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`"]
```

Execute the units tests in batch

Included with the package are a number of files, ending in "-tests.m", that con-

tain 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 : 17]

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 = 103

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 : Automatic
Success rate : 100 % Tests run : 23]

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

TestReportObject[Title : Automatic
Success rate : 100 % Tests run : 17]

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

TestReportObject[Title : Automatic
Success rate : 100 % Tests run : 12]

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

TestReportObject[Title : Automatic
Success rate : 100 % Tests run : 14]

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

TestReportObject[Title : Automatic
Success rate : 100 % Tests run : 17]