_opt = terminator(10**4,time_steps=len(times), total_time= total_time_evo,
                              epsilon= ((0.1*2*pi)/(10**3)))
```

```
In [138]: dt = (2*pi)/500
          F(xi_opt, dt)
```

```
Out[138]: -0.9548953874356939
```

```
In [139]: L_full_maker(xi_opt, dt)
```

```
Out[139]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.220 & (0.033-0.026j) & (0.033+0.026j) & 0.780 \\ (-0.038+0.005j) & (-0.065-0.048j) & (0.394+0.015j) & (0.038-0.005j) \\ (-0.038-0.005j) & (0.394-0.015j) & (-0.065+0.048j) & (0.038+0.005j) \\ 0.780 & (-0.033+0.026j) & (-0.033-0.026j) & 0.220 \end{pmatrix}$$

### 2.4.8 try

```
In [140]: xi_opt = terminator(10**4,time_steps=len(times), total_time= total_time_evo,
                              epsilon= ((0.1*2*pi)/(10**4)))
```

```
In [141]: dt = (2*pi)/500
          F(xi_opt, dt)
```

```
Out[141]: -0.9547563688866646
```

```
In [142]: L_full_maker(xi_opt, dt)
```

```
Out[142]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.220 & (0.035-0.027j) & (0.035+0.027j) & 0.780 \\ (-0.038+0.005j) & (-0.064-0.048j) & (0.394+0.012j) & (0.038-0.005j) \\ (-0.038-0.005j) & (0.394-0.012j) & (-0.064+0.048j) & (0.038+0.005j) \\ 0.780 & (-0.035+0.027j) & (-0.035-0.027j) & 0.220 \end{pmatrix}$$

### 2.4.9 try

```
In [143]: xi_opt = terminator(10**4,time_steps=len(times), total_time= total_time_evo,
                              epsilon= ((0.1*2*pi)/(10**10)))

In [144]: dt = (2*pi)/500
          F(xi_opt, dt)

Out[144]: -0.9547464377063413

In [145]: L_full_maker(xi_opt, dt)

Out[145]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.220 & (0.035-0.027j) & (0.035+0.027j) & 0.780 \\ (-0.038+0.005j) & (-0.064-0.048j) & (0.394+0.012j) & (0.038-0.005j) \\ (-0.038-0.005j) & (0.394-0.012j) & (-0.064+0.048j) & (0.038+0.005j) \\ 0.780 & (-0.035+0.027j) & (-0.035-0.027j) & 0.220 \end{pmatrix}$$

### 2.4.10 try

```
In [146]: xi_opt = terminator(10**4,time_steps=10**3, total_time= total_time_evo,
                              epsilon= ((0.1*2*pi)/(10**3)))

In [147]: dt = (2*pi)/(10**3)#(2*pi)/500
          F(xi_opt, dt)

Out[147]: -4.865356728657208

In [148]: L_full_maker(xi_opt, dt)

Out[148]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.756 & (-0.333-0.089j) & (-0.333+0.089j) & 0.244 \\ (-0.257+0.072j) & (-0.401+0.138j) & (0.147-0.054j) & (0.257-0.072j) \\ (-0.257-0.072j) & (0.147+0.054j) & (-0.401-0.138j) & (0.257+0.072j) \\ 0.244 & (0.333+0.089j) & (0.333-0.089j) & 0.756 \end{pmatrix}$$

### 2.5 Versions

```
In [119]: from qutip.ipynbtools import version_table

          version_table()

Out[119]: <IPython.core.display.HTML object>

In [120]: cnot()

Out[120]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{pmatrix}$$