UniDyn--Study-03.nb

John A. Marohn
jam99@cornell.edu
Cornell University

Abstract: This demonstration notebook loads the **UniDyn** package and the supplemental functions in the /studies directory. Verify that the new **Evolver** function passes all its unit tests. Show examples of **Evolver** involving one spin, two spins, the harmonic oscillator, quantum optics, and electron transfer.

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 and the /studies directory.

EDIT THE FOLLOWING PATH STRINGS:

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

In[3]:=
```
$UniDynStudyPath =
  "/Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/
    studies";
```

YOU SHOULD NOT NEED TO EDIT ANYTHING FROM HERE ONWARDS.

Load and test the packages

An execution-time test

In[4]:=
```
testevolver := Module[{Ix, Iy, Iz, ω, Δ, time},
    SpinSingle$CreateOperators[Ix, Iy, Iz];
    CreateScalar[ω, Δ];
    $Assumptions = {Element[Δ, Reals], Element[ω, Reals], ω ≥ 0};
    time = 1000 * Timing[Evolver[Δ Iz + ω Ix, t, Iz]][[1]];
    Print["run time = ", time, " ms"];
    Return[time]
   ];
```

Unidyn and Evolver version 1

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[5]:=
```
$Path = AppendTo[$Path, $UniDynPath];
FindFile["UniDyn`"]
```

Out[6]=
```
/Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/unidyn/UniDyn.m
```

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

In[7]:=
```
$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, $\rho(0)$] calculates $\rho(t)$ = Exp[−I H t] $\rho(0)$ Exp[+I H t], assuming that H is time independent, according to the commutation rules followed by $\rho(0)$ and H.

Carry out the unit tests for the **Unidyn** functions.

In[9]:=
```
SetDirectory[$UniDynPath];
```

In[10]:=
```
fn = FileNames["*-tests.m"];
test$report = TestReport /@ fn;
TableForm[Table[test$report [[k]], {k, 1, Length[test$report]}]]
```

Out[12]//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]

How long to compute an off-resonance Rabi nutation?

In[13]:=
```
t1 = testevolver;
```

SpinSingle$CreateOperators: Creating spin operators.

SpinSingle$CreateOperators: Adding spin commutations relations.

SpinSingle$CreateOperators: No angular momentum L defined.

run time = 562.663 ms

Extras and Evolver version 2

Now load the functions and packages in the **/studies directory**.

In[14]:=
```
$Path = AppendTo[$Path, $UniDynStudyPath];
FindFile["Inv`"]
FindFile["SpinBoson`"]
FindFile["Evolve2`"]
```

Out[15]= /Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/studies/Inv.m

Out[16]= /Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/studies/SpinBoson.m

Out[17]= /Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/studies/Evolve2.m

In[18]:=
```
Needs["Inv`"]
Needs["SpinBoson`"]
Needs["Evolve2`"]
```

Inv: Inv[a] returns the inverse of the expression.

SpinBoson$CreateOperators: SpinBoson$CreateOperators[Ix,Iy,Iz,Ip,Im,aR,aL] creates Ix, Iy, Iz spin one half angular−momentum operators; the associated spin raising and lowering operators Ip, Im; and harmonic−oscllator raising and lowering operators aR, aL.

Evolve: Symbol Evolve appears in multiple contexts {Evolve2`, Evolve`}; definitions in context Evolve2` may shadow or be shadowed by other definitions.

AllCommutingQ: Symbol AllCommutingQ appears in multiple contexts {Evolve2`, Evolve`}; definitions in context Evolve2` may shadow or be shadowed by other definitions.

VisualComplexity: Symbol VisualComplexity appears in multiple contexts {Evolve2`, Evolve`}; definitions in context Evolve2` may shadow or be shadowed by other definitions.

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, $\rho(0)$] calculates $\rho(t) = \text{Exp}[-\text{I H t}]\ \rho(0)\ \text{Exp}[+\text{I H t}]$, assuming that H is time independent, according to the commutation rules followed by $\rho(0)$ and H.

Extend the directory to the **/studies directory** and run all the tests found there.

In[21]:=
```
SetDirectory[$UniDynStudyPath];
```

In[22]:=
```
fn = FileNames["*-tests.m"];
test$report = TestReport /@ fn;
TableForm[Table[test$report [[k]], {k, 1, Length[test$report]}]]
```

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.

Out[24]//TableForm=

TestReportObject[Title: Test Report: Evolve2−tests.m
Success rate: 100% Tests run: 22]

TestReportObject[Title: Test Report: Inv−tests.m
Success rate: 100% Tests run: 22]

TestReportObject[Title: Test Report: SpinBoson−tests.m
Success rate: 100% Tests run: 2]

How long to compute an off-resonance Rabi nutation?

In[25]:=
```
t2 = testevolver;
```

SpinSingle$CreateOperators: Creating spin operators.

SpinSingle$CreateOperators: Adding spin commutations relations.

SpinSingle$CreateOperators: No angular momentum L defined.

```
run time = 15.564 ms
```

Execution-time comparison

In[26]:=
```
Print["Evolver version 2 is ",
 t1 / t2, "x faster than Evolver version 1."]
```

```
Evolver version 2 is 36.1516x faster than Evolver version 1.
```

One spin

Define a single spin *I*, leaving the spin angular momentum unspecified.

In[27]:=
```
Clear[Ix, Iy, Iz, ω, Δ, I$p$sym, I$m$sym, I$p, I$m]
SpinSingle$CreateOperators[Ix, Iy, Iz];
CreateScalar[ω, Δ];
$Assumptions = {Element[Δ, Reals], Element[ω, Reals], ω ≥ 0};
```

SpinSingle$CreateOperators: Creating spin operators.

SpinSingle$CreateOperators: Adding spin commutations relations.

SpinSingle$CreateOperators: No angular momentum L defined.

Free evolution off resonance.

In[31]:=
```
Evolver[Δ Iz, t, Ix]
```

Out[31]=
```
Ix Cos[t Δ] + Iy Sin[t Δ]
```

Nutation on resonance.

In[32]:=
```
Evolver[ω Ix, t, Iz]
```

Out[32]=
```
Iz Cos[t ω] - Iy Sin[t ω]
```

Nutation off resonance.

In[33]:=
```
Collect[Evolver[Δ Iz + ω Ix, t, Iz], {Ix, Iy, Iz}]
```

Out[33]=
$$\text{Ix}\left(\frac{\Delta\,\omega}{\Delta^2+\omega^2}-\frac{\Delta\,\omega\,\text{Cos}\left[\text{t}\,\sqrt{\Delta^2+\omega^2}\,\right]}{\Delta^2+\omega^2}\right)+\text{Iz}\left(1-\frac{\omega^2}{\Delta^2+\omega^2}+\frac{\omega^2\,\text{Cos}\left[\text{t}\,\sqrt{\Delta^2+\omega^2}\,\right]}{\Delta^2+\omega^2}\right)-\frac{\text{Iy}\,\omega\,\text{Sin}\left[\text{t}\,\sqrt{\Delta^2+\omega^2}\,\right]}{\sqrt{\Delta^2+\omega^2}}$$

Check that the on-resonance nutation result is recovered at zero resonance offset.

In[34]:=
```
Collect[Evolver[Δ Iz + ω Ix, t, Iz], {Ix, Iy, Iz}] /. {Δ → 0} // PowerExpand
```

Out[34]=
```
Iz Cos[t ω] - Iy Sin[t ω]
```

Define spin raising and lowering operators, I_+ and I_-.

In[35]:=
```
I$p$sym = Ix + I Iy;
I$m$sym = Ix - I Iy;
```

Define a rule for rewriting I_x and I_y in terms of the raising and lowering operators.

In[37]:=
```
CreateOperator[{{I$p, I$m}}]
IpIm$rules = {Ix → 1/2 (I$p + I$m), Iy → 1/(2 I) (I$p - I$m)};
```

Check the spin raising and lowering operator commutation relations.

In[39]:=
```
{Comm[I$p$sym, Iz], Comm[I$m$sym, Iz], Comm[I$p$sym, I$m]} /. IpIm$rules //
 Simplify
```

Out[39]=
```
{-I$p, I$m, Comm[I$p, I$m]}
```

Evolve the raising operator I_+ under off-resonance free evolution.
This does not simplify nicely.

In[40]:=
```
Evolver[Δ Iz, t, I$p$sym] // FullSimplify
```

Out[40]=
```
Mult[ℯ^(t Δ (-i Mult[Ix,Inv[Ix+i Iy]]+Mult[Iy,Inv[Ix+i Iy]])), Ix] +
 i Mult[ℯ^(t Δ (-i Mult[Ix,Inv[Ix+i Iy]]+Mult[Iy,Inv[Ix+i Iy]])), Iy]
```

If we first use Evolve to expand the sums in I_+ and I_-, then Evolver knows what to do.

In[41]:=
```
({Evolve[Δ Iz, t, I$p$sym], Evolve[Δ Iz, t, I$m$sym]} /. Evolve → Evolver //
    TrigToExp) /. IpIm$rules // Simplify
```

Out[41]=
```
{ℯ^(-i t Δ) I$p, ℯ^(i t Δ) I$m}
```

In[42]:=
```
Clear[Ix, Iy, Iz, ω, Δ, I$p$sym, I$m$sym, I$p, I$m]
```

Two spins

Define two spins, *I* and *S*.
The *S* spin has total angular momentum 1/2 while the angular momentum of the *I* spin remains unspecified.

In[43]:=
```
Clear[Ix, Iy, Iz, Sx, Sy, Sz, J]
CreateOperator[{{Ix, Iy, Iz}, {Sx, Sy, Sz}}];
SpinSingle$CreateOperators[Ix, Iy, Iz];
SpinSingle$CreateOperators[Sx, Sy, Sz, 1 / 2];
CreateScalar[J];
```

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

Evolve I_x under a weak scalar coupling between *I* and *S* spins.
This evolution step is the key idea enabling two-dimensional nuclear magnetic resonance experiments and quantum computing.

In[48]:=
```
Evolver[J Mult[Iz, Sz], t, Ix]
```

Out[48]=
$$\text{Ix}\,\text{Cos}\left[\frac{\text{J t}}{2}\right] + 2\,\text{Mult}[\text{Iy}, \text{Sz}]\,\text{Sin}\left[\frac{\text{J t}}{2}\right]$$

Evolving S_x does not give an analogously nice answer because the angular momentum of the *I* spin is unspecified.

In[49]:=
```
Evolver[J Mult[Iz, Sz], t, Sx]
```

Out[49]=
$$\text{Mult}\left[\text{Cos}\left[\text{J t}\,\sqrt{\text{Mult}[\text{Iz}, \text{Iz}]}\,\right], \text{Sx}\right] + \text{Mult}\left[\text{Sin}\left[\text{J t}\,\sqrt{\text{Mult}[\text{Iz}, \text{Iz}]}\,\right], \frac{\text{Mult}[\text{Iz}, \text{Sy}]}{\sqrt{\text{Mult}[\text{Iz}, \text{Iz}]}}\right]$$

In[50]:=
```
Clear[Ix, Iy, Iz, Sx, Sy, Sz, J]
```

Harmonic oscillator

In[51]:=
```
Clear[aL, aR, ω, Q$sym, P$sym, Q, P, ω, δq, δp, A, ϕ, Δ, H$0, H$1, H$1$ϕ, H$2]
OscSingle$CreateOperators[aL, aR];
CreateScalar[ω, δq, δp, A, ϕ, Δ];
```

OscSingle$CreateOperators: Creating oscillator operators.

OscSingle$CreateOperators: Adding oscillator commutations relations.

Define the number operator.

In[54]:=
```
Nop = Mult[aR, aL];
```

Define position and momentum operators.

In[55]:=
```
Q$sym = (aR + aL) / Sqrt[2];
P$sym = I (aR - aL) / Sqrt[2];
```

Define a rule for rewriting raising and lowering operators in terms of position and momentum operators.

In[57]:=
```
CreateOperator[{{Q, P}}]
```

```
QP$rules = {aR → (Q - I P) / Sqrt[2], aL → (Q + I P) / Sqrt[2]};
```

Define the harmonic oscillator Hamiltonian.

In[59]:=
```
H$0 = ω (Nop + 1 / 2);
```

Evolve the position and momentum operators under the harmonic oscillator Hamiltonian.

In[60]:=
```
{Evolver[H$0, t, Q$sym], Evolver[H$0, t, P$sym]} /. QP$rules // Simplify
```

Out[60]=
$$\{Q \cos[t\,\omega] - P \sin[t\,\omega],\ P \cos[t\,\omega] + Q \sin[t\,\omega]\}$$

A position kick.

In[61]:=
```
{Evolver[-δq P$sym, 1, Q$sym], Evolver[-δq P$sym, 1, P$sym]} /. QP$rules // Simplify
```

Out[61]=
$$\{Q + \delta q,\ P\}$$

A momentum kick.

In[62]:=
```
{Evolver[δp Q$sym, 1, Q$sym], Evolver[δp Q$sym, 1, P$sym]} /. QP$rules // Simplify
```

Out[62]=
$$\{Q,\ P + \delta p\}$$

Evolution under a force .

In[63]:=
```
H$1 = - A Q$sym;
```

In[64]:=
```
{Evolver[H$1, t, Q$sym], Evolver[H$1, t, P$sym]} /. QP$rules // Simplify
```

Out[64]=
$$\{Q,\ P - A\,t\}$$

Evolution under a force with a phase factor .

In[65]:=
```
H$1$ϕ = A 1/√2 (Exp[-I ϕ] aR + Exp[I ϕ] aL );
{Evolver[H$1$ϕ, t, Q$sym], Evolver[H$1$ϕ, t, P$sym]} /. QP$rules // Simplify
```

Out[66]=
$$\{Q + A\,t \sin[\phi],\ P + A\,t \cos[\phi]\}$$

Evolution under a squeezing Hamiltonian.

In[67]:=
```
H$2 = -Δ/2 I (Exp[-I ϕ] Mult[aR, aR] - Exp[I ϕ] Mult[aL, aL]);
```

In[68]:=
```
ans = {Evolver[H$2, t, Q$sym], Evolver[H$2, t, P$sym]} /. QP$rules // Simplify //
   ExpToTrig // Simplify
```

Out[68]=
$$\{Q \cosh[t\,\triangle] + (Q \cos[\phi] - P \sin[\phi]) \sinh[t\,\triangle],\ P \cosh[t\,\triangle] - (P \cos[\phi] + Q \sin[\phi]) \sinh[t\,\triangle]\}$$

We can see that the squeezing is phase-dependent .

In[69]:=
```
ans /. ϕ → 0 // TrigToExp
```

Out[69]=
$$\left\{e^{t\,\Delta}\,Q,\ e^{-t\,\Delta}\,P\right\}$$

In[70]:=
```
ans /. ϕ → π // TrigToExp
```

Out[70]=
$$\left\{e^{-t\,\Delta}\,Q,\ e^{t\,\Delta}\,P\right\}$$

Evolution under the harmonic oscillator and squeezing Hamiltonians.

In[71]:=
```
ans =
 {Evolver[H$0 + H$2, t, Q$sym], Evolver[H$2, t, P$sym]} /. QP$rules // Simplify //
    ExpToTrig // Simplify // Collect[#, {Q, P}] &
```

Out[71]=
$$\left\{-\frac{P\,(\omega+\Delta\,\text{Sin}[\phi])\,\text{Sinh}\left[t\,\sqrt{\Delta^2-\omega^2}\right]}{\sqrt{\Delta^2-\omega^2}}+Q\left(\text{Cosh}\left[t\,\sqrt{\Delta^2-\omega^2}\right]+\frac{\Delta\,\text{Cos}[\phi]\,\text{Sinh}\left[t\,\sqrt{\Delta^2-\omega^2}\right]}{\sqrt{\Delta^2-\omega^2}}\right),\right.$$
$$\left.-Q\,\text{Sin}[\phi]\,\text{Sinh}[t\,\Delta]+P\,(\text{Cosh}[t\,\Delta]-\text{Cos}[\phi]\,\text{Sinh}[t\,\Delta])\right\}$$

The position and momentum are somewhat simpler when $\phi = 0$ and $\phi = \pi/2$.

In[72]:=
```
ans /. ϕ → 0 // Simplify // Collect[#, {Q, P}] &
```

Out[72]=
$$\left\{-\frac{P\,\omega\,\text{Sinh}\left[t\,\sqrt{\Delta^2-\omega^2}\right]}{\sqrt{\Delta^2-\omega^2}}+Q\left(\text{Cosh}\left[t\,\sqrt{\Delta^2-\omega^2}\right]+\frac{\Delta\,\text{Sinh}\left[t\,\sqrt{\Delta^2-\omega^2}\right]}{\sqrt{\Delta^2-\omega^2}}\right),\right.$$
$$\left.P\,(\text{Cosh}[t\,\Delta]-\text{Sinh}[t\,\Delta])\right\}$$

In[73]:=
```
ans /. ϕ → π / 2 // Simplify // Collect[#, {Q, P}] &
```

Out[73]=
$$\left\{Q\,\text{Cosh}\left[t\,\sqrt{\Delta^2-\omega^2}\right]+\frac{P\,(-\Delta-\omega)\,\text{Sinh}\left[t\,\sqrt{\Delta^2-\omega^2}\right]}{\sqrt{\Delta^2-\omega^2}},\ P\,\text{Cosh}[t\,\Delta]-Q\,\text{Sinh}[t\,\Delta]\right\}$$

In[74]:=
```
Clear[aL, aR, ω, Q$sym, P$sym, Q, P, ω, δq, δp, A, ϕ, Δ, H$0, H$1, H$1$ϕ, H$2]
```

Quantum optics

In[75]:=
```
Clear[Ix, Iy, Iz, I$p, I$m, aR, aL, ω0, ω, λ, Δ]

SpinBoson$CreateOperators[Ix, Iy, Iz, I$p, I$m, aR, aL];
```

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.

SpinBoson$CreateOperators: Creating operators.

SpinBoson$CreateOperators: Adding Ip and Im commutations relations.

SpinBoson$CreateOperators: Adding Ip and Im simplification rules.

SpinBoson$CreateOperators: Adding aL and aR normal ordering rule.

In[77]:=
```
CreateScalar[{ω$0, ω, λ}];
$Assumptions = {ω$0 ∈ Reals, ω0 ∈ Reals, λ ∈ Complexes};
```

Define the Hamiltonian.
Here λ is a real coupling constant; we need the factor of I in the last term so that the Hamiltonian is Hermitian.

In[79]:=
```
H$0 = ω$0 Iz + ω Mult[aR, aL] + I λ Mult[I$p + I$m, aL - aR];
```

Two Hamiltonians we will use to create an interaction representation.

In[80]:=
```
H1$T = 1/2 (ω$0 + ω) Iz;
H2$T = 1/2 (ω$0 + ω) Mult[aR, aL];
```

Now implement a two-fold interaction representation.
Here $\Delta = \omega\$0 - \omega$ is the frequency difference between the two-level system and the radiation.

In[82]:=
```
temp = Evolve[H1$T, t, H$0 ] /. Evolve → Evolver;
H$0$T = (Evolve[H2$T, t, temp] /. Evolve → Evolver ) - H1$T - H2$T ;
H$0$T = H$0$T // Collect[#, {Iz, Mult[aR, aL], Mult[I$m, aR],
      Mult[I$p, aL], Mult[I$p, aR], Mult[I$m, aL]}, Simplify] & ;
H$0$T = H$0$T /. {ω$0 → ω + Δ}
```

Out[85]=
$$\frac{Iz\,\Delta}{2} - \frac{1}{2}\Delta\,\text{Mult}[aR, aL] + i\,e^{i\,t\,(\Delta+2\,\omega)}\,\lambda\,\text{Mult}[I\$m, aL] - i\,\lambda\,\text{Mult}[I\$m, aR] + i\,\lambda\,\text{Mult}[I\$p, aL] - i\,e^{-i\,t\,(\Delta+2\,\omega)}\,\lambda\,\text{Mult}[I\$p, aR]$$

In the interaction representation, the Hamiltonian is still time-dependent.
Compute a 0th-order average Hamiltonian by integrating in time over on period of oscillation of the

oscillatory terms.

The result agres

In[86]:=
```
H$0$T$avg = (Δ + 2 ω)/(2 π) Integrate[H$0$T, {t, 0, (2 π)/(Δ + 2 ω)}] // Expand
```

Out[86]=
$$\frac{Iz\,\Delta}{2} - \frac{1}{2}\,\Delta\,\text{Mult}[aR, aL] - i\,\lambda\,\text{Mult}[I\$m, aR] + i\,\lambda\,\text{Mult}[I\$p, aL]$$

In[87]:=
```
Clear[Ix, Iy, Iz, I$p, I$m, aR, aL, ω0, ω, λ, Δ]
```

Electron transfer

In[88]:=
```
Clear[Ix, Iy, Iz, I$p, I$m, aR, aL, Id, P1, P2, Δ, ω, g, ħ, ϵ$1, ϵ$2, J]
Clear[H$T, IdIz$rules, H$0, H$0$T, H$0$T$prime, H$J, H$J$T]

SpinBoson$CreateOperators[Ix, Iy, Iz, I$p, I$m, aR, aL];
```

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.

SpinBoson$CreateOperators: Creating operators.

SpinBoson$CreateOperators: Adding lp and lm commutations relations.

SpinBoson$CreateOperators: Adding lp and lm simplification rules.

SpinBoson$CreateOperators: Adding aL and aR normal ordering rule.

In[91]:=
```
CreateScalar[{Δ, ω, g, ħ, ϵ$1, ϵ$2, J}];
$Assumptions = {Δ ∈ Reals, ω1 ∈ PositiveReals, g ∈ PositiveReals};
```

Define a Hamiltonian we are going to use to transform into an interaction representation. This Hamiltonian will implement an electronic-state-dependent translation.

In[93]:=
```
H$T = -2 g I Mult[Iz, aR - aL ];
```

Take the I_+ and I_- operators into this interaction representation.

In[94]:=
```
Evolver[H$T, 1, #] & /@ {I$p, I$m}
```

Out[94]=
$$\left\{\text{Mult}\left[e^{2\,(aL-aR)\,g}, I\$p\right], \text{Mult}\left[e^{-2\,(aL-aR)\,g}, I\$m\right]\right\}$$

Create an identify operator which commutes with everything.

In[95]:=
```
CreateOperator[{{Id}}];
Id /: Comm[Id, _] := 0
Id /: Comm[_, Id] := 0
```

Create projection operators, P_1 and P_2.
Create a rule for writing the identity operator and I_z in terms of the projection operators.

In[98]:=
```
CreateOperator[{{P1, P2}}]
IdIz$rules = {Id → P1 + P2, Iz → 1/2 (P2 - P1)};
```

Define the electron transfer Hamiltonian.

In[100]:=
```
H$0 = ħ ω/2 (Mult[aR, aL] + Mult[aL, aR]) +
   1/2 (ϵ$1 + ϵ$2) Id + (ϵ$1 - ϵ$2) Iz - 2 g ħ ω Mult[aR + aL, Iz];
```

Transform the Hamiltonian into the interaction representation.

In[101]:=
```
H$0$T = Evolve[H$T, 1, H$0] /. Evolve → Evolver // MultSort //
  Collect[#, {Id, Iz, Mult[aR, aL]}, Simplify] &
```

Out[101]=
$$\text{Iz}\ (\epsilon\$1 - \epsilon\$2) + \frac{1}{2}\ \text{Id}\ (\epsilon\$1 + \epsilon\$2) + \frac{1}{2} \times \left(1 - 2\ g^2\right)\ \omega\ \hbar + \omega\ \hbar\ \text{Mult}[\text{aR, aL}]$$

Modify the above result a little, by inserting the identity operator .

In[102]:=
```
H$0$T$prime = Iz (ϵ$1 - ϵ$2) + 1/2 Id (ϵ$1 + ϵ$2) + 1/2 × (1 - 2 g^2 Id) ω ħ + ω ħ Mult[aR, aL];
```

Now simplify the result to obtain the textbook answer.
In the interaction representation, the coupling between the two-level systems at the vibrational system is absent.

In[103]:=
```
(H$0$T$prime /. IdIz$rules // Simplify ) //
 Collect[#, {P1, P2, Mult[aR, aL]}, Expand] &
```

Out[103]=
$$\frac{\omega\ \hbar}{2} + \text{P2}\ \left(\epsilon\$1 - g^2\ \omega\ \hbar\right) + \text{P1}\ \left(\epsilon\$2 - g^2\ \omega\ \hbar\right) + \omega\ \hbar\ \text{Mult}[\text{aR, aL}]$$

Define the electronic coupling Hamiltonian.

In[104]:=
```
H$J = - J (I$p + I$m);
```

Transform the electronic coupling Hamiltonian into the interaction representation .

In[105]:=
```
H$J$T = Evolve[H$T, 1, H$J] /. Evolve → Evolver
```

Out[105]=
$$-\text{J}\ \left(\text{Mult}\left[e^{-2\ (\text{aL}-\text{aR})\ g},\ \text{I\$m}\right] + \text{Mult}\left[e^{2\ (\text{aL}-\text{aR})\ g},\ \text{I\$p}\right]\right)$$

We would like to evolve the transformed electronic coupling Hamiltonian by the transformed Hamiltonian.
Transform first the operators in the transformed electronic coupling Hamiltonian.

In[106]:=
```
Evolver[H$0$T, t, #] & /@ {aL, aR, I$m, I$p}
```

Out[106]=
$$\left\{aL\, e^{i\, t\, \omega\, \hbar},\ aR\, e^{-i\, t\, \omega\, \hbar},\ e^{i\, t\, (\epsilon\$1-\epsilon\$2)}\ I\$m,\ e^{-i\, t\, (\epsilon\$1-\epsilon\$2)}\ I\$p\right\}$$

The Evolver algorithm is not quite smart enough to carry out this final evolution, although with the above transformed operators we can perform the evolution by inspection.
The following command just hangs ...

```
Evolver[H$0$T,t,H$J$T,  quiet -> False]
```

To perform the evolution “by inspection”, substitute the operators by the transformed operators. This procedure recovers the textbook answer.

In[107]:=
```
H$J$T  /. {aL → aL e^(i t ω ħ), aR → aR e^(-i t ω ħ), I$m → e^(i t (ϵ$1-ϵ$2)) I$m, I$p → e^(-i t (ϵ$1-ϵ$2)) I$p}
```

Out[107]=
$$-J\left(e^{i\, t\, (\epsilon\$1-\epsilon\$2)}\ \text{Mult}\left[e^{-2\left(-aR\, e^{-i\, t\, \omega\, \hbar}+aL\, e^{i\, t\, \omega\, \hbar}\right) g},\ I\$m\right]+e^{-i\, t\, (\epsilon\$1-\epsilon\$2)}\ \text{Mult}\left[e^{2\left(-aR\, e^{-i\, t\, \omega\, \hbar}+aL\, e^{i\, t\, \omega\, \hbar}\right) g},\ I\$p\right]\right)$$

In[108]:=
```
Clear[Ix, Iy, Iz, I$p, I$m, aR, aL, Id, P1, P2, Δ, ω, g, ħ, ϵ$1, ϵ$2, J]
Clear[H$T, IdIz$rules, H$0, H$0$T, H$0$T$prime, H$J, H$J$T]
```