scratchpad1

February 2, 2019

1 Imports

In [71]:
```
# # Imports

# ## Qutip imports 1

# In[1]:


from qutip.operators import sigmax, sigmay, sigmaz, identity, qeye
from qutip.operators import position, momentum, num, create, destroy
from qutip.operators import commutator, qdiags
from qutip.simdiag import simdiag
from qutip.states import basis, ket2dm
from qutip.tensor import tensor
from qutip.qip.gates import swap, rx, ry, rz, cnot
from qutip.qobj import Qobj
from qutip.visualization import hinton
from qutip.visualization import matrix_histogram_complex, matrix_histogram
from qutip.random_objects import rand_herm, rand_unitary, rand_dm


# ## Qutip imports 2

# In[3]:


#from


# ## Numpy imports

# In[2]:


from numpy import sin, cos, tan, real, imag,  log, conj
from numpy import array, append, linspace, arange
```

```
from numpy import add, sqrt, abs, dot
from numpy.random import random, random_sample, rand, seed, RandomState
from numpy import concatenate, hstack, vstack, block, dstack, vsplit
from numpy import trace, diag
from numpy import ones, zeros, ones_like, zeros_like
from numpy import amax, amin, nanmax, nanmin
from numpy import outer, multiply
# from numpy import pi


# ## Scipy imports

# In[17]:


from scipy.integrate import ode, odeint, complex_ode
from scipy.optimize import minimize
from scipy.linalg import eigh, inv, norm, expm
# from scipy.linalg import
# from scipy import


# ## Matplotlib imports

# In[18]:


from matplotlib.pyplot import plot, figure, show, savefig, axes
from matplotlib.pyplot import xlabel, ylabel, title, legend
from matplotlib import rcParams
from matplotlib.pyplot import style
from matplotlib.pyplot import xlim, ylim, axis
# beware not same as axes
from matplotlib.pyplot import subplot, subplots, text
from matplotlib.pyplot import GridSpec
from matplotlib.pyplot import scatter, colorbar


pgf_with_rc_fonts = {"pgf.texsystem": "pdflatex"}
rcParams.update(pgf_with_rc_fonts)
style.use('seaborn-whitegrid')


# ## Math imports

# In[19]:
```

```
from math import pi
from math import exp


# ## Cmath imports

# ## Date and datetime imports

# In[20]:


from datetime import date
from datetime import datetime# now


# ## Os imports

# In[21]:


from os import getcwd, mkdir, chdir
from os.path import abspath, join


# ## Sympy imports

# In[22]:


from sympy import Function, dsolve, Eq, Derivative, symbols
# x, y, z, t = symbols('x y z t')
# k, m, n = symbols('k m n', integer=True)
# f, g, h = symbols('f g h', cls=Function)


# ## Miscellaneous imports

# ## Extra useful functions

# In[23]:


def rint(x):
    print("x = ", x)
    return None


# # Next chapter
```

```
# ## sub topic 1

# ## sub topic 2

# ## sub topic 3

# ### sub sub topic 1
```

2 Next chapter

2.1 memory clear (uses regex, so be careful)

```
In [30]: %reset_selective -f var1, var2  # replace var1, var2 with your defined ones
```

2.2 sub topic 2

```
In [31]: cnot()
```

Out[31]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

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

```
In [32]: (qeye(2)- sigmaz())/2
```

Out[32]:
Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

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

```
In [33]: (-qeye(2) + sigmax())
```

Out[33]:
Quantum object: dims = [[2], [2]], shape = (2, 2), type = oper, isherm = True

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

```
In [34]: tensor((qeye(2)- sigmaz())/2,(-qeye(2) + sigmax())          )
```

Out[34]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

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

```
In [35]: (1j*(pi/4)*tensor((qeye(2)- sigmaz())/2,(-qeye(2) + sigmax())          )).expm()
```

Out[35]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

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

```
In [10]: (1*(pi/4)*tensor((qeye(2)- sigmaz())/2,(-qeye(2) + sigmax())          )).expm()
```

Out[10]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.604 & 0.396 \\ 0.0 & 0.0 & 0.396 & 0.604 \end{pmatrix}$$

```
In [12]: (1j*(pi/4)*(1j)*tensor((qeye(2)- sigmaz())/2,(-qeye(2) + sigmax())  )).expm()
```

Out[12]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 2.905 & -1.905 \\ 0.0 & 0.0 & -1.905 & 2.905 \end{pmatrix}$$

2.3 sub topic 3

2.3.1 sub sub topic 1

```
In [13]: cnot()
```

Out[13]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

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

```
In [14]: cnot().eigenenergies()

Out[14]: array([-1.,  1.,  1.,  1.])

In [15]: cnot().eigenstates()
```

```
Out[15]: (array([-1.,  1.,  1.,  1.]),
          array([Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
          Qobj data =
          [[ 0.        ]
           [ 0.        ]
           [ 0.70710678]
           [-0.70710678]],
                 Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
          Qobj data =
          [[0.]
           [1.]
           [0.]
           [0.]],
                 Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
          Qobj data =
          [[1.]
           [0.]
           [0.]
           [0.]],
                 Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
          Qobj data =
          [[0.        ]
           [0.        ]
           [0.70710678]
           [0.70710678]]], dtype=object))

In [16]: cnot().diag()

Out[16]: array([1., 1., 0., 0.])

In [20]: c_eigenvalue, c_eigenvec_arr = cnot().eigenstates()

In [23]: c_eigenvec

Out[23]: array([Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
         Qobj data =
         [[ 0.        ]
          [ 0.        ]
          [ 0.70710678]
          [-0.70710678]],
                Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
         Qobj data =
         [[0.]
          [1.]
          [0.]
          [0.]],
                Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
         Qobj data =
         [[1.]
```

```
          [0.]
          [0.]
          [0.]],
                Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
         Qobj data =
         [[0.        ]
          [0.        ]
          [0.70710678]
          [0.70710678]]], dtype=object)
```

```
In [24]: c_eigenvalue
```

```
Out[24]: array([-1.,  1.,  1.,  1.])
```

```
In [27]: qdiags(2)
```

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

        TypeError                                 Traceback (most recent call last)

        <ipython-input-27-61a25a066823> in <module>()
    ----> 1 qdiags(2)


        TypeError: qdiags() missing 1 required positional argument: 'offsets'
```

```
In [37]: simdiag(array([sigmax(), qeye(2)]))
```

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

        AttributeError                            Traceback (most recent call last)

        <ipython-input-37-54d465f50d4b> in <module>()
    ----> 1 simdiag(array([sigmax(), qeye(2)]))


        /anaconda3/envs/qutip-env/lib/python3.6/site-packages/qutip/simdiag.py in simdiag(ops,
         82     eigvals, eigvecs = la.eig(A.full())
         83     zipped = zip(-eigvals, range(len(eigvals)))
    ---> 84     zipped.sort()
         85     ds, perm = zip(*zipped)
         86     ds = -np.real(np.array(ds))


        AttributeError: 'zip' object has no attribute 'sort'
```

```
In [26]: d = qdiags(c_eigenvalue)
         d
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

<ipython-input-26-c2334747d78d> in <module>()
----> 1 d = qdiags(c_eigenvalue)
      2 d


TypeError: qdiags() missing 1 required positional argument: 'offsets'
```

```
In [38]: Qobj(diag(c_eigenvalue.full()))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)

<ipython-input-38-66948a1e6342> in <module>()
----> 1 Qobj(diag(c_eigenvalue.full()))


AttributeError: 'numpy.ndarray' object has no attribute 'full'
```

```
In [40]: ceq = Qobj(diag(c_eigenvalue))
         ceq
```

Out[40]:

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

$$\begin{pmatrix} -1.000 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

```
In [42]: Qobj(arange(9).reshape((3,3)))
```

Out[42]:

Quantum object: dims = [[3], [3]], shape = (3, 3), type = oper, isherm = False

$$\begin{pmatrix} 0.0 & 1.0 & 2.0 \\ 3.0 & 4.0 & 5.0 \\ 6.0 & 7.0 & 8.0 \end{pmatrix}$$

```
In [44]: inter = [a.full() for a in c_eigenvec]

In [45]: inter

Out[45]: [array([[ 0.         +0.j],
                 [ 0.         +0.j],
                 [ 0.70710678+0.j],
                 [-0.70710678+0.j]]), array([[0.+0.j],
                 [1.+0.j],
                 [0.+0.j],
                 [0.+0.j]]), array([[1.+0.j],
                 [0.+0.j],
                 [0.+0.j],
                 [0.+0.j]]), array([[0.         +0.j],
                 [0.         +0.j],
                 [0.70710678+0.j],
                 [0.70710678+0.j]])]

In [46]: array(c_eigenvec)

Out[46]: array([Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
         Qobj data =
         [[ 0.        ]
          [ 0.        ]
          [ 0.70710678]
          [-0.70710678]],
                Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
         Qobj data =
         [[0.]
          [1.]
          [0.]
          [0.]],
                Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
         Qobj data =
         [[1.]
          [0.]
          [0.]
          [0.]],
                Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
         Qobj data =
         [[0.        ]
          [0.        ]
          [0.70710678]
          [0.70710678]]], dtype=object)

In [47]: array(inter)

Out[47]: array([[[ 0.         +0.j],
                  [ 0.         +0.j],
```

```
                [ 0.70710678+0.j],
                [-0.70710678+0.j]],

               [[ 0.        +0.j],
                [ 1.        +0.j],
                [ 0.        +0.j],
                [ 0.        +0.j]],

               [[ 1.        +0.j],
                [ 0.        +0.j],
                [ 0.        +0.j],
                [ 0.        +0.j]],

               [[ 0.        +0.j],
                [ 0.        +0.j],
                [ 0.70710678+0.j],
                [ 0.70710678+0.j]]])
```

```
In [48]: crq = Qobj(array(inter))
         crq
```

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

TypeError                                 Traceback (most recent call last)

<ipython-input-48-fb8a58b562e3> in <module>()
----> 1 crq = Qobj(array(inter))
      2 crq


/anaconda3/envs/qutip-env/lib/python3.6/site-packages/qutip/qobj.py in __init__(self, i
277             do_copy = copy
278             if not isinstance(inpt, fast_csr_matrix):
--> 279                 _tmp = sp.csr_matrix(inpt, dtype=complex, copy=do_copy)
280                 _tmp.sort_indices() #Make sure indices are sorted.
281                 do_copy = 0


/anaconda3/envs/qutip-env/lib/python3.6/site-packages/scipy/sparse/compressed.py in __i
 77                             self.format)
 78             from .coo import coo_matrix
---> 79             self._set_self(self.__class__(coo_matrix(arg1, dtype=dtype)))
 80
 81         # Read matrix dimensions given, if any


/anaconda3/envs/qutip-env/lib/python3.6/site-packages/scipy/sparse/coo.py in __init__(s
```

```
    179
    180                 if M.ndim != 2:
--> 181                     raise TypeError('expected dimension <= 2 array or matrix')
    182                 else:
    183                     self._shape = check_shape(M.shape)


    TypeError: expected dimension <= 2 array or matrix
```

In [55]:
```
crq = Qobj(block([inter[0], inter[1], inter[2], inter[3]]))
```

In [56]:
```
crq
```

Out[56]:

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

$$\begin{pmatrix} 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.707 & 0.0 & 0.0 & 0.707 \\ -0.707 & 0.0 & 0.0 & 0.707 \end{pmatrix}$$

In [57]:
```
ceq
```

Out[57]:

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

$$\begin{pmatrix} -1.000 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

In [58]:
```
crq*ceq*(crq.dag())
```

Out[58]:

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

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.000 \\ 0.0 & 0.0 & 1.000 & 0.0 \end{pmatrix}$$

In [60]:
```
cnot()
```

Out[60]:

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

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

In [59]:
```
(crq.dag())*ceq*crq
```

Out[59]:
Quantum object: dims = [[4], [4]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.000 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

In [62]:
```
 = tensor(sigmax(), sigmax()) + tensor(sigmay(), sigmay())
```

Out[62]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 2.0 & 0.0 \\ 0.0 & 2.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

In [63]:
```
(-1j*(1/8)*).expm()
```

Out[63]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.969 & -0.247j & 0.0 \\ 0.0 & -0.247j & 0.969 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

In [64]:
```
(1j*(1/8)*).expm()
```

Out[64]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.969 & 0.247j & 0.0 \\ 0.0 & 0.247j & 0.969 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

In [65]:
```
(1j*(1/4)*).expm()
```

Out[65]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.878 & 0.479j & 0.0 \\ 0.0 & 0.479j & 0.878 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

In [66]:
```
(1j*(pi/4)*).expm()
```

Out[66]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

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

In [67]:
```
(-1j*(pi/8)*).expm()
```

Out[67]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.707 & -0.707j & 0.0 \\ 0.0 & -0.707j & 0.707 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

In [69]:
```
 = tensor(sigmax(), sigmax())
```

Out[69]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

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

In [70]:
```
(1j*(pi/4)*).expm()
```

Out[70]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.707 & 0.0 & 0.0 & 0.707j \\ 0.0 & 0.707 & 0.707j & 0.0 \\ 0.0 & 0.707j & 0.707 & 0.0 \\ 0.707j & 0.0 & 0.0 & 0.707 \end{pmatrix}$$

In [75]:
```
one = basis(2,0)
one
```

Out[75]:
Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket

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

In [76]:
```
zero = basis(2,1)
zero
```

Out[76]:
Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket

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

In [78]:
```
aas = (1/sqrt(2))*(tensor(one, zero) + 1j*tensor(zero, one))
```

In [79]:
```
aas
```

Out[79]:
Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket

$$\begin{pmatrix} 0.0 \\ 0.707 \\ 0.707j \\ 0.0 \end{pmatrix}$$

In [80]:
```
ket2dm(aas)
```

Out[80]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

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

In [70]:
```
(1j*(pi/4)*).expm()*
```

Out[70]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.707 & 0.0 & 0.0 & 0.707j \\ 0.0 & 0.707 & 0.707j & 0.0 \\ 0.0 & 0.707j & 0.707 & 0.0 \\ 0.707j & 0.0 & 0.0 & 0.707 \end{pmatrix}$$

In [ ]:
```
#g = pi/(4 * T) T = 2 * np.pi
#H = g * (tensor(sigmax(), sigmax()) + tensor(sigmay(), sigmay()))
```

In [84]:
```
H = (1/(4 * 2 ))* (tensor(sigmax(), sigmax()) + tensor(sigmay(), sigmay()))
```

In [85]:
```
(-1j*H*2*pi).expm()
```

Out[85]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

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

In [86]:
```
(1j*H*2*pi).expm()
```

Out[86]:

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

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

In [87]:
```
(H*2*pi).expm()
```

Out[87]:

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

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 2.509 & 2.301 & 0.0 \\ 0.0 & 2.301 & 2.509 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$