Notes on Kronecker Products from Quantum Hamiltonians

Matthew Otten

January 19, 2017

Abstract

Basic notes on utilizing the structure of Kronecker products for quantum Hamiltonians.

1 Introduction

When calculating a coupled quantum system, the total Hilbert space is a Kronecker product of the Hilbert spaces of the individual systems. This is at the heart of the difficulty in calculating a large number of coupled quantum systems; the total Hilbert space size grows exponentially. For tractable Hilbert spaces, we can utilize the structure of Kronecker products to accelerate solutions of coupled quantum Hamiltonians.

2 Kronecker Product Basics

Most of the following is taken from Wikipedia. The Kronecker product between two matrices $\mathbf{A}, n \times m$ and $\mathbf{B}, p \times q$ is defined as

$$\mathbf{M} = \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{00}\mathbf{B} & \cdots & a_{0n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m0}\mathbf{B} & \cdots & a_{mn}\mathbf{B}. \end{bmatrix} \tag{1}$$

The resulting matrix is size $np \times mq$. Within this paper, we will only concern ourselves with square matrices, $m = n, q = p$. Since we will be interested in generating these Kronecker products within code, we outline the algorithm for generating $\mathbf{M}$ here.

```
do i_a=0,n
  do j_a=0,n
    do i_b=0,p
      do j_b=0,p
        M(p*i_a + i_b,p*j_a+j_b) = A(i_a,j_a)*B(i_b,j_b)
      end
    end
  end
end
```

Here, we list several useful Kronecker product identities. The Kronecker product is bilenear and associative:

$$\mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) = \mathbf{A} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{C} \tag{2}$$

$$(\mathbf{A} + \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes \mathbf{C} + \mathbf{B} \otimes \mathbf{C} \tag{3}$$

$$(k\mathbf{A}) \otimes \mathbf{B} = \mathbf{A} \otimes (k\mathbf{B}) = k(\mathbf{A} \otimes \mathbf{B}) \tag{4}$$

$$(\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) \tag{5}$$

Generally, the Kronecker product is non-commutative. The *mixed-product* property states

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}). \tag{6}$$

Furthermore, if $\mathbf{A}$ and $\mathbf{B}$ are invertible, then

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}. \tag{7}$$

We can also represent the multiplication of three matrices using the Kronecker product:

$$\mathbf{ABC} = (\mathbf{C}^{\mathbf{T}} \otimes \mathbf{A})vec(\mathbf{B}), \tag{8}$$

where $vec(\mathbf{B})$ takes the columns of $\mathbf{B}$ and stacks them, creating a single column vector.

Courtesy of Paul Fischer, there are other identities concerning the inversion of $\mathbf{C} = (\mathbf{A} \otimes I + I \otimes \mathbf{B})$, which we will add later.

3 Construction of Coupled Quantum Hamiltonians

To construct coupled quantum Hamiltonians, we must first construct the operators for each subsystem. Each subsystem has its own creation and annihilation operators; those are sufficient to define most (all?) Hamiltonians involving that subsystem. Since the creation and annihilation operators are related by the adjoint operation, we only truly need to describe the annihilation operator. For

example, given a subsystem with n_a levels, let's define the annihilation operator,

$$a = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \sqrt{3} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ z0 & 0 & 0 & 0 & \cdots & 0 & \sqrt{n_a - 1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{9}$$

a is a matrix of size $n_a \times n_a$. We can create a generating function which can create this matrix using only the number of levels, since we know, *a priori*, that the only nonzeros will be on the super-diagonal, and the value is always $\sqrt{i+1}$, where i is the row index (counting from 0), and with corresponding column index, $j = i+1$. The creation operator can also be defined by transposing the i and j of the annihilation operator, and the number operator $a^\dagger a$ has a similarly simple definition, where the value is simply i, and corresponding column index $j = i$. This allows us, regardless of the number of levels of the subsystem, to store only a single integer, the number of levels, that uniquely defines the annihilation operator.

A couple Hamiltonian has multiple subsystems, possibly with different numbers of levels. Constructing the full Hamiltonian involves taking Kronecker products of our base operators to obtain their representation in the full Hilbert space. For example, let us take three operators, a, b, and c, with levels n_a, n_b, and n_c, respectively. The full Hilbert space size is then $N = n_a n_b n_c$. We will always use the convention that a, b, and c represent the operators within their own (small) Hilbert space, and $\tilde{a}, \tilde{b}$, and $\tilde{c}$ represent the operators in the total Hilbert space, where

$$\begin{aligned} \tilde{a} &= a \otimes I_b \otimes I_c, \\ \tilde{b} &= I_a \otimes b \otimes I_c, \\ \tilde{a} &= I_a \otimes I_b \otimes c, \end{aligned} \tag{10}$$

and I_x is the identity matrix of size n_x. An everpresent term in many Hamiltonians is the number operator, $\tilde{a}^\dagger \tilde{a}$. We note that since a and $a^\dagger$ both reside in the same subspace, the combination of the two operators can be done within that subspace, leading to

$$\tilde{a}^\dagger \tilde{a} = a^\dagger a \otimes I_{bc}. \tag{11}$$

We have used the fact that $I_b \otimes I_c = I_{bc}$, the identity matrix of size $n_b n_c$. Combining terms from different subspaces, such as that needed in a coupling term like $\tilde{a}\tilde{b}^\dagger$, is also quite simple:

$$\begin{aligned} \tilde{a}\tilde{b}^\dagger &= (a \otimes I_{bc})(I_a \otimes b^\dagger \otimes I_c) \\ &= a I_a \otimes I_{bc}(b^\dagger \otimes I_c) \\ &= a \otimes b^\dagger \otimes I_c. \end{aligned} \tag{12}$$

This follows from the mixed-product property, eq. 6. Interestingly,

$$\begin{aligned}\tilde{b}^\dagger \tilde{a} &= (I_a \otimes b^\dagger \otimes I_c)(a \otimes I_{bc}) \\ &= I_a a \otimes (b^\dagger \otimes I_c) I_{bc} \\ &= a \otimes b^\dagger \otimes I_c,\end{aligned} \tag{13}$$

which simply shows that operators from different subspaces commute.

4 Efficiently Generating Hamiltonians in the Full Hilbert Space

So far, we have described generating operators within their subspace and given forms for the combination of operators in the full Hilbert space. We also want to generate operators in the full Hilbert space in both a space- and time-efficient manner, since the full Hilbert space is much, much larger than the subspaces.

4.1 Operators from a Single Subspace

First, let us define the algorithm for generating $\tilde{a} = a \otimes I_{bc}$. There are several simplifications from the general algorithm described in section 2, the foremost being that we are taking a Kronecker product with the identity matrix, which is diagonal and has entries of value one. Furthermore, a is superdiagonal, with a simple generating function. This leads to the following algorithm for generating $\tilde{a}$:

```
do k1=0,n_b*n_c
  do i=0,n_a-1
    a_tilde(n_b*n_c*i+k1,n_b*n_c*(i+1)+k1) = sqrt(i+1)
  end
end
```

Here, we have made use of several facts previously identified in this paper. First, we used the generating function for a, where, given row index i, we have column index $j = i+1$ and value $\sqrt{i+1}$. The loop is only of size $n_a - 1$ because the superdiagonal only hase $n_a - 1$ elements. We have also used the diagonal structure of I_{bc} to reduce the two outer loops of the general algorithm into one loop. With these two simplifications, our generating function only touches the nonzero elements, doing the minimum number of calculations needed. Furthermore, we are able to generate the much larger $\tilde{a}$ from only three integer values: n_a, n_b, and n_c, the number of levels of each subsystem.

Now, let us define a similar algorithm for $\tilde{c}$.

```
do k2=0,n_a*n_b
  do i=0,n_c-1
    c_tilde(n_c*k2+i,n_c*k2+(i+1)) = sqrt(i+1)
  end
end
```

We have used similar tricks here; the only difference is that the position of the identity matrix has switch from being after the operator to being before the operator. Nonetheless, judicious application of the general algorithm with the diagonal and superdiagonal structure leads to a similarly simple, minimal algorithm.

Finally, we now describe $\tilde{b}$. This is slightly more complicated then $\tilde{a}$ and $\tilde{b}$ because we now have identity matrices both before and after the operator, b. Rather than define the general algorithm for $\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}$, which would certainly lead to the answer, we will combine the above algorithms for $\tilde{a}$ and $\tilde{c}$. This leads to the following algorithm for generating $\tilde{b} = I_a \otimes b \otimes I_c$:

```
do k1=0,n_c
  do k2=0,n_a
    do i=0,n_b-1
      b_tilde(i*n_c+k1+k2*n_b*n_c,(i+1)*n_c+k1+k2*n_b*n_c) = sqrt(i+1)
    end
  end
end
```

We note that the algorithm for $\tilde{b}$ is general in the sense that it can arbitrarily define any size identity matrix before and after an operator, and, because $I_b \otimes I_c = I_{bc}$, it is sufficient to generate the full Hilbert space representation of any single subspace operator, regardless of how many total operators there are. For example, take some operator

$$\begin{aligned} \tilde{d} &= I_a \otimes I_b \otimes I_c \otimes d \otimes I_e \otimes I_f \\ &= I_{abc} \otimes d \otimes I_{ef}. \end{aligned} \tag{14}$$

We can simply use the algorithm for $\tilde{b}$, but now we have a larger n_{before} and n_{after}.

4.2 Operators from Multiple Subspaces

If we were only concerned with operators from a single subspace, as was described in the previous section, where would be no reason to use a combined Hilbert space - all subsystems would be independent and the full dynamics could be described by the dynamics of the smaller, subspace Hamiltonians only. Combining operators from different subspaces allows for interesting physics to develop from the coupling. We now describe how to efficiently generate coupling terms, such as $\tilde{a}\tilde{b}^\dagger$.

As we saw in eq. 13, $\tilde{a}\tilde{b}^\dagger = a \otimes b^\dagger \otimes I_c$. First, let us focus just on $a \otimes b^\dagger$; the Kronecker product with I_c could then be done using the algorithms in the previous section. Both a and b are superdiagonal within their subspaces with associated generating functions. Using this, we define the following algorithm to generate $a \otimes b$,

```
do i_a = 0,n_a-1
  do i_b = 0,n_b-1
    a_bdag_tilde(n_b*i_a+i_b,n_b*(i_a+1)+(i_b+1)) = sqrt(i_a+1)*sqrt(i_b+1)
  end
end
```

In many cases, the operators will have some identity matrix between them, such $\tilde{a}\tilde{c}^\dagger = a \otimes I_b \otimes c$. In this case, we can first do $I_b \otimes c$, followed by $a \otimes (I_b \otimes c)$.

```
do i_a = 0,n_a-1
  do k3 = 0,n_b
    do i_c = 0,n_c-1
      a_cdag_tilde(n_c*n_b*i_a+n_c*k3+i_c,n_c*n_b*(i_a)+n_c*k3+(i_c+1))
                  = sqrt(i_a+1)*sqrt(i_c+1)
    end
  end
end
```

Though the indices are starting to get a little complex, we are still generating the full space representations from just a few integers. Combining the above algorithm with the final algorithm from the previous section describing $I_a \otimes b \otimes I_c$ is sufficient to fully generate all one- and two-operator Hamiltonian terms. We do not explicitly show that here both because it is cumbersome, and because our implementation of these algorithms utilizes a subroutine that specifically calculates $I_{before} crossx cross I_{after}$, given a value from x, that values i, j pair and the number of levels of x. This allows for efficient reuse of this subroutine both in one- and two-operator generations, as well as allows for less cumbersome index calculations. One- and two-operator terms make up many of the most interesting Hamiltonians, such as the Hubbard model, Jaynes-Cummings, etc, and, as such, we do not extend this analysis to three or more operator terms, though it is rather trivial (albeit tiresome) to do.

4.2.1 Structure of Combined Operators

TODO: Describe combinations of sub and superdiagonal becoming another sub or superdiagonal. Offsetnew = level2*nbetween*offset1 + offset2

These algorithms are extremely space efficient. To describe the Hamiltonian in the full Hilbert space of m operators requires only m integers to be stored, regardless of the full Hilbert space size. For example, if we wanted to combine 10 operators, each with 10 levels, the full Hilbert space size would be 10^{10}, or 10 billion. We can effectively build that Hamiltonian from only 10 integers. This points to an incredible memory savings, even over storing the Hamiltonian sparsely. These algorithms can be used to develop a matrix-free algorithm for time-stepping or steady-state solutions. If the matrix is desired, this is the most? efficient way to generate it.

5 Lindblad Operators and Superoperator Space

We have thus far described how to efficiently generate Hamiltonians. These are sufficient for solving pure state dynamics within the Schrodinger equation. In many cases, simulating open quantum systems is necessary. One way of doing this is the Liouville matrix master equation,

$$\dot{\rho} = -i[H, \rho] + L(C)\rho = -i(H\rho - \rho H) + L(C)\rho, \tag{15}$$

where H is the Hamiltonian and $L(C)\rho$ is a Lindblad superoperator,

$$L(C)\rho = C\rho C^\dagger - \frac{1}{2}(C^\dagger C\rho + \rho C^\dagger C). \tag{16}$$

Generally, there could be many Lindblad superoperators, one for each dissipative channel in the system, but we used only one here for brevity's sake. The previous section described how to generate H efficiently, but $L(C)$ cannot similarly be represented as a matrix with the same dimensions as H, say $N \times N$, as it is a superoperator (an operator which acts on the space of operators). This can also be seen from the first term in $L(C)$, $C\rho C^\dagger$, which certainly cannot be represented by a simple matrix of the same dimensions as C.

If we work in superoperator space, however, we can represent $L(C)$ as a matrix of dimension $N^2 \times N^2$. Additionally, we can represent $[H, \rho]$ as a matrix. To construct these superoperator matrices, we make use of eq. 8, which describes how to transform matrix products into a (long) vecter multiplied by a Kronecker product. For example,

$$H\rho I_N - I_N\rho H = (I_N \otimes H - H \otimes I_N)vec(\rho), \tag{17}$$

where we inserted an identity matrix to be able to make use of eq. 8. For the Lindblad, we have

$$L(\tilde{C})\rho = (\tilde{C} \otimes \tilde{C} - \frac{1}{2}(I_N \otimes \tilde{C}^\dagger\tilde{C} + \tilde{C}^\dagger\tilde{C} \otimes I_N)vec(\rho). \tag{18}$$

This allows us to fully transform the master equation, eq. 15, into

$$\begin{aligned} \dot{\tilde{\rho}} &= A\tilde{\rho} \\ &= \Big(- i(I_N \otimes H - H \otimes I_N) + \tilde{C} \otimes \tilde{C} - \frac{1}{2}(I_N \otimes \tilde{C}^\dagger\tilde{C} + \tilde{C}^\dagger\tilde{C} \otimes I_N)\Big)\tilde{\rho}, \end{aligned} \tag{19}$$

where we have defined $\tilde{\rho} = vec(\rho)$.

The superoperator matrix, A, may seem unwieldy because of its greatly increased size. While it may be much bigger, we can utilize the algorithms of section **??** to efficiently generate all of the terms. Terms of the form $I_N \otimes X$ or $X \otimes I_N$ simply increase the sizes of the identity matrices involved, but do not change the algorithm in anyway. The only other term, $\tilde{C} \otimes \tilde{C}$ is also quite simple,

$$\tilde{C} \otimes \tilde{C} = I_{bef} \otimes C \otimes I_{af} \otimes I_{bef} \otimes C \otimes I_{af}, \tag{20}$$

where I_{bef} is the Hilbert space size before operator C and I_{af} is the Hilbert space after. This term can be simply generated with the algorithm describing $a \otimes I_b \otimes c$ (with the addition of I_{bef} and I_{af}, of course). Efficiently constructing A in this way can, similar to the previous section, allow for space-efficient matrix-free methods, or can allow for the construction of the (very, very sparse) A to be input into some linear solver which utilizes a preconditioner based off the matrix A. Since the best preconditioner for A is, of course, A^{-1}, and we have an incredible amount of structure in our matrix A, it is natural to ask if we can efficiently construct A^{-1}, or at least a very good approximation.

6 Efficient Generation of the Inverse of A