# Wolfram Mathematica
# Lindblad Operator Package Tutorial

Devesh Karthik
Department of Physics, University of Connecticut, Storrs, CT

June 5, 2024

## 1 Introduction

The Lindblad master equation

$$\frac{d\rho}{dt} = -i\left[H,\rho\right] + \sum_i \gamma_i [L_i \rho L_i^\dagger - \frac{1}{2}\left\{L_i^\dagger L_i, \rho\right\}] \tag{1}$$

models the time evolution of open quantum systems using the density matrix $\rho$, the Hamiltonian of the system $H$, dissipators $L_i$, and amplitudes of dissipation $\gamma_i$.

The Lindblad operator $\mathcal{L}$, otherwise known as the Lindbladian, acts upon $\rho$ as the matrix representation of the right-hand side of the Lindblad master equation. We can use the Lindblad Operator package in Mathematica to create the Lindbladian.

In this tutorial, we will consider an open system of the following qualities:

1. Hamiltonian: $H = K{a^\dagger}^2 a^2 - \epsilon_2\left({a^\dagger}^2 + a^2\right)$
   - Dimension: $N = 15$
   - Kerr Non-Linearity Parameter: $K = 1$
   - Control Parameter: $\epsilon_2 = 3$
2. Dissipation: $\kappa = 0.1$
3. Mean Number of Photons in Environment: $\eta = 0.1$
4. Dissipators: $L_1 = a, L_2 = a^\dagger$
5. Amplitudes of Dissipation: $\gamma_1 = \kappa\left(\eta + 1\right), \gamma_2 = \kappa\eta$
6. Initial state: Ground state $\phi_0$.

## 2 Preparing the System

We begin by defining the constants of our system. For simplicity, we let $\hbar = 1$.

```
dim = 15
kerr = 1
e2 = 3
Kappa = 0.1
nth = 0.1
g1 = Kappa*(nth + 1)
g2 = Kappa*nth
gs = {g1, g2}
h = 1
```

We can now define our Hamiltonian as a matrix of dimension $15 \times 15$.

```
Do[bb[k] = h k, {k, 0, dim - 1}]
Do[Do[HHo[i, j] = 0., {j, 0, dim - 1, 1}], {i, 0, dim - 1, 1}]
HHo[0, 0] = 0
Do[HHo[k, k] = kerr bb[k] (bb[k] - 1), {k, 0, dim - 1, 1}]
Do[HHo[k, k + 2] = -e2 Sqrt[(k + 1) (k + 2)]; HHo[k + 2, k] = HHo[k, k + 2],
{k, 0, dim - 3, 1}]
H = Table[Table[HHo[i, j], {j, 0, dim - 1, 1}], {i, 0, dim - 1, 1}]
```

Finally, we define our dissipators: the creation and annihilation operators. The annihilation operator is defined by first creating a $15 \times 16$ dimensional matrix with the elements of the shifted diagonal as the square-root of the row number. Then, we remove the last column to create our annihilation operator in a Hilbert space of size 15.

```
a = Normal[SparseArray[Table[{i, i + 1} -> Sqrt[i], {i, 1, dim, 1}]]][[1;;-1,1;;-2]]
```

Since the creation operator is the hermitian conjugate of the annihilation operator, and because all elements of the annihilation operator are real, we can take the transpose of $a$ to get $a^\dagger$. We then package both $a$ and $a^\dagger$ for ease of use.

```
ad = Transpose[a]
Cs = {a, ad}
```

## 3 The Lindblad Operator Package

The Lindblad Operator package defines the Lindbladian as a matrix of dimension $N^2 \times N^2$ and exports it into a `.dat` file as a list of $N^4$ complex numbers. We can open the package by running the command `Import["https://bit.ly/3V17kNv"]`.

The `LINDBLAD` function employs the same structure of the Eq. (1) and takes 4 inputs to create the Lindbladian operator: $H$ the Hamiltonian as an $N \times N$ matrix, $Cs$ a list of the dissipators $L_i$, $gs$ a list of the corresponding amplitudes $\gamma_i$, and the value of $\hbar$. The code below successfully creates the Lindbladian operator from our example.

```
LINDBLAD[H, Cs, gs, h]
```

If continuing to work in Mathematica, calling the command `LINDBLADN` is sufficient to access the Lindbladian defined above.

However, if migrating to a different environment, we need to access the file `L.dat` that contains the elements of the Lindbladian. If no directory was specified, the file saves in the Documents folder. To specify a directory, simply run the function

```
SetDirectory["\Target\Directory\Here"]
```

with the desired file path inside the quotations before defining the Lindbladian through the `LINDBLAD` function.

## 4 Importing the Lindbladian into Python

The information of the Lindbladian is now stored in the `L.dat` file. To access it in Python, we begin by moving the file into the same location as our Python program. Then, we run the following code to save the content of the file as a Python variable `object`.

```
file = open("L.dat", "r")
object = file.read()
file.close()
```

Since `object` is a string, the information inside cannot be accessed directly. Thus, in order to create our matrix, we must convert our string of information into an array of dimension $N^2 \times N^2$.

We first replace the imaginary number representation in Mathematica (represented by I) with the format Python recognizes as a complex number.

```
object = object.replace("I", f"1j")
```

Then, we cut out extra spaces surrounding operators $+$ and $-$, replace every indent and newline with a space, and finally split the data at every space.

```
object = object.replace(" + ","+").replace(" - ","-")
data = list(object.replace("\t", " ").replace("\n", " ").split())
```

The variable `data` is an array of size $N^4$ containing a string representation of each element in the Lindbladian. We can now iterate through each element non-zero element of the array and convert our string into usable numerical data with the `eval()` function.

```
import numpy as np
rows = []
for i in range(len(data)):
    if data[i] != "0.":
        rows.append(i)
datas = []
for i in range(len(rows)):
    datas.append(np.complex128(eval(data[rows[i]])))

lind = np.zeros(len(data), dtype=complex)
for i in range(len(rows)):
    lind[rows[i]] = datas[i]
```

The variable `lind` now holds an array of length $N^4$ containing each element of the lindbladian. We can resize this into a square matrix of dimensions $N^2 \times N^2$ to represent our Lindblad operator.

```
dim = int(np.sqrt(len(data)))
lind = lind.reshape(dim, dim)
```

In our example, `lind` is now of dimensions $225 \times 225$.

## 5 Calculating $T_X$ in Python

An important quantity that can be derived from the Lindbladian is $T_X$, which represents the decay rate of the system's survival probability. This value is calculated through a method known as Singular Value Decomposition, which provides an exact representation of a matrix through factorization.

We can determine $T_X$ with the following code:

```
from numpy.linalg import svd
Tx = 1/min(svd(lind)[1][:-1])
```

For the example we have been following, $T_X = 100.46$. Though the full implications of this value are not clear yet, it is certain that the $T_X$ value of a lindbladian plays a critical role in further understanding open quantum system dynamics.

## 6 Exercise

For the system mentioned in the Introduction, graph the survival probability in the time span $0 \leq t \leq 10$ using your environment of choice (Mathematica, Python, Julia, Fortran, etc).