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 package.

EDIT THE FOLLOWING PATH STRING:

```
$PackagePath =
  "/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.

```
$Path = AppendTo[$Path, $PackagePath];
FindFile["UniDyn`"]
```

```
/Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/UniDyn/UniDyn.m
```

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

```
$VerboseLoad = True;
Needs["UniDyn`"]
```

```
You are using the version of NCAlgebra which is found in:

  /Users/jam99/Downloads/NC
```

```
You can now use "<< NCAlgebra`" to load NCAlgebra or "<< NCGB`" to load NCGB.
NCMultiplication.m loaded
NC1SetCommands.m loaded
NCInverses.m loaded
NCTransposes.m loaded
NCAdjoints.m loaded
NCCo.m loaded
NCRoots.m loaded
NC2SetCommands.m loaded
NCCollect.m loaded
NCSubstitute.m loaded
NCMonomial.m loaded
NCSolve.m loaded
NCTools.m loaded
NC2SimplifyRational.m loaded
NC1SimplifyRational.m loaded
NCSimplifyRational.m loaded
NCComplex.m loaded
NCMatMult.m loaded
NCDiff.m loaded
NCSchur.m loaded
NCAlias.m loaded
Grabs.m loaded
NCTaylorCoeff.m loaded
NCConvexity.m and NCGuts.m loaded
NCRealizationFunctions.m loaded
NCTeXForm.m loaded
NCTeX::Using 'open' as PDFViewer.
NCTeX.m loaded
NCMaster.m loaded
NCOutput.m loaded
```

```
------------------------------------------------------------
NCAlgebra - Version 4.0.6
Compatible with Mathematica Version 9

Authors:
  J. William Helton*
  Mauricio de Oliveira*
  Mark Stankus*
  Robert L. Miller⌗

* Math Dept, UCSD
⌗ General Atomics Corp
  La  Jolla, CA 92093

Copyright:
  Helton and Miller June 1991
  Helton 2002
  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.


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.
  Primary support in 2010 is from the
    NSF Division of Mathematical Sciences.

If you
  (1) are a user,
  (2) want to be a user,
  (3) refer to NCAlgebra in a publication, or
  (4) have had an interesting experience with NCAlgebra,
let us know by sending an e-mail message to

  ncalg@math.ucsd.edu.

We do not want to restrict access to NCAlgebra, but do
  want to keep track of how it is being used.

For NCAlgebra updates see the web page:

  www.math.ucsd.edu/~ncalg
------------------------------------------------------------
You are using the version of NCAlgebra which is found in:

  /Users/jam99/Downloads/NC

You can now use "<< NCAlgebra`" to load NCAlgebra or "<< NCGB`" to load NCGB.
```

```
You have already loaded NCAlgebra.m
```

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}].

```
You are using the version of NCAlgebra which is found in:

  /Users/jam99/Downloads/NC

You can now use "<< NCAlgebra`" to load NCAlgebra or "<< NCGB`" to load NCGB.

You have already loaded NCAlgebra.m
```

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.

```
You are using the version of NCAlgebra which is found in:

  /Users/jam99/Downloads/NC

You can now use "<< NCAlgebra`" to load NCAlgebra or "<< NCGB`" to load NCGB.

You have already loaded NCAlgebra.m
```

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

```
You are using the version of NCAlgebra which is found in:

  /Users/jam99/Downloads/NC

You can now use "<< NCAlgebra`" to load NCAlgebra or "<< NCGB`" to load NCGB.

You have already loaded NCAlgebra.m
```

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.

```
You are using the version of NCAlgebra which is found in:

  /Users/jam99/Downloads/NC

You can now use "<< NCAlgebra`" to load NCAlgebra or "<< NCGB`" to load NCGB.

You have already loaded NCAlgebra.m
```

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.

```
You are using the version of NCAlgebra which is found in:

  /Users/jam99/Downloads/NC

You can now use "<< NCAlgebra`" to load NCAlgebra or "<< NCGB`" to load NCGB.

You have already loaded NCAlgebra.m
```

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 and pretty-print the directory name.

```
SetDirectory[$PackagePath];
TableForm[{{$PackagePath}}, TableHeadings → {None, {"Directory"}}]
```

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.

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

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.

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

tests$run$total = Plus @@ (tr[[#]]["TestsSucceededCount"] & /@
     List @@ Table[k, {k, 1, Length[tr]}]);
Print[Style["Total test run = " <> ToString[tests$run$total ],
  FontWeight → Bold, FontSize → 18, FontColor → Blue]]
```

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: 23]

TestReportObject[Title: Test Report: Mult-tests.m
Success rate: 100% Tests run: 17]

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]

Total test run = 109

Execute the units tests one-by-one

AND execute the tests in an order determined by us. This is useful for debugging

```
SetDirectory[$PackagePath];
TableForm[{{$PackagePath}}, TableHeadings → {None, {"Directory"}}]
```

Directory
/Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/unidyn

```
$VerboseLoad = False;
Needs["UniDyn`"]
```

```
TestReport[FileNames["OpCreate-tests.m"][[1]]]
```

TestReportObject[Title: Test Report: OpCreate-tests.m
Success rate: 100% Tests run: 23]

```
TestReport[FileNames["Mult-tests.m"][[1]]]
```

TestReportObject[Title: Test Report: Mult-tests.m
Success rate: 100% Tests run: 17]

```
TestReport[FileNames["Comm-tests.m"][[1]]]
```

TestReportObject[Title: Test Report: Comm-tests.m
Success rate: 100% Tests run: 12]

```
TestReport[FileNames["Spins-tests.m"][[1]]]
```

TestReportObject[Title: Test Report: Spins-tests.m
Success rate: 100% Tests run: 14]

```
TestReport[FileNames["Evolve-tests.m"][[1]]]
```

TestReportObject[Title: Test Report: Evolve-tests.m
Success rate: 100% Tests run: 23]