UniDyn--Study-01.nb

John A. Marohn
jam99@cornell.edu
Cornell University

Abstract: This demonstration notebook loads the **UniDyn** package and executes the package's unit tests.

Set the path to the package

Check the Mathematica version number .

```
In[1]:= $VersionNumber
Out[1]= 12.3
```

Tell *Mathematica* the path to the directory containing the packages.

EDIT THE FOLLOWING PATH STRING:

```
In[2]:= $UniDynPath =
          "/Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/
            unidyn";
```

YOU SHOULD NOT NEED TO EDIT ANYTHING FROM HERE ONWARDS.

Load the package

Append the package path to the system path. Before trying to load the package, ask *Mathematica* to find it. This is a test that we directed *Mathematica* to the correct directory. The output of this command should be the full system path to the UniDyn.m file.

```
In[3]:= $Path = AppendTo[$Path, $UniDynPath];
        FindFile["UniDyn`"]
Out[4]= /Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/unidyn/UniDyn.m
```

Now that we are confident that the path is set correctly, load the package. Setting the global $VerboseLoad variable to True will print out the help strings for key commands in the package.

In[5]:=
```
$VerboseLoad = True;
Needs["UniDyn`"]
```

CreateOperator: CreateOperator[] is used to batch–define a bunch of operators. Example: CreateOperator[{{Ix, Iy, Iz},{Sx,Sy,Sz}}] will create six operators, where each of the operators in the first list will commute with each of the operators of the second list.

CreateScalar: CreateScalar[list] is used to batch–define a bunch of scalars. The parameter list can be a single scalar or a list of scalars. Example: CreateScalar[{w1,w2}].

NCSort: NCSort[list] sorts the operators in list into canonical order.

SortedMult: SortedMult[list] returns Mult[list$ordered], where list$ordered are the elements of list sorted into canonical order.

MultSort: MultSort[NonCommutativeMultiplyt[list]] returns returns NonCommutativeMultiply[list$ordered], where list$ordered are the elements of list sorted into canonical order.

Comm: Comm[a,b] calculates the commutator of two operators.

SpinSingle$CreateOperators: SpinSingle$CreateOperators[Ix,Iy,Iz,L] creates Ix, Iy, and Iz angular momentum operators and defines their commutation relations. When the total angular momentum L = 1/2, additional rules are defined to simplify products of the angular momentum operators. When the total angular momentum L is unspecified, no such simplification rules are defined.

OscSingle$CreateOperators: OscSingle$CreateOperators[aL,aR] creates a raising operator aR and a lowering operator aL for single harmonic oscillator and defines the operator commutation relations.

Evolve: Evolve[H, t, ρ] represents unitary evolution of the density operator ρ for a time t under the Hamiltonian H. This function expands according to simplification rules but leaves the evolution unevaluated.

Evolver: Evolver[H, t, ρ(0)] calculates ρ(t) = Exp[–I H t] ρ(0) Exp[+I H t], assuming that H is time independent, according to the commutation rules followed by ρ(0) and H.

Execute the units tests in batch

Included with the package are a number of files, ending in “-tests.m”, that contain tests of the package’s functions -- so-called unit tests. Set the working directory to the package directory .

In[7]:=
```
SetDirectory[$UniDynPath];
```

Get the names of all the unit-testing files included with the package, following my convention that the unit testing file end in “-tests.m”).   Carry out the unit tests. This takes a second.

In[8]:=
```
fn = FileNames["*-tests.m"];
test$report = TestReport /@ fn;
TableForm[Table[test$report[[k]], {k, 1, Length[test$report]}]]
```

Out[10]//TableForm=

```
TestReportObject[ Title: Test Report: Comm–tests.m    Success rate: 100%   Tests run: 23 ]
TestReportObject[ Title: Test Report: Evolve–tests.m  Success rate: 100%   Tests run: 23 ]
TestReportObject[ Title: Test Report: Mult–tests.m    Success rate: 100%   Tests run: 18 ]
TestReportObject[ Title: Test Report: OpQ–tests.m     Success rate: 100%   Tests run: 21 ]
TestReportObject[ Title: Test Report: Osc–tests.m     Success rate: 100%   Tests run: 22 ]
TestReportObject[ Title: Test Report: Spins–tests.m   Success rate: 100%   Tests run: 14 ]
```

Explore non-commutative multiplication

In[11]:=
```
Clear[Ix, Iy, Iz, aL, aR, ω, Δ];

CreateOperator[{{Ix, Iy, Iz}, {aL, aR}}];
SpinSingle$CreateOperators[Ix, Iy, Iz, 1 / 2];
OscSingle$CreateOperators[aL, aR];

CreateScalar[{ω, Δ}];
$Assumptions = {Element[ω, Reals], Element[Δ, Reals]};
```

SpinSingle$CreateOperators: Spin operators already exist.

SpinSingle$CreateOperators: Adding spin commutations relations.

SpinSingle$CreateOperators: Angular momentum L = 1/2. Adding operator simplification rules.

OscSingle$CreateOperators: Oscillator operators already exist.

OscSingle$CreateOperators: Adding oscillator commutations relations.

In[17]:=
```
Evolver[ω Ix, t, Iz, quiet → False]
```

ρ matrix = $\begin{pmatrix} Iz \\ -Iy\,\omega \\ -Iz\,\omega^2 \\ Iy\,\omega^3 \\ Iz\,\omega^4 \end{pmatrix}$

Ω = $\begin{pmatrix} 0 & -\omega^2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

system of equations = ({Private`x4$15173′[Private`time$15173] == $-\omega^2$ Private`x3$15173[Privat

1st solution = Private`x1$15173 → Function[{Private`time$15173},
Iz Cos[Private`time$15173 ω] - Iy Sin[Private`time$15173 ω]]

1st solution w/ substitution = Iz Cos[t ω] - Iy Sin[t ω]

Out[17]= Iz Cos[t ω] - Iy Sin[t ω]

We can get the solution "by hand" this way:

In[18]:=
```
MatrixExp[( 0  -ω^2  0  0
            1   0    0  0
            0   1    0  0
            0   0    1  0 ) t] . ( Iy ω^3
                                   -Iz ω^2
                                   -Iy ω
                                    Iz ) // ExpToTrig // FullSimplify //
  Collect[#, {Ix, Iy, Iz}] &
```

Out[18]= {Iy ω^3 Cos[t ω] + Iz ω^3 Sin[t ω], -Iz ω^2 Cos[t ω] + Iy ω^2 Sin[t ω],
-Iy ω Cos[t ω] - Iz ω Sin[t ω], Iz Cos[t ω] - Iy Sin[t ω]}

The solution we want is the last element , Iz Cos[t ω]-Iy Sin[t ω].

In[19]:=
```
Evolver[Δ Iz + ω Ix, t, Ix, quiet → False]
```

$$\rho\ \texttt{matrix} = \begin{pmatrix} \texttt{Ix} \\ \texttt{Iy}\,\Delta \\ -\Delta\,(\texttt{Ix}\,\Delta - \texttt{Iz}\,\omega) \\ -\texttt{Iy}\,\Delta\,(\Delta^2+\omega^2) \\ \Delta\,(\texttt{Ix}\,\Delta - \texttt{Iz}\,\omega)\,(\Delta^2+\omega^2) \end{pmatrix}$$

$$\Omega = \begin{pmatrix} 0 & -\Delta^2-\omega^2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\texttt{system of equations} = \left(\{\texttt{Private`x4\$15807}'[\texttt{Private`time\$15807}] == (-\Delta^2-\omega^2)\ \texttt{Private`x3\$15807}[\texttt{P} \right.$$

$$\texttt{1st solution} = \texttt{Private`x1\$15807} \to$$

$$\texttt{Function}\Big[\{\texttt{Private`time\$15807}\}, \frac{1}{2\sqrt{-\Delta^2-\omega^2}\,(\Delta^2+\omega^2)}\, e^{-\texttt{Private`time\$15807}\sqrt{-\Delta^2-\omega^2}} \Big(-\texttt{Iy}\,\Delta^3 + e^{2\,\texttt{Private`time\$15807}\sqrt{-\Delta^2-\omega^2}}\,\texttt{Iy}\,\Delta^3 - \texttt{Iy}\,\Delta\,\omega^2 + e^{2\,\texttt{Private`time\$15807}\sqrt{-\Delta^2-\omega^2}}\,\texttt{Iy}\,\Delta\,\omega^2 + \texttt{Ix}\,\Delta^2\sqrt{-\Delta^2-\omega^2} + e^{2\,\texttt{Private`time\$15807}\sqrt{-\Delta^2-\omega^2}}\,\texttt{Ix}\,\Delta^2\sqrt{-\Delta^2-\omega^2} - \texttt{Iz}\,\Delta\,\omega\sqrt{-\Delta^2-\omega^2} + 2\,e^{\texttt{Private`time\$15807}\sqrt{-\Delta^2-\omega^2}}\,\texttt{Iz}\,\Delta\,\omega\sqrt{-\Delta^2-\omega^2} - e^{2\,\texttt{Private`time\$15807}\sqrt{-\Delta^2-\omega^2}}\,\texttt{Iz}\,\Delta\,\omega\sqrt{-\Delta^2-\omega^2} + 2\,e^{\texttt{Private`time\$15807}\sqrt{-\Delta^2-\omega^2}}\,\texttt{Ix}\,\omega^2\sqrt{-\Delta^2-\omega^2}\Big)\Big]$$

$$\texttt{1st solution w/ substitution} = \frac{1}{2\sqrt{-\Delta^2-\omega^2}\,(\Delta^2+\omega^2)}\, e^{-t\sqrt{-\Delta^2-\omega^2}} \Big(-\texttt{Iy}\,\Delta^3 + e^{2t\sqrt{-\Delta^2-\omega^2}}\,\texttt{Iy}\,\Delta^3 - \texttt{Iy}\,\Delta\,\omega^2 + e^{2t\sqrt{-\Delta^2-\omega^2}}\,\texttt{Iy}\,\Delta\,\omega^2 + \texttt{Ix}\,\Delta^2\sqrt{-\Delta^2-\omega^2} + e^{2t\sqrt{-\Delta^2-\omega^2}}\,\texttt{Ix}\,\Delta^2\sqrt{-\Delta^2-\omega^2} - \texttt{Iz}\,\Delta\,\omega\sqrt{-\Delta^2-\omega^2} + 2\,e^{t\sqrt{-\Delta^2-\omega^2}}\,\texttt{Iz}\,\Delta\,\omega\sqrt{-\Delta^2-\omega^2} - e^{2t\sqrt{-\Delta^2-\omega^2}}\,\texttt{Iz}\,\Delta\,\omega\sqrt{-\Delta^2-\omega^2} + 2\,e^{t\sqrt{-\Delta^2-\omega^2}}\,\texttt{Ix}\,\omega^2\sqrt{-\Delta^2-\omega^2}\Big)$$

Out[19]=
$$\frac{\omega\,(\texttt{Iz}\,\Delta + \texttt{Ix}\,\omega) + \Delta\,(\texttt{Ix}\,\Delta - \texttt{Iz}\,\omega)\,\texttt{Cosh}\left[t\sqrt{-\Delta^2-\omega^2}\right] - \texttt{Iy}\,\Delta\sqrt{-\Delta^2-\omega^2}\,\texttt{Sinh}\left[t\sqrt{-\Delta^2-\omega^2}\right]}{\Delta^2+\omega^2}$$

These two operators show up in a unitary evolution used to describe electron-transfer theory

In[20]:=
```
op1 = Δ Iz;
op2 = ω (Mult[aL, Iz] - Mult[Iz, aR]);
```

The Evolve function goes not recognize op2 as proprtional to op1, with the proportionality constant ω (aL - aR). However, if you take the commutator of the two terms, you get zero. This is one way to see they are proportional.

In[22]:=
```
Comm[op1, op2]
```

Out[22]= 0

We might want to sort the Mult first

In[23]:=
```
op2 /. Mult → SortedMult
```
Out[23]= ω (Mult[Iz, aL] - Mult[Iz, aR])

Note how the Divide operator behaves

In[24]:=
```
? Divide
```
Out[24]=

Symbol

x/y or Divide[x, y] is equivalent to $x\, y$^ −1.

In[25]:=
```
Clear[Ix, Iy, Iz]
```
In[26]:=
```
SpinSingle$CreateOperators[Ix, Iy, Iz];
```
SpinSingle$CreateOperators : Creating spin operators.

SpinSingle$CreateOperators : Adding spin commutations relations.

SpinSingle$CreateOperators : No angular momentum L defined.

The Divide operator handles simple cases .

In[27]:=
```
Divide[ω Ix, Ix]
```
Out[27]= ω

The Divide operator is not smart enough to cancel an operator in Mult[], the non-commutative multiply operator.

In[28]:=
```
Divide[Mult[Ix, Ix], Ix]
```
Out[28]= $\frac{\text{Mult[Ix, Ix]}}{\text{Ix}}$

In[29]:=
```
Clear[aL, aR]
```
In[30]:=
```
OscSingle$CreateOperators[aL, aR];
```
OscSingle$CreateOperators : Creating oscillator operators.

OscSingle$CreateOperators : Adding oscillator commutations relations.

Again, the Divide operator is not very smart .

In[31]:=
```
Divide[ω Mult[aL, Ix], Ix]
```
Out[31]= $\frac{\omega\, \text{Mult[aL, Ix]}}{\text{Ix}}$

Create an Inverse

We need an operator inverse! Start with bottom-out and empty cases.

In[32]:=
```
Clear[Inv];
Inv[a$sym_?ScalarQ] := 1 / a$sym
Inv[] := 1
```

In[35]:=
```
Inv[ω]
```

Out[35]= $\frac{1}{\omega}$

In[36]:=
```
Inv[Iz]
```

Out[36]= Inv[Iz]

The central property of the inverse is that the inverse of an operator times the operator is one. One way to define the inverse is as an upvalue.

In[37]:=
```
Iz /: Mult[Inv[Iz], Iz] := 1
Iz /: Mult[Iz, Inv[Iz]] := 1
```

In[39]:=
```
Mult[Inv[Iz], Iz]
```

Out[39]= 1

In[40]:=
```
Mult[Iz, Inv[Iz]]
```

Out[40]= 1

This works, but is a pain, because now you have to define upvalues for *every operator*. Is there a more general way to make the inverse work like this? Try these definitions:

In[41]:=
```
HoldPattern[ Mult[a$sym___, Inv[b$sym_?OperatorQ], b$sym_?OperatorQ, c$sym___]] :=
 Mult[a$sym, c$sym]
HoldPattern[Mult[a$sym___, b$sym_?OperatorQ,
   Inv[b$sym_?OperatorQ], c$sym___] ] := Mult[a$sym, c$sym]
```

The HoldPattern[] is important, or Inv[] will get evaluated before the pattern is defined. These patterns work as I would expect.

In[43]:=
```
Mult[Inv[Ix], Ix]
```

Out[43]= 1

In[44]:=
```
Mult[Ix, Inv[Ix]]
```

Out[44]= 1

In[45]:=
```
Mult[Inv[Ix], ω Ix]
```

Out[45]= ω

Mathematica does not know what to do with the following expression, because it does not yet know how to factor our scalars from the inverse.

In[46]:=
```
Mult[ Inv[ω Ix], Ix]
```
Out[46]=
```
Mult[Inv[Ix ω], Ix]
```

Try to explain to *Mathematica* how to factor out scalars . When you have a scalar times an operator, the inverse is the product of the two inverses, with the inverse of the scalar being one over the scalar.

In[47]:=
```
Inv[Times[a$sym_?ScalarQ, b$sym_?OperatorQ]] := Times[Inv[a$sym], Inv[b$sym]]
Inv[Times[a$sym_?OperatorQ, b$sym_?ScalarQ]] := Times[Inv[b$sym], Inv[a$sym]]
```

In[49]:=
```
{Inv[ω Ix], Inv[Ix ω]}
```
Out[49]= $\left\{\frac{\mathrm{Inv}[\mathrm{Ix}]}{\omega}, \frac{\mathrm{Inv}[\mathrm{Ix}]}{\omega}\right\}$

Now this comes out right:

In[50]:=
```
Mult[ Inv[ω Ix], Ix]
```
Out[50]= $\frac{1}{\omega}$

Distribute the inverse over addition.

In[51]:=
```
Inv[a$sym___, b$sym_Plus, c$sym___] := Plus @@ (Inv[a$sym, #, c$sym] &) /@ List @@ b$sym
```

In[52]:=
```
Inv[ω Ix + Δ Iy]
```
Out[52]= $\frac{\mathrm{Inv}[\mathrm{Ix}]}{\omega} + \frac{\mathrm{Inv}[\mathrm{Iy}]}{\Delta}$

In[53]:=
```
Inv[ω Ix + Δ Mult[Iy, aL]]
```
Out[53]= $\frac{\mathrm{Inv}[\mathrm{Ix}]}{\omega} + \frac{\mathrm{Inv}[\mathrm{Mult}[\mathrm{Iy}, \mathrm{aL}]]}{\Delta}$

The inverse applied to a Mult[] of operators should give a list of Inv[] operators multiplied together, with the list order reversed.

In[54]:=
```
Inv[a$sym___, b$sym_Mult, c$sym___] :=
 Mult @@ (Inv[a$sym, #, c$sym] &) /@ Reverse[List @@ b$sym]
```

In[55]:=
```
Inv[Mult[Iy, aL]]
```
Out[55]=
```
Mult[Inv[aL], Inv[Iy]]
```

In[56]:=
```
Inv[Mult[ω Iy, Δ aL]]
```
Out[56]= $\frac{\mathrm{Mult}[\mathrm{Inv}[\mathrm{aL}], \mathrm{Inv}[\mathrm{Iy}]]}{\Delta\ \omega}$

In[57]:=
```
Mult[Inv[Mult[aL, Iy]], aL, Iy]
```
Out[57]= 1

We would really like to simplify the product of these two operators.

In[58]:=
```
{Inv[op1], op2}
```

Out[58]= $\left\{\frac{\text{Inv}[\text{Iz}]}{\triangle}, \omega\,(\text{Mult}[\text{aL}, \text{Iz}] - \text{Mult}[\text{Iz}, \text{aR}])\right\}$

Here is the product .

In[59]:=
```
Mult[Inv[op1], op2]
```

Out[59]= $\frac{\omega\,(-\text{aR} + \text{Mult}[\text{Inv}[\text{Iz}], \text{aL}, \text{Iz}])}{\triangle}$

The second term is now simplified, but the first term is not. Does sorting help?

In[60]:=
```
Mult[Inv[op1], op2 /. Mult → SortedMult]
```

Out[60]= $\frac{\omega\,(-\text{aR} + \text{Mult}[\text{Inv}[\text{Iz}], \text{aL}, \text{Iz}])}{\triangle}$

It does!

This works

In[61]:=
```
Mult[Inv[Mult[aL, Iz]], Mult[aL, Iz]]
```

Out[61]= 1

But this does not .

In[62]:=
```
Mult[Inv[Mult[Iz, aL]], Mult[aL, Iz]]
```

Out[62]= Mult[Inv[aL], Inv[Iz], aL, Iz]

Does sorting help?

In[63]:=
```
Mult[
 Inv[Mult[Iz, aL]] /. Mult → SortedMult,
 Mult[aL, Iz] /. Mult → SortedMult]
```

Out[63]= Mult[Inv[aL], Inv[Iz], aL, Iz]

Again, it does!

Try some more cases.

In[64]:=
```
Mult[Inv[I$p], Mult[I$p, aL, aL]]
```

Out[64]= Mult[aL, aL]

In[65]:=
```
Mult[aL, Inv[I$p], Mult[I$p, aL, aL, aL]]
```

Out[65]= Mult[aL, aL, aL, aL]

In[66]:=
```
Mult[aL, Inv[I$p], Mult[I$p, aL], aL, aL]
```

Out[66]= Mult[aL, aL, aL, aL]

Test-drive the inverse

Two-spin problem

Clear angular momentum operators for two spins .

In[67]:=
```
Clear[J, Ix, Iy, Iz];
CreateScalar[{J}];
$Assumptions = {Element[J, Reals]};
CreateOperator[{{Ix, Iy, Iz}, {Sx, Sy, Sz}}];
SpinSingle$CreateOperators[Ix, Iy, Iz];
SpinSingle$CreateOperators[Sx, Sy, Sz];
```

SpinSingle$CreateOperators : Spin operators already exist.

SpinSingle$CreateOperators : Adding spin commutations relations.

SpinSingle$CreateOperators : No angular momentum L defined.

SpinSingle$CreateOperators : Spin operators already exist.

SpinSingle$CreateOperators : Adding spin commutations relations.

SpinSingle$CreateOperators : No angular momentum L defined.

In[73]:=
```
terms = Evolver[J Mult[Iz, Sz], t, Ix, quiet → False]
```

$$\rho \text{ matrix } = \begin{pmatrix} \text{Ix} \\ \text{J Mult[Iy, Sz]} \\ -\text{J}^2 \text{ Mult[Ix, Sz, Sz]} \\ -\text{J}^3 \text{ Mult[Iy, Sz, Sz, Sz]} \\ \text{J}^4 \text{ Mult[Ix, Sz, Sz, Sz, Sz]} \end{pmatrix}$$

Evolver : Unrecognized evolution

Out[73]= $\{\text{Ix}, \text{J Mult[Iy, Sz]}, -\text{J}^2 \text{ Mult[Ix, Sz, Sz]}, -\text{J}^3 \text{ Mult[Iy, Sz, Sz, Sz]}, \text{J}^4 \text{ Mult[Ix, Sz, Sz, Sz, Sz]}\}$

We can solve the system using a 3 x 3 set of equations . Try working backwards to find coefficients.

In[74]:=
```
Mult[Inv[terms[[2]]], terms[[3]]]
```

Out[74]= $-\text{J Mult[Inv[Sz], Inv[Iy], Ix, Sz, Sz]}$

This first coefficient is a mess . Look at the next possible coefficient.

In[75]:=
```
Mult[Inv[terms[[1]]], terms[[3]]]
```

Out[75]= $-\text{J}^2 \text{ Mult[Sz, Sz]}$

This one is much simpler, and contains no I-spin operators. Write down the formal solution to this problem:

In[76]:=
```
solution1 = MatrixExp[{{0, -J^2 Mult[Sz, Sz], 0}, {1, 0, 0}, {0, 1, 0}}] .
    {-J^2 Mult[Ix, Sz, Sz], J Mult[Iy, Sz], Ix} // ExpToTrig // Simplify;
```

In[77]:=
```
solution1[[1]] // Simplify // Expand
```

Out[77]= $-J^2 \text{Cos}\left[J \sqrt{\text{Mult[Sz, Sz]}}\right] \text{Mult[Ix, Sz, Sz]} - J^2 \text{Mult[Iy, Sz]} \sqrt{\text{Mult[Sz, Sz]}} \text{Sin}\left[J \sqrt{\text{Mult[Sz, Sz]}}\right]$

We can also solve the system using a 4 x 4 set of equations . Again, try working backwards to find coefficients. Start with

In[78]:=
```
Mult[Inv[terms[[3]]], terms[[4]]]
```

Out[78]= J Mult[Inv[Sz], Inv[Sz], Inv[Ix], Iy, Sz, Sz, Sz]

which is a mess . The next coefficient is again nice.

In[79]:=
```
Mult[Inv[terms[[2]]], terms[[4]]]
```

Out[79]= $-J^2 \text{Mult[Sz, Sz]}$

Contains no I-spin operators, nice. Write down the formal solution to this problem:

In[80]:=
```
solution2 =
  MatrixExp[{{0, -J^2 Mult[Sz, Sz], 0, 0}, {1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}}] .
    {-J^3 Mult[Iy, Sz, Sz, Sz], -J^2 Mult[Ix, Sz, Sz],
     J Mult[Iy, Sz], Ix} // ExpToTrig // Simplify;
```

In[81]:=
```
solution1[[2]] // Simplify // Expand
```

Out[81]= $J \text{Cos}\left[J \sqrt{\text{Mult[Sz, Sz]}}\right] \text{Mult[Iy, Sz]} - \frac{J \text{Mult[Ix, Sz, Sz]} \text{Sin}\left[J \sqrt{\text{Mult[Sz, Sz]}}\right]}{\sqrt{\text{Mult[Sz, Sz]}}}$

There is something wrong with my MatrixExp solution! For one, the units don’t work.

The first term, when traced against Ix, gives the expected splitting and intensity pattern I think. To see this we have to expand the Cos[] function carefully. We can do this using Sylverster’s theorem. Compare this answer to my quantum-mechanics notes.

Lesson: Having an operator in a Cosine[] or Sin[] might be ok.

Follow up: Why isn’t MatrixExp not giving the correct solution?

Spin - boson problem

Create a set of spin and harmonic-oscillator variables.

In[82]:=
```
Clear[λ, t, Ix, Iy, Iz, I$m, I$p, aL, aR, H];
CreateScalar[{λ, t, g}];
$Assumptions = {Element[λ, Reals], Element[t, Reals], Element[g, Reals]};
CreateOperator[{{Ix, Iy, Iz, I$m, I$p}, {aL, aR}}];
SpinSingle$CreateOperators[Ix, Iy, Iz, 1 / 2];
OscSingle$CreateOperators[aL, aR];
```

SpinSingle$CreateOperators: Spin operators already exist.

SpinSingle$CreateOperators: Adding spin commutations relations.

SpinSingle$CreateOperators: Angular momentum L = 1/2. Adding operator simplification rules.

OscSingle$CreateOperators: Oscillator operators already exist.

OscSingle$CreateOperators: Adding oscillator commutations relations.

Add raising and lowering operator commutation rules

In[88]:=
```
I$p /: Comm[I$p, I$m] := 2 Iz;
I$p /: Comm[I$p, Iz] := -I$p;
I$m /: Comm[I$m, I$p] := -2 Iz;
I$m /: Comm[I$m, Iz] := I$m;
Iz /: Comm[Iz, I$p] := I$p;
Iz /: Comm[Iz, I$m] := -I$m;
```

and simplification rules

In[94]:=
```
I$p /: NonCommutativeMultiply[a___, I$p, I$p, b___] := 0;
I$m /: Mult[a___, I$m, I$m, b___] := 0;
```

I$p /: Mult[a___, I$p, Iz, b___] := $-\frac{1}{2}$ Mult[a, I$p, b];

I$p /: Mult[a___, I$p, I$m, b___] := $\frac{1}{2}$ Mult[a, b] + Mult[a, Iz, b];

I$m /: Mult[a___, I$m, Iz, b___] := $\frac{1}{2}$ Mult[a, I$m, b];

I$m /: Mult[a___, I$m, I$p, b___] := $\frac{1}{2}$ Mult[a, b] - Mult[a, Iz, b];

I$z /: Mult[a___, I$z, Ip, b___] := $\frac{1}{2}$ Mult[a, I$p, b];

I$z /: Mult[a___, I$z, I$m, b___] := $-\frac{1}{2}$ Mult[a, I$m, b];

Try to achieve normal ordering for harmonic oscillator

In[102]:=
```
aR /: Mult[a___, aL, aR, b___] := Mult[a, aR, aL, b] + Mult[a, b]
```

Try it out in a couple cases

In[103]:=
```
Mult[aL, aR] // Expand
```
Out[103]= $1 + \text{Mult}[aR, aL]$

In[104]:=
```
1/2 (Mult[aL, aR] + Mult[aR, aL])  // Expand
```
Out[104]= $\frac{1}{2} + \text{Mult}[aR, aL]$

In[105]:=
```
Mult[aL, aR, aR]
```
Out[105]= $2\, aR + \text{Mult}[aR, aR, aL]$

In[106]:=
```
Mult[aL, aR, aL, aR]
```
Out[106]= $1 + 3\, \text{Mult}[aR, aL] + \text{Mult}[aR, aR, aL, aL]$

The normal ordering rule above seems to work .

Here is a unitary rotation that Evolver[] cannot resolve.

In[107]:=
```
terms = Evolver[λ Mult[Iz, aL], t, I$p, quiet → False]
```

$$\rho\ \text{matrix} = \begin{pmatrix} I\$p \\ -i\,\lambda\, \text{Mult}[I\$p, aL] \\ -\lambda^2\, \text{Mult}[I\$p, aL, aL] \\ i\,\lambda^3\, \text{Mult}[I\$p, aL, aL, aL] \\ \lambda^4\, \text{Mult}[I\$p, aL, aL, aL, aL] \end{pmatrix}$$

Evolver : Unrecognized evolution

Out[107]= $\{I\$p, -i\,\lambda\, \text{Mult}[I\$p, aL], -\lambda^2\, \text{Mult}[I\$p, aL, aL], i\,\lambda^3\, \text{Mult}[I\$p, aL, aL, aL], \lambda^4\, \text{Mult}[I\$p, aL, aL, aL, aL]\}$

What is the proportionality constant? Start “dividing” terms[[4]] by lower terms

In[108]:=
```
Mult[
 Inv[terms[[3]] /. Mult → SortedMult],
 terms[[4]] /. Mult → SortedMult]
```
Out[108]= $-i\, aL\, \lambda$

What if we just look at the lowest two terms?

In[109]:=
```
Mult[
 Inv[terms[[1]] /. Mult → SortedMult],
 terms[[2]] /. Mult → SortedMult]
```
Out[109]= $-i\, aL\, \lambda$

This last division gives the same answer, but now we just have to solve **one** differential equation.

Here is another unitary rotation that Evolve[] cannot resolve .

```
In[110]:= terms = Evolver[2 I g (Mult[Iz, aR] - Mult[Iz, aL]), t, I$p] // Simplify

Evolver: Unrecognized evolution

Out[110]= {I$p, -2 g (Mult[I$p, aL] - Mult[I$p, aR]),
  -4 g^2 (I$p - Mult[I$p, aL, aL] + 2 Mult[I$p, aR, aL] - Mult[I$p, aR, aR]),
  8 g^3 (3 Mult[I$p, aL] - 3 Mult[I$p, aR] - Mult[I$p, aL, aL, aL] +
     3 Mult[I$p, aR, aL, aL] - 3 Mult[I$p, aR, aR, aL] + Mult[I$p, aR, aR, aR]),
  16 g^4 (3 I$p - 6 Mult[I$p, aL, aL] + 12 Mult[I$p, aR, aL] - 6 Mult[I$p, aR, aR] +
     Mult[I$p, aL, aL, aL, aL] - 4 Mult[I$p, aR, aL, aL, aL] + 6 Mult[I$p, aR, aR, aL, aL] -
     4 Mult[I$p, aR, aR, aR, aL] + Mult[I$p, aR, aR, aR, aR])}
```

What is the proportionality constant? Print out the four terms. They get increasingly complicated.

```
In[111]:= terms[[1]]

Out[111]= I$p

In[112]:= terms[[2]]

Out[112]= -2 g (Mult[I$p, aL] - Mult[I$p, aR])

In[113]:= terms[[3]]

Out[113]= -4 g^2 (I$p - Mult[I$p, aL, aL] + 2 Mult[I$p, aR, aL] - Mult[I$p, aR, aR])

In[114]:= terms[[4]]

Out[114]= 8 g^3 (3 Mult[I$p, aL] - 3 Mult[I$p, aR] - Mult[I$p, aL, aL, aL] +
    3 Mult[I$p, aR, aL, aL] - 3 Mult[I$p, aR, aR, aL] + Mult[I$p, aR, aR, aR])
```

Look at "divisions" from terms 4 down to 1

```
In[115]:= Mult[Inv[terms[[3]] /. Mult → SortedMult],
       terms[[4]] /. Mult → SortedMult] // Simplify

Out[115]= -g (17 aL - 8 aR - 9 Inv[aL] + 6 Inv[aR] - 2 Mult[aL, aL, aL] +
    6 Mult[aR, aL, aL] - 6 Mult[aR, aR, aL] + 2 Mult[aR, aR, aR] -
    3 Mult[Inv[aL], aR, aL] + Mult[Inv[aL], aR, aR] + 6 Mult[Inv[aL], Inv[aL], aR] +
    3 Mult[Inv[aL], Inv[aR], aL] - 6 Mult[Inv[aR], aL, aL] -
    6 Mult[Inv[aR], Inv[aR], aL] - 6 Mult[Inv[aL], Inv[aL], aR, aL, aL] +
    6 Mult[Inv[aL], Inv[aL], aR, aR, aL] - 2 Mult[Inv[aL], Inv[aL], aR, aR, aR] -
    Mult[Inv[aL], Inv[aR], aL, aL, aL] + 2 Mult[Inv[aR], Inv[aR], aL, aL, aL])

In[116]:= Mult[Inv[terms[[2]] /. Mult → SortedMult],
      terms[[4]] /. Mult → SortedMult]

Out[116]= -1/(2 g) (8 g^3 (3 - Mult[aL, aL] - 3 Mult[Inv[aL], aR] + 3 Mult[Inv[aL], aR, aL, aL] -
       3 Mult[Inv[aL], aR, aR, aL] + Mult[Inv[aL], aR, aR, aR]) -
    8 g^3 (-3 + 3 Mult[aL, aL] - 3 Mult[aR, aL] + Mult[aR, aR] +
       3 Mult[Inv[aR], aL] - Mult[Inv[aR], aL, aL, aL]))
```

In[117]:=
```
Mult[Inv[terms[[1]] /. Mult → SortedMult],
     terms[[4]] /. Mult → SortedMult]
```

Out[117]=
```
8 g^3
 (3 aL - 3 aR - Mult[aL, aL, aL] + 3 Mult[aR, aL, aL] - 3 Mult[aR, aR, aL] + Mult[aR, aR, aR])
```

There are a mess. Look as "divisions" from terms 3 down to 1

Out[ ]=
```
-8 g^3 (Mult[I$p, aL, aL, aL] - Mult[I$p, aL, aL, aR] -
   Mult[I$p, aL, aR, aL] + Mult[I$p, aL, aR, aR] - Mult[I$p, aR, aL, aL] +
   Mult[I$p, aR, aL, aR] + Mult[I$p, aR, aR, aL] - Mult[I$p, aR, aR, aR])
```

In[118]:=
```
Mult[Inv[terms[[2]] /. Mult → SortedMult],
      terms[[3]] /. Mult → SortedMult] // Simplify
```

Out[118]=
```
-2 g (3 aL - aR - Inv[aL] + Inv[aR] -
   2 Mult[Inv[aL], aR, aL] + Mult[Inv[aL], aR, aR] - Mult[Inv[aR], aL, aL])
```

In[119]:=
```
Mult[Inv[terms[[1]] /. Mult → SortedMult],
     terms[[3]] /. Mult → SortedMult]
```

Out[119]=
```
-4 g^2 (1 - Mult[aL, aL] + 2 Mult[aR, aL] - Mult[aR, aR])
```

Also a mess. Look as "divisions" from terms 2 down to 1.

In[120]:=
```
Mult[Inv[terms[[1]] /. Mult → SortedMult],
     terms[[2]] /. Mult → SortedMult]
```

Out[120]=
```
-2 (aL - aR) g
```

This is tractable! So maybe start by matching the lowest terms and work upwards instead?

Conclusions

I like my 4 x 4 matrix, one-side-fits-all, solution.

For two coupled spins, you might be able to apply the 4 x 4 matrix solution. I need to write out the coupled differential equations.

For spin-boson problems you often just have **one** coupled differential equation. In this case the 4 x 4 matrix method unnecessarily complicated. And I cannot resolve the “division” in the 4 x 4 matrix for the simplest spin-boson problem. Arrgh.

So perhaps it would instead be better to work upwards from 1 x 1, to 2 x 2, upwards to 4 x 4.