The goal is to watch how energy flows in a nanoparticle/molecule system upon illumination with light. We represent the nanoparticle and the molecule system with level density matrices where the levels represent the ground state and excited states in the system.. The diagonal density matrix element gives the probability that the nanoparticle will be found in the ground state, gives the probability that the nanoparticle will be found in the 2nd excited state, etc.

**Note: Remember in our C code, there are two important differences in how these matrices are stored.**

1. **We start indexing from 0, not from 1**
2. **We store**  **matrices as vectors of length**

Consequently, there is a mapping between how we think of the matrices naturally and how we store their data on the computer.

Consider the following example of storing on a computer when is a 2x2 matrix:

Notice that each element in our math notation, , maps to our array in C , , in the following way:

where in the current example. This mapping will apply for any size matrix as long as the dimension is specified.

**Equations of Motion**

The central equations solved by the computer are the Lioville-Lindblad equations of motion for the density matrices of the nanoparticle and the molecule:

where

and

The equations are coupled together by the terms. Solving Equation (1) updates to , and similarly for Equation (2).

Importantly, requires calculation of (which depends on and requires calculation of (which depends on . This means, that every time we solve Equation 1 to update , we must update **before solving Equation (2)**. Similarly, after Equation (2) is solved to update , must be updated before solving Equation (1) again.

Therefore, the outline of the coupled update procedure is the following:

1. RK3 update for under
2. Update and
3. RK3 update for under
4. Update and
5. Back to a. until max number of updates has occurred.