

UniDyn--Demo-Scratch.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

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

EDIT THE FOLLOWING PATH STRING:

```
In[5]:= $NCPATH = "/Users/jam99/Dropbox";  
$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[11]:= $Path = AppendTo[$Path, $NCPATH];  
$Path = AppendTo[$Path, $UniDynPath];  
FindFile["UniDyn`"]  
FindFile["NC`"]
```

```
Out[13]= /Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/unidyn/UniDyn.m
```

```
Out[14]= /Users/jam99/Dropbox/NC/init.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[15]:= $VerboseLoad = False; (* Set to load quietly *)  
Needs["UniDyn`"]
```

NC: You are using the version of NCAIgebra which is found in: "/Users/jam99/Dropbox/NC/".

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

 NCAlgebra - Version 5.0.6
 Compatible with Mathematica Version 10 and above

Authors:

J. William Helton*
 Mauricio de Oliveira&

* Math, UCSD, La Jolla, CA
 & MAE, UCSD, La Jolla, CA

with major earlier contributions by:

Mark Stankus\$
 Robert L. Miller‡

\$ Math, Cal Poly San Luis Obispo
 ‡ General Atomics Corp

Copyright:

Helton and de Oliveira 2017
 Helton 2002
 Helton and Miller June 1991
 All rights reserved.

The program was written by the authors and by:

David Hurst, Daniel Lamm, Orlando Merino, Robert Obar,
 Henry Pfister, Mike Walker, John Wavrik, Lois Yu,
 J. Camino, J. Griffin, J. Oval, T. Shaheen, John Shopple.
 The beginnings of the program come from eran@slac.
 Considerable recent help came from Igor Klep.

Current primary support is from the
 NSF Division of Mathematical Sciences.

This program was written with support from
 AFOSR, NSF, ONR, Lab for Math and Statistics at UCSD,
 UCSD Faculty Mentor Program,
 and US Department of Education.

For NCAlgebra updates see:

www.github.com/NCAlgebra/NC
www.math.ucsd.edu/~ncalg

 **NCAlgebra:** All lower cap single letter symbols (e.g. a,b,c,...) were set as noncommutative.

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 and pretty-print the directory name.

```
In[17]:= SetDirectory[$UniDynPath];
TableForm[{{$UniDynPath}}, TableHeadings → {None, {"Directory"}}]
```

```
Out[18]//TableForm=
Directory
/Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/unidyn
```

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”). Pretty-print the names of the unit-test files included with the package.








```
In[19]:= fn = FileNames["*-tests.m"];
TableForm[{{$fn}}, TableHeadings → {None, {"Test files found"}}]
```

```
Out[20]//TableForm=
Test files found
Comm-tests.m
Evolve-tests.m
Mult-tests.m
OpCreate-tests.m
Osc-tests.m
Spins-tests.m
```

Finally, carry out the unit tests and make a report.

```
In[23]:= test$report = TestReport /@ fn;
TableForm[Table[test$report[[k]], {k, 1, Length[test$report]}]]
```

```
Out[24]//TableForm=
```

TestReportObject [ 	Title: Test Report: Comm-tests.m Success rate: 100% Tests run: 12]
TestReportObject [ 	Title: Test Report: Evolve-tests.m Success rate: 100% Tests run: 27]
TestReportObject [ 	Title: Test Report: Mult-tests.m Success rate: 100% Tests run: 20]
TestReportObject [ 	Title: Test Report: OpCreate-tests.m Success rate: 100% Tests run: 23]
TestReportObject [ 	Title: Test Report: Osc-tests.m Success rate: 100% Tests run: 20]
TestReportObject [ 	Title: Test Report: Spins-tests.m Success rate: 100% Tests run: 14]

Make a report.

```
tests$passed$total = Plus @@ (test$report[[#]]["TestsSucceededCount"] & /@
  List @@ Table[k, {k, 1, Length[test$report]}]);
tests$failed$total = Plus @@ (test$report[[#]]["TestsFailedCount"] & /@
  List @@ Table[k, {k, 1, Length[test$report]}]);

Print[Style[ToString[tests$passed$total] <> " tests passed",
  FontWeight → Bold, FontSize → 18, FontColor → Blue]]
Print[Style[ToString[tests$failed$total] <> " tests failed",
  FontWeight → Bold, FontSize → 18, FontColor → Red]]
```

116 tests passed

0 tests failed

Single spin: Taking repeated commutators

Create a single-spin system to play with.

```
In[29]:= Needs["UniDyn`"];
Clear[H, Ix, Iy, Iz, Δ, ω]
CreateScalar[{Δ, ω}];
SpinSingle$CreateOperators[Ix, Iy, Iz];

SpinSingle$CreateOperators: Creating spin operators.
SpinSingle$CreateOperators: Adding spin commutations relations.
SpinSingle$CreateOperators: No angular momentum L defined.
```

Define a Hamiltonian

```
In[33]:= H = Δ Iz + ω Ix;
```

Repeated commutators

Define a function to take n commutators of an operator Op with the $-i$ times the Hamiltonian H .

```
In[34]:= Clear[RepeatedComm];
RepeatedComm[1, H_, Op_] := List[Op];
RepeatedComm[n_, H_, Op_] := Prepend[RepeatedComm[n - 1, H, Op],
  -I Comm[H, RepeatedComm[n - 1, H, Op][[1]]]];

```

Example calculation of repeated commutators

```
In[37]:=  $\sigma$  = RepeatedComm[5, H, Iz] // Expand // Simplify;
 $\sigma$  // MatrixForm
```

```
Out[38]//MatrixForm=
```

$$\begin{pmatrix} \omega (-I_x \Delta + I_z \omega) (\Delta^2 + \omega^2) & & & \\ I_y \omega (\Delta^2 + \omega^2) & & & \\ \omega (I_x \Delta - I_z \omega) & & & \\ -I_y \omega & & & \\ I_z & & & \end{pmatrix}$$

Evolution

Use the Evolver function to calculate the evolution of the I_x operator under the on-resonance Zeeman Hamiltonian first.

```
In[39]:= $Assumptions = { $\Delta \in \text{Reals}$ ,  $\Delta > 0$ };
SetOptions[Evolver, quiet → True];
(* Set this to False when debugging. *)
Evolver[ $\Delta$  Iz, t, Ix]
```

```
Out[41]= Ix Cos[t  $\Delta$ ] + Iy Sin[t  $\Delta$ ]
```

As a check, use the Evolver function to calculate the evolution of the I_x operator under the off-resonance Zeeman Hamiltonian.

```
In[42]:= $Assumptions = { $\Delta \in \text{Reals}$ ,  $\Delta > 0$ ,  $\omega \in \text{Reals}$ ,  $\omega \geq 0$ };
SetOptions[Evolver, quiet → True];
(* Set this to False when debugging. *)
Evolver[H, t, Ix]
```

```
Out[44]=
```

$$\frac{1}{2 (\Delta^2 + \omega^2)^{3/2}} e^{-i t \sqrt{\Delta^2 + \omega^2}} \left(i I_y \Delta^3 - i e^{2 i t \sqrt{\Delta^2 + \omega^2}} I_y \Delta^3 + i I_y \Delta \omega^2 - i e^{2 i t \sqrt{\Delta^2 + \omega^2}} I_y \Delta \omega^2 + I_x \Delta^2 \sqrt{\Delta^2 + \omega^2} + e^{2 i t \sqrt{\Delta^2 + \omega^2}} I_x \Delta^2 \sqrt{\Delta^2 + \omega^2} - I_z \Delta \omega \sqrt{\Delta^2 + \omega^2} + 2 e^{i t \sqrt{\Delta^2 + \omega^2}} I_z \Delta \omega \sqrt{\Delta^2 + \omega^2} - e^{2 i t \sqrt{\Delta^2 + \omega^2}} I_z \Delta \omega \sqrt{\Delta^2 + \omega^2} + 2 e^{i t \sqrt{\Delta^2 + \omega^2}} I_x \omega^2 \sqrt{\Delta^2 + \omega^2} \right)$$

Some simplification is needed to get a nice-looking answer.

```
In[45]:= $Assumptions = {Δ ∈ Reals, Δ > 0, ω ∈ Reals, ω ≥ 0};
ρ = Collect[Expand[Simplify[ExpToTrig[Evolver[H, t, Ix]]]],
  {Ix, Iy, Iz}, Simplify]
```

$$\text{Out[46]} = \frac{I_x \left(\omega^2 + \Delta^2 \cos \left[t \sqrt{\Delta^2 + \omega^2} \right] \right)}{\Delta^2 + \omega^2} + \frac{2 I_z \Delta \omega \sin \left[\frac{1}{2} t \sqrt{\Delta^2 + \omega^2} \right]^2}{\Delta^2 + \omega^2} + \frac{I_y \Delta \sin \left[t \sqrt{\Delta^2 + \omega^2} \right]}{\sqrt{\Delta^2 + \omega^2}}$$

Quantum optics

Operators

```
In[47]:= Needs["UniDyn`"];
Clear[H, Ix, Iy, Iz, Δ, ω, g,
  F, aL, aR, Q$sym, P$sym, Q, P, QP$rules]
CreateScalar[{Δ, ω, Δω, g, F, ϕ}];
CreateOperator[{{Ix, Iy, Iz}, {aL, aR}}];
SpinSingle$CreateOperators[Ix, Iy, Iz, 1/2];
OscSingle$CreateOperators[aL, aR];

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

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[56]:= Q$sym /. QP$rules // Simplify
P$sym /. QP$rules // Simplify
```

Out[56]= Q

Out[57]= P

Hamiltonians

Free evolution

```
In[58]:= $Assumptions = {Element[ $\omega$ , Reals],  $\omega > 0$ };
H$0 =  $\frac{\omega}{2}$  (aR ** aL + aL ** aR);
In[60]:= Simplify[Evolver[H$0, t, Q$sym] /. QP$rules]
Out[60]= Q Cos[t  $\omega$ ] - P Sin[t  $\omega$ ]
```

Position or momentum kick

```
In[61]:= Clear[ $\delta x$ ,  $\delta p$ ];
$Assumptions = {Element[ $\delta x$ , Reals], Element[ $\delta p$ , Reals]};
H$0$x$kick =  $\delta x$  P$sym;
H$0$p$kick =  $\delta p$  Q$sym;
In[65]:= Simplify[Evolver[H$0$x$kick, t, Q$sym] /.
QP$rules ~Join~ {t → 1}] // Simplify
Simplify[Evolver[H$0$p$kick, t, P$sym] /.
QP$rules ~Join~ {t → 1}] // Simplify
Out[65]= Q -  $\delta x$ 
Out[66]= P +  $\delta p$ 
```

Phase Kick

```
In[67]:= $Assumptions = {Element[ $\omega$ , Reals],  $\omega > 0$ };
H$0$phase$kick =  $\frac{(\omega + \Delta\omega)}{2}$  (aR ** aL + aL ** aR);
In[69]:= Simplify[Evolver[H$0$phase$kick, t, Q$sym] /. QP$rules] /.
{ $\Delta\omega \rightarrow \Delta\phi / t$ } // Simplify
Out[69]= Q Cos[ $\Delta\phi + t \omega$ ] - P Sin[ $\Delta\phi + t \omega$ ]
```

Force

```
In[70]:= $Assumptions = {Element[F, Reals], F > 0};
H$1 = - F Q$sym;
```



```
In[72]:= Simplify[Evolver[H$1, t, Q$sym] /. QP$rules]
Simplify[Evolver[H$1, t, P$sym] /. QP$rules]
```

```
Out[72]= Q
```

```
Out[73]= P - F t
```

```
In[74]:= Simplify[Evolver[H$1, t, aR]]
Simplify[Evolver[H$1, t, aL]]
```

```
Out[74]= aR +  $\frac{i F t}{\sqrt{2}}$ 
```

```
Out[75]= aL -  $\frac{i F t}{\sqrt{2}}$ 
```

Squeezing

```
In[76]:= $Assumptions =
{Element[Δ, Reals], Δ > 0, Element[ω, Reals], ω > 0};
```

```
In[77]:= Simplify[Evolver[- $\frac{\Delta}{2}$  I (aR ** aR - aL ** aL), t, #] /. QP$rules] & /@
{Q$sym, P$sym}
```

```
Out[77]= {et Δ Q, e-t Δ P}
```

```
In[78]:= Simplify[Evolver[ $\frac{\Delta}{2}$  I (aR ** aR - aL ** aL), t, #] /. QP$rules] & /@
{Q$sym, P$sym}
```

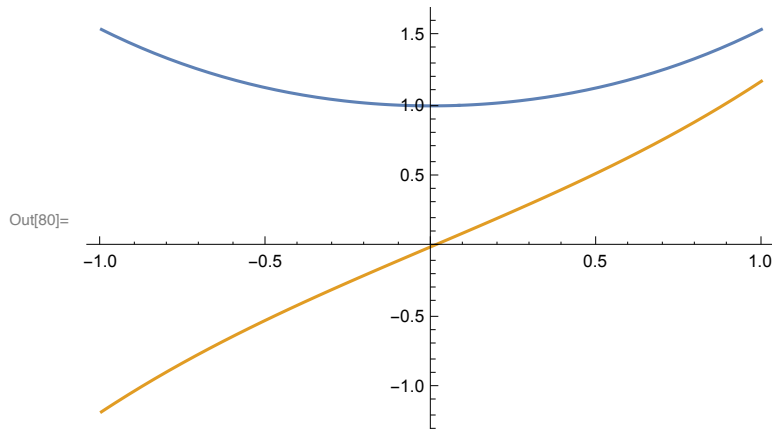
```
Out[78]= {e-t Δ Q, et Δ P}
```

```
In[79]:= Expand[Simplify[ExpToTrig[Evolver[ $\frac{\Delta}{2}$  (aR ** aR + aL ** aL), t, #] /.
QP$rules]]] & /@ {Q$sym, P$sym}
```

```
Out[79]= {Q Cosh[t Δ] + P Sinh[t Δ], P Cosh[t Δ] + Q Sinh[t Δ]}
```

Let's remind ourselves what the hyperbolic functions look like

```
In[80]:= Plot[{Cosh[T], Sinh[T]}, {T, -1, 1}]
```



```
In[81]:= Collect[Simplify[ExpToTrig[
  Evolver[ $\frac{\omega}{2} (aR ** aL + aL ** aR) - \frac{\Delta}{2} I (aR ** aR - aL ** aL),$ 
  t, #] /. QP$rules]], {Q, P}] & /@ {Q$sym, P$sym}
```

Out[81]=

$$\left\{ -\frac{P \omega \sinh\left[t \sqrt{\Delta^2 - \omega^2}\right]}{\sqrt{\Delta^2 - \omega^2}} + Q \left(\cosh\left[t \sqrt{\Delta^2 - \omega^2}\right] + \frac{\Delta \sinh\left[t \sqrt{\Delta^2 - \omega^2}\right]}{\sqrt{\Delta^2 - \omega^2}} \right), \right.$$

$$\left. \frac{Q \omega \sinh\left[t \sqrt{\Delta^2 - \omega^2}\right]}{\sqrt{\Delta^2 - \omega^2}} + P \left(\cosh\left[t \sqrt{\Delta^2 - \omega^2}\right] - \frac{\Delta \sinh\left[t \sqrt{\Delta^2 - \omega^2}\right]}{\sqrt{\Delta^2 - \omega^2}} \right) \right\}$$

Another quantum optics example

```
In[82]:= Needs["UniDyn`"];
Clear[λ, t, Ix, Iy, Iz, I$m, I$p, aL, aR, H];
CreateScalar[{λ, t}];
CreateOperator[{{Ix, Iy, Iz, I$m, I$p}, {aL, aR}}];
OscSingle$CreateOperators[aL, aR];
```

OscSingle\$CreateOperators: Oscillator operators already exist.

OscSingle\$CreateOperators: Adding oscillator commutations relations.

```
In[87]:= I$p /: Comm[I$p, I$m] = 2 I$z;
I$p /: Comm[I$p, I$z] = -I$p;
I$m /: Comm[I$m, I$p] = -2 I$z;
I$m /: Comm[I$m, I$z] = I$m;
I$z /: Comm[I$z, I$p] = I$p;
I$z /: Comm[I$z, I$m] = -I$m;
```

```
In[93]:= I$z /: NonCommutativeMultiply[a___, I$z, I$z, b___] :=
  1
  - NonCommutativeMultiply[a, b];
  4
I$p /: NonCommutativeMultiply[a___, I$p, I$p, b___] := 0;
I$m /: NonCommutativeMultiply[a___, I$m, I$m, b___] := 0;
```

```
I$p /: NonCommutativeMultiply[a___, I$p, I$z, b___] :=
  1
  - NonCommutativeMultiply[a, I$p, b];
  2
I$p /: NonCommutativeMultiply[a___, I$p, I$m, b___] :=
  1
  NonCommutativeMultiply[a, b] + NonCommutativeMultiply[a, I$z, b];
  2
```

```
I$m /: NonCommutativeMultiply[a___, I$m, I$z, b___] :=
  1
  NonCommutativeMultiply[a, I$m, b];
  2
I$m /: NonCommutativeMultiply[a___, I$m, I$p, b___] :=
  1
  NonCommutativeMultiply[a, b] - NonCommutativeMultiply[a, I$z, b];
  2
```

```
I$z /: NonCommutativeMultiply[a___, I$z, I$p, b___] :=
  1
  NonCommutativeMultiply[a, I$p, b];
  2
I$z /: NonCommutativeMultiply[a___, I$z, I$m, b___] :=
  1
  - NonCommutativeMultiply[a, I$m, b];
  2
```

```
In[102]:= H = λ (I$p ** aL + I$m ** aR);
Comm[I H, aL]
```

```
Out[103]= - i I$m λ
```

```

In[104]:= Comm[I H, Comm[I H, aL]]
Out[104]= 2 λ2 Iz ** aL

In[105]:= Comm[I H, aR]
Out[105]= i I$ p λ

In[106]:= Comm[I H, Comm[I H, aR]]
Out[106]= 2 λ2 Iz ** aR

In[107]:= Comm[I H, Comm[I H, Comm[I H, aL]]] // Simplify
Out[107]= -i λ3 (I$m - 2 I$m ** aL ** aR + 2 I$p ** aL ** aL)

In[108]:= Comm[I H, Comm[I H, Comm[I H, Comm[I H, aL]]]] // Simplify
Out[108]= -λ4 (3 aL - 4 Iz ** aL + 4 Iz ** aL ** aL ** aR + 4 Iz ** aL ** aR ** aL)

In[109]:= Evolver[λ Iz, t, I$p]
          Evolver[λ Iz, t, I$m]
Out[109]= e-i t λ I$p
Out[110]= ei t λ I$m

In[111]:= Evolver[λ (I$p ** aL + I$m ** aR), t, aL] // Simplify
          Evolver: Unrecognized evolution
Out[111]= {aL, i I$m λ, 2 λ2 Iz ** aL, i λ3 (I$m - 2 I$m ** aL ** aR + 2 I$p ** aL ** aL),
          -λ4 (3 aL - 4 Iz ** aL + 4 Iz ** aL ** aL ** aR + 4 Iz ** aL ** aR ** aL)}

In[112]:= Evolver[λ (I$m ** aR), t, aL] // Simplify
Out[112]= aL + i I$m t λ

In[113]:= Evolver[λ (I$p ** aL + I$m ** aR), t, (aL - aR)] // Simplify
          Evolver: Unrecognized evolution
Out[113]= {aL - aR, i (I$m + I$p) λ, 2 λ2 (Iz ** aL - Iz ** aR),
          i λ3 (I$m - I$p - 2 I$m ** aL ** aR + 2 I$m ** aR ** aR + 2 I$p ** aL ** aL - 2 I$p ** aR ** aL),
          -λ4 (3 aL - 3 aR - 4 Iz ** aL - 4 Iz ** aR + 4 Iz ** aL ** aL ** aR +
          4 Iz ** aL ** aR ** aL - 4 Iz ** aR ** aL ** aR - 4 Iz ** aR ** aR ** aL)}

```

Failing cases

Off-resonance evolution of Iz is touchy

Force with a phase factor

This fails because Mathematica cannot recognize that

$\text{Comm}[aL e^{-i\phi}, aR] \rightarrow e^{-i\phi} \text{Comm}[aL, aR] \rightarrow e^{-i\phi}$. Conclude that the `Comm[]` function is not yet smart enough to factor a *function of scalar* out of a commutator.

```
In[124]:= CreateScalar[{F, ϕ}];
$Assumptions = {Element[F, Reals], F > 0};
H$1 = F (Exp[-i ϕ] aL + Exp[i ϕ] aL);
```

In[127]:= **Evolver**[H\$1, t, aR]

Evolver: Unrecognized evolution

$$\begin{aligned}
 \text{Out[127]} = & \left\{ aR, -i F \left(\text{Comm}[aL e^{i\phi}, aR] + \text{Comm}[aL (e^{i\phi})^{-1}, aR] \right), \right. \\
 & -F^2 \left(\text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, aR]] + \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]] + \right. \\
 & \quad \left. \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, aR]] + \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]] \right), \\
 & i F^3 \left(\text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, aR]]] + \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, \right. \\
 & \quad \left. \text{Comm}[aL (e^{i\phi})^{-1}, aR]]] + \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, aR]]] + \right. \\
 & \quad \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]] + \\
 & \quad \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, aR]]] + \\
 & \quad \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]] + \\
 & \quad \left. \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, aR]]] + \right. \\
 & \quad \left. \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]] \right), \\
 & F^4 \left(\text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, aR]]]] + \right. \\
 & \quad \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]]] + \\
 & \quad \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, aR]]]] + \\
 & \quad \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]]] + \\
 & \quad \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, aR]]]] + \\
 & \quad \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]]] + \\
 & \quad \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, aR]]]] + \\
 & \quad \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]]] + \\
 & \quad \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, aR]]]] + \\
 & \quad \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]]] + \\
 & \quad \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, aR]]]] + \\
 & \quad \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]]] + \\
 & \quad \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL e^{i\phi}, aR]]]] + \\
 & \quad \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]]] + \\
 & \quad \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL e^{i\phi}, aR]]]] + \\
 & \quad \left. \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, \text{Comm}[aL (e^{i\phi})^{-1}, aR]]]] \right\}
 \end{aligned}$$