UniDyn--Demo-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

Tell *Mathematica* the path to the directory containing the packages. For the $NCPath variable, put the directly where the /NC folder is installed; the $NCPath name should not end with /NC.

EDIT THE FOLLOWING PATH STRINGS:

In[•]:=
```
$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[•]:=
```
$Path = AppendTo[$Path, $NCPath];
$Path = AppendTo[$Path, $UniDynPath];
FindFile["UniDyn`"]
FindFile["NC`"]
```

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

Out[•]= /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[*]:=
```
$VerboseLoad = True;
Needs["UniDyn`"]
```

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

Join: Incompatible elements in Join[⟨| intt → RowBox[{∫, RowBox[{■, RowBox[{𝑑, □}]}]}], dintt → RowBox[{SubsuperscriptBox[∫, ■, □], RowBox[{□, RowBox[{𝑑, □}]}]}], rintt → RowBox[{UnderscriptBox[∫, RowBox[{■, ∈, □}]], □}], sumt → RowBox[{UnderoverscriptBox[∑, RowBox[{■, =, □}], □], □}], ≪40≫, cK → TemplateBox[{}, CombinatorK], cS → TemplateBox[{}, CombinatorS], cW → TemplateBox[{}, CombinatorW], cY → TemplateBox[{}, CombinatorY] |⟩, ≪9≫, ≪3≫] cannot be joined.

Join: Incompatible elements in Join[⟨| intt → RowBox[{∫, RowBox[{■, RowBox[{𝑑, □}]}]}], dintt → RowBox[{SubsuperscriptBox[∫, ■, □], RowBox[{□, RowBox[{𝑑, □}]}]}], rintt → RowBox[{UnderscriptBox[∫, RowBox[{■, ∈, □}]], □}], sumt → RowBox[{UnderoverscriptBox[∑, RowBox[{■, =, □}], □], □}], ≪40≫, cK → TemplateBox[{}, CombinatorK], cS → TemplateBox[{}, CombinatorS], cW → TemplateBox[{}, CombinatorW], cY → TemplateBox[{}, CombinatorY] |⟩, ≪9≫, ≪4≫] cannot be joined.

```
-----------------------------------------------------------------
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. Ovall, 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.

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

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

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

Out[•]//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.

In[•]:=
```
tr = TestReport /@ fn;
TableForm[Table[tr [[k]], {k, 1, Length[tr]}]]
```

Out[•]//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.

In[158]:=
```
tests$passed$total =
  Plus @@ (tr[[#]]["TestsSucceededCount"] & /@ List @@ Table[k, {k, 1, Length[tr]}]);
tests$failed$total = Plus @@
   (tr[[#]]["TestsFailedCount"] & /@ List @@ Table[k, {k, 1, Length[tr]}]);
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

Execute the units tests one-by-one

Re-execute the tests in an order determined by us. This is useful for debugging. Running the *Evolve-test.m* file takes a minute.

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

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

```
In[•]:= $VerboseLoad = False;
        Needs["UniDyn`"]
```

```
In[•]:= TestReport[FileNames["OpCreate-tests.m"][[1]]]
```

```
Out[•]= TestReportObject[ Title: Test Report: OpCreate–tests.m
                          Success rate: 100%  Tests run: 23 ]
```

```
In[•]:= TestReport[FileNames["Mult-tests.m"][[1]]]
```

```
Out[•]= TestReportObject[ Title: Test Report: Mult–tests.m
                          Success rate: 100%  Tests run: 20 ]
```

```
In[•]:= TestReport[FileNames["Comm-tests.m"][[1]]]
```

```
Out[•]= TestReportObject[ Title: Test Report: Comm–tests.m
                          Success rate: 100%  Tests run: 12 ]
```

```
In[•]:= TestReport[FileNames["Spins-tests.m"][[1]]]
```

```
Out[•]= TestReportObject[ Title: Test Report: Spins–tests.m
                          Success rate: 100%  Tests run: 14 ]
```

```
In[•]:= TestReport[FileNames["Evolve-tests.m"][[1]]]
```

```
Out[•]= TestReportObject[ Title: Test Report: Evolve–tests.m
                          Success rate: 100%  Tests run: 27 ]
```

Congratulations

At this point you should have

(1) loaded the NCAlgebra and UniDyn packages and

(2) run the UniDyn units tests demonstrating that UniDyn is working as expected.