Spectral: User Manual

E.V.Corrêa Silva[a,*], G.A. Monerat[a], G. de Oliveira Neto[b], L.G. Ferreira Filho[a]

[a] *Depto. de Matemática, Física e Computação, Faculdade de Tecnologia, Universidade do Estado do Rio de Janeiro, Rod. Pres. Dutra km 298 s/n, 27537–000, Resende, RJ, Brazil*

[b] *Depto. de Física, Instituto de Ciências Exatas, Universidade Federal de Juiz de Fora, 36036–330, Juiz de Fora, MG, Brazil*

Abstract

This is the user manual to the `Spectral` package, a companion to the article *"Spectral: Solving the Schroedinger and Wheeler-DeWitt equations in the positive semi-axis by the spectral method"*, by the same authors.

Contents

*Corresponding author.

E-mail address: eduardo.vasquez@pq.cnpq.br

1. Introduction

This manual describes in detail each command of the *Spectral* package. The sequence of presentation has been guided by the examples, all of them referring to the solution of the Schroedinger equation for the quantum half-harmonic oscillator [1],

$$\left(-\frac{d^2}{dx^2} + kx^2\right)\Psi(x) = E\Psi(x,t), \tag{1}$$

for $x \geq 0$, in which k is a positive constant. The function $\psi(x)$ is the spatial part of the wave-function of a quantum system, and E is the energy eigenvalue.

The `Spectral` package offers tools for:

- defining a problem (section 2) and recalling that definition (section 3);
- defining or recalling the Octave executable file path (section 4), which is necessary if interaction with the Octave system is desired;
- solving the eigenvalue problem by Maple computation capabilities, with the option of using also Octave resources (section 5);
- displaying eigenvalues and eigenfunctions (sections 6 and 7);
- recalling the number of significant digits used in calculations (section 8);
- building a normalized wave function as an arbitrary superposition of eigenfunctions (section 9);
- calculating and plotting the probability density function (section 10 and 11);
- calculating the expected position (section 12) and its uncertainty for a given wave function, with the possibility of saving data to human-readable files (section 13), and plotting them (section 14);
- calculating the norm of the wave function as a function of time for checking purposes (section 15);
- checking the solution regarding the effective satisfaction of the eigenvalue equation and orthonormality of the eigenfunctions (sections 16 and 17);
- comparing the eigenvalue spectra of different problems (section 18).

In spite of our preference for long descriptive procedure names (for the sake of code readability), the reader is reminded of the `macro` and `alias` commands of Maple [2], which allow referring to those procedures by shorter names.

The command descriptions that follow refer to several equations contained in the paper, which are reproduced in this manual.

We address differential eigenvalue equations of the form

$$\left(-\frac{d^2}{dx^2} + f(x)\right)\psi(x) = E\ g(x)\psi(x). \tag{2}$$

for $x \in [0, \infty)$, in which $f(x)$ and $g(x)$ are known continuous functions of the spatial variable x, $\psi(x)$ is the spatial part of the wave-function of a quantum system, and E is the energy eigenvalue. The Schroedinger equation is obtained as a special case, by letting $g(x) = 1$. Restricting the domain

to $x \in [0, L]$ is equivalent to placing two infinite impenetrable barriers at $x = 0$ and $x = L$, thus imposing the boundary conditions $\psi(0) = \psi(L) = 0$.

An (approximate) eigenfunction $\psi_p(x)$ corresponding to the energy level E_p is expanded in the orthonormal basis of sine functions as

$$\psi_p(x) \approx \sum_{n=1}^{N} A_n^{(p)} \sqrt{\frac{2}{L}} \sin\left(\frac{n\pi x}{L}\right), \tag{3}$$

in which the coefficients $A_n^{(p)}$ are yet to be determined, and a finite number N of basis functions has been chosen.

The inner product of functions is defined as

$$(\Psi, \Phi) = \int_0^\infty dx \; \Psi^*(x)\, w(x) \Phi(x)\,, \tag{4}$$

in which the weight function $w(x)$ is chosen so that the Hamiltonian operator is self-adjoint.

The combination of the eigenvalue differential equation (2) and the expansion (3) yields the matricial eingenvalue equation

$$\Delta\, A^{(p)} \; = E_p\, \Gamma A^{(p)}\,, \tag{5}$$

in which $A^{(p)}$ is a $1 \times N$ column vector with elements $A_m^{(p)}$, Δ is a $N \times N$ square matrix with elements

$$\Delta_{n,m} = \left(\frac{n\pi}{L}\right)^2 \delta_{n,m} + \Phi_{n,m} \tag{6}$$

so that

$$\Phi_{n,m} = \frac{2}{L} \int_0^L \sin\left(\frac{n\pi x}{L}\right) f(x) \sin\left(\frac{m\pi x}{L}\right) dx \tag{7}$$

and Γ is the $N \times N$ square matrix with elements given by

$$\Gamma_{n,m} = \frac{2}{L} \int_0^L \sin\left(\frac{n\pi x}{L}\right) g(x) \sin\left(\frac{m\pi x}{L}\right) dx. \tag{8}$$

The inner product (4), combined with the expansion (5), leads to the definition of the dot product

$$A^{(p)} \cdot A^{(q)} \equiv \sum_{n=1}^{N} \sum_{m=1}^{N} A_n^{(p)} W_{nm} A_m^{(q)} \tag{9}$$

in which the weight matrix W is obtained from the weight function $w(x)$ as

$$W_{nm} \equiv \frac{2}{L} \int_0^L dx \;\; \sin\left(\frac{n\pi x}{L}\right) \;\; w(x) \;\; \sin\left(\frac{m\pi x}{L}\right). \tag{10}$$

2. Defining a problem

Command.

```
Problem
```

Calling sequence.

```
Problem(Ffunction = fx, Gfunction=gx, Variable=x, Length=L,
   NumberOfLevels=N, Weight=wx, Label=Lb, NumberOfDigits=N)
```

Parameters.

- `Ffunction = fx`, in which `fx` is a maple expression representing the function $f(x)$ of Eq.(2).
- `Gfunction = gx`, in which `gx` is a maple expression representing the function $g(x)$ of Eq.(2).
- `Variable = x`, in which `x` is a maple name representing the independent spatial variable x of Eq.(2).
- `Length=L`, in which `L` is a real positive number representing the upper interval amplitude/upper limit L of Eq.(3).
- `NumberOfLevels = N`, in which `N` is a positive integer number representing the number N of basis functions used in the expansion of the eigenfunction $\psi_p(x)$ in Eq.(3).
- `Weight = wx`, (optional) in which `wx` is a maple expression representing the weight function $w(x)$ in the inner product (4). Its default value is $w(x) = 1$.
- `Label = Lb`, (optional), in which `Lb` is a string that will be used as root name for all the output files related to this problem. Its default value is `"Problem"`.
- `NumberOfDigits=N`, (optional), in which `N` is a positive integer for the number of digits to be used in calculations executed by Maple. Its default value [1] is `15`. Octave calculations are not affected by this option.

The order of appearance of these parameters is irrelevant.

Description.
The input data is interpreted and parameters that characterize the eigenvalue problem (such as the functions $f(x)$ and $g(x)$ in Eq.(2)) *and* its approximate solution (such as the number of levels to be used) are defined. A *problem-module*, i.e., a *module* data structure [2] that represents the problem, is returned. The problem-module also provides the functionality for solving the problem and manipulating its solution. For checking purposes, `Problem` recalls the problem definition data to the user.

Example.
The following input line sets up the half-harmonic oscillator problem

$$\left(-\frac{d^2}{dx^2} + kx^2\right)\Psi(x) = E\Psi(x), \tag{11}$$

for $k = 1$ over the interval $x \in [0, 10]$ with 100 levels, and assigns it to the variable `p`.

```
> p := Problem(Ffunction = x^2, Gfunction=1, Variable = x, Length=10,
NumberOfLevels = 100, Weigth=1, Label="HalfOscillator"):
```

```
Problem definition

Number of energy levels:, 100
Interval amplitude:  , 10, (float:  , 10., )
f function:, x^2
g function:, 1
w function (weight):, 1
Independent variable:  , x
Label :, "HalfOscillator"
Number of Digits to be used:, 15
```

1. A warning message will be issued if the user chooses a value higher than 15, on the possibility of a "kernel connection has been lost", observed in Maple 13.

```
> type(p,‘module‘); #A module has been assigned to p.
```

```
true
```

In the remaining examples of this manual, we will refer to the same problem defined here, stored in the *problem-module* `p`.

3. Recalling a problem description

Command.

`IsDescribed`

Calling sequence.

`p:-IsDescribed()`

Here, `p` is a problem-module (see section 2 of this manual).

Description.
A description of the problem, similar to the one issued when `Problem` was invoked, is shown. Additional information on the calculation status is yielded.

Example.

```
> p:-IsDescribed();
```

```
Problem definition

Number of energy levels:, 100
Interval amplitude:  , 10, (float:  , 10., )
f function:, x^2
g function:, 1
w function (weight):, 1
Independent variable:  , x
Label :, "HalfOscillator"
Number of Digits to be used:, 15
The energy spectrum has not been calculated yet.
```

4. Defining or showing the Octave path

Command.

`octavedir`

Calling sequences.

`octavedir()`

`octavedir(path)`

Parameters.

– `path` is a string or name describing the path in which the Octave executable file[2] can be found. Notice that a backslash (\) must be written as a double backsklash (\\) or a single normal slash (/).

Description.
Called with no arguments, the command `octavedir` shows the path currently defined for the Octave executable file. If invoked with a single argument, it wiil define that path. Using `octavedir` is mandatory if calculations are to be carried out by Octave; otherwise, it is optional. (See section 5 of this manual.)

Example.
```
> octavedir("C:/Programs/Octave/bin");# Defines the Octave path.
                                        "C:/Programs/Octave/bin"

> octavedir();# Shows Octave path.
                                        "C:/Programs/Octave/bin"
```

5. Calculating approximate eigenvalues and eigenfunctions

Commands.

`IsSolved`

`IsSolvedByOctave`

Calling sequences.

`p:-IsSolved()`

`p:-IsSolved(NumberOfDigits = d)`

`p:-IsSolvedByOctave()`

Here, `p` is a problem-module (see section 2 of this manual).

Parameters.

– `NumberOfDigits=d`, in which `d` is a positive integer representing the initial number of digits Maple should use in calculations. It is the value initially[3] assigned to the environment variable `Digits`; by default, that value is 10. This option is not available for calculations with Octave, which works with the fixed number of 15 digits.

Description.
Approximate eigenvalues and eigenfunctions can be calculated by the procedures `IsSolved` or `IsSolvedbyOctave`, exported by the problem-module. The `IsSolved` procedure relies exclusively on Maple resources, whereas `IsSolvedbyOctave` generates a script file in Octave code [3] for the solution of the matricial eigenvalue equation (5). The script is interpreted by the Octave engine, upon automatic request of the Maple system. The output files thus produced are then read by Maple, and the solution is processed and stored in the problem-module. The solution is not shown; access to it is described in the subsequent sections of this manual. For more details on the Maple-Octave interaction, the reader is referred to our paper.

2. For Octave 3.2.4 for Windows, this is the file `octave-3.2.4.exe`.

3. In earlier versions of Maple such as Maple 12, spectrum calculations would (wrongly) yield complex eigenvalues, which could only be corrected by increasing the number of digits used (with a higher cost of calculation, naturally). If not all eigenvalues obtained are real, the number of digits will be automatically increased by about 10%, and the calculation will be repeated until a real spectrum is obtained.

Example.

```
> p:-IsSolvedByOctave();

                                        Time elapsed:  , 3.292, seconds
                The eigenvalue/eigenvector problem has been completely solved with 15 digits used.
```

(The actual elapsed time may vary.) On the other hand, by invoking the command

```
> p:-IsSolved();
```

calculations will rely solely on Maple resources.

Some differences in precision may be found in results for different versions of Maple, due to differences in the way Maple native procedures deal with the numerical integrals and/or handle linear systems of equations.

Occasionally, the command `CheckSolutions:-Numerically` may issue a warning message like *"warning: scalar product of levels [...] yielded a complex number. I will record just the real part"*. This is due to the way different versions of Maple deal with the numeric integrals involved in the calculation of scalar products. These integrals are calculated by Maple native procedures; sometimes they issue a small spurious imaginary part or even a null float imaginary part ($0.\ I$) which is not automatically discarded. By "small", we mean a number which is of order 10^{-D}, in which D is about the number of significant digits used. It gets smaller as D increases; so, we know it is a numerical artifact that can be safely discarded. The same holds for the result $0.\ I$ issued by Maple, indicating that such quantity is null *up to* the working precision, so it shouldn't simply vanish from sight. By the very definition of the scalar product, addressed in the warning message, we know that it is a real number; so, any small spurious imaginary part that eventually shows up can be safely ignored.

6. Displaying eigenvalues

Command.

`ItsEigenvalues`

Calling sequences.

`p:-ItsEigenvalues()`

`p:-ItsEigenvalues(n)`

`p:-ItsEigenvalues(n..m)`

Here, `p` is a problem-module.

Parameters.

– `n` and `m` are positive integers.

Description.
When called with no parameters, `ItsEigenvalues` returns the list of all calculated eigenvalues in ascending order. With a single positive integer parameter `n`, the n-th lowest parameter is returned. With a range of positive integer parameters `n` and `m`, the sequence from the n-th to the m-th lowest eigenvalues is returned.

Example.

```
> p:-ItsEigenvalues();
```

$$[2.999999999999684, 6.999999999999813, 10.99999999999995, 15, 19.00000000000002,$$
$$\textit{(removed output)},$$
$$981.7557676524347, 1003.909496673033, 1039.651576236899]$$

```
> p:-ItsEigenvalues(1); # 1st lowest eigenvalue.
```

$$2.999999999999684$$

```
> p:-ItsEigenvalues(3..5);# 3rd to 5th lowest eigenvalues.
```

$$10.99999999999995,\ 15,\ 19.00000000000002$$

The analytical prediction for the energy eigenvalues E_n for this potential is well-known[1],

$$E_n = 2\,(n+1)\,. \quad n \in \{1, 3, 5, \ldots\}. \tag{12}$$

Let us compute the 10 lowest levels from that expression,

```
> Prediction10 := seq(2*(nn + 1/2),nn=[seq(2*(i-1)+1,i=1..10)]);
```

$$\texttt{Prediction10} := 3, 7, 11, 15, 19, 23, 27, 31, 35, 39;$$

and compare them to the 10 lowest eigenvalues that have been calculated by the package.

```
> Calculated10 := p:-ItsEigenvalues(1..10);
# 10 lowest eigenvalues
```

$$\texttt{Calculated10} := 2.999999999999684, 6.999999999999813, 10.99999999999995,\ 15, 19.00000000000002, 22.99999999999997,$$
$$26.99999999999996, 30.99999999999994, 35.00000000000004, 39;$$

```
> Calculated10 - Prediction10;
```

$$-3.2\ 10^{-13}, -1.9\ 10^{-13},\ 0., 0., 0., 0., 0., -1.\ 10^{-13}, 0., 0$$

The largest error is of order 10^{-13}.

7. Displaying eigenfunctions

Command.

`ItsEigenfunction`

Calling sequence.

`p:-ItsEigenfunction(x,n)`

Parameters.

- `x` is a name that represents the spatial variable.
- `n` is positive integer wich represents the order of the eigenfunction.

Here, `p` is a problem-module (see section 2 of this manual).

Description.
The n-th normalized approximate eigenfunction is returned, as a function of the spatial variable x.

Example.
The analytical expression of the eigenfunctions for the half-harmonic oscillator (1) is well-known [1],

$$\psi_n(x) = \frac{\sqrt{2}}{\sqrt{2^n n!}} \left(\frac{1}{\pi}\right)^{1/4} \; H_n(x) e^{-x^2/2}, \tag{13}$$

for $n \in \{1, 3, 5, \ldots\}$ in which $H_n(x)$ is the $n-$th Hermite polynomial. The extra factor $\sqrt{2}$ accounts for the normalization over $x \in [0, \infty)$, rather than $x \in (-\infty, \infty)$. Let us compare, say, the eigenfunction associated to the 2nd lowest energy obtained by this formula to the one calculated numerically.

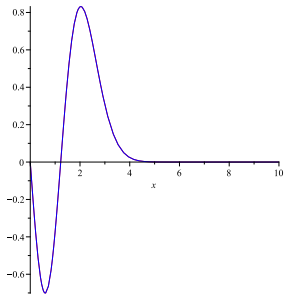
```
> F := p:-ItsEigenfunction(x,2):  #2nd eigenfunction, approximate.
> sort(F); # Just for the lexical ordering the sin arguments.
```

$$\tfrac{1}{5}\sqrt{5}\,(-0.130007759252790\,\sin(\pi\,x) - 4.38893458950433\;10^{-7}\,\sin(2\,\pi\,x) + \\ \textit{(removed output)} \\ -5.55480467868582\;10^{-18}\sin\left(\tfrac{97}{10}\pi\,x\right) + 5.42625230709934\;10^{-18}\,\sin\left(\tfrac{99}{10}\,\pi\,x\right))$$

```
> F1 := radsimp(subs(n=3, sqrt(2)*(1/Pi)^(1/4)*1/sqrt(2^ n * n!)  *
HermiteH(n, x) * exp(-x^2/2)));
# Analytical expression, simplified with radsimp.
```

$$\frac{1}{12}\,\frac{\sqrt{6}\,\mathrm{HermiteH}(3,x)\,e^{-\frac{1}{2}x^2}}{\pi^{1/4}}$$

```
> plot([F,F1],x=0..10,color=[red,blue]);
```



The curves show good agreement. (The approximate result `F` may differ by a phase $e^{i\pi}$ from the exact expression `F1`.)

The tools for checking the orthonormality of the approximate eigenfunctions are treated in sections 16 and 17 of this manual.

8. Recalling the number of digits used in calculations

Command.

`ItsNumberOfDigits`

Calling sequence.

`p:-ItsNumberOfDigits()`

Here, `p` is a problem-module (see section 2 of this manual).

Description.
The value assigned to the environment variable `Digits` during calculations is returned. For calculations with `IsCalculatedByOctave` this always amounts to 15; for calculations with `IsCalculated`, the actual value may vary.

Example.
```
> p:-ItsNumberOfDigits();
```

15

9. Building a wave function

Command.

`WaveFunction`

Calling sequence.

`p:-WaveFunction(x,t,ListOfCoefficients)`

Parameters.

- `x`, a name or real number representing the position in which the wave function is evaluated.
- `t`, a name or real number representing the time in which the wave function is evaluated.
- `ListOfCoefficients`, a list of N names or complex numbers representing the coefficients C_i of the linear combination (14).

Description.
A *wave function* or *wave package* $\Psi(x,t)$ is a (normalized) linear combination of eigenstates $\psi_n(x,t)$, which represent a quantum state of the system at any time t. Its general form for time-independent potentials (as in our context, because $f(x)$ and $g(x)$ in Eq.(2) do not depend on time) and taking into consideration only the N lowest-energy eigenstates is

$$\Psi(x,t) = \sum_{n=1}^{N} C_n \psi_n(x) e^{-iE_n t}. \tag{14}$$

The complex coefficients C_n can be determined (up to a phase), for instance, by the initial wave package

$$\Psi(x,0) = \sum_{n=1}^{N} C_n \psi_n(x) \;\longrightarrow\; |C_n|^2 = \int_0^{\infty} dx\; \psi_n^*(x) w(x) \Psi(x,0) \tag{15}$$

in which $w(x)$ is the weight function. They should also satisfy the normality condition

$$\int_0^{\infty} dx\; \Psi^*(x,t) w(x) \Psi(x,t) = 1, \tag{16}$$

which leads to

$$\sum_{i=1}^{N} |C_i|^2 = 1. \tag{17}$$

The package `Spectral` lets one choose arbitrary coefficients C_i for the eigenvalues, automatically normalizing the wave function. For instance, an equiprobable superposition of the N states can be set by choosing $C_i = 1$, for $i = \{1, 2, \ldots, N\}$.

Example.
The following example returns the normalized wave function corresponding to the superposition of the three lowest levels with arbitrary coefficients $(C1, C2, C3)$, for arbitrary position x and time t.

```
> p:-WaveFunction(x,t,[C1,C2,C3,0$97]);
```

10. Calculating the probability density

Command.

```
ProbabilityDensity
```

Calling sequence.

```
p:-ProbabilityDensity(x,t,ListOfCoefficients)
```

Parameters.

- `x`, a name or real number representing the position in which the wave function is evaluated.
- `t`, a name or real number representing the time in which the wave function is evaluated.
- `ListOfCoefficients`, a list of N names or complex numbers representing the coefficients C_i of the linear combination (14).

Description.
The expression of the probability density $\Psi(x,t)^*\Psi(x,t)$ for a wave function $\Psi(x,t)$ (represented by the list of coefficients in `ListOfCoefficients`) at position `x` and time `t` is returned. The wave function is automatically normalized, and calling `WaveFunction` is not necessary.

Example.
The following example returns the probability density corresponding to the wave package in which the three lowest levels are combined, with arbitrary coefficients $(C1, C2, C3)$, for arbitrary position x and time t.

```
> p:-ProbabilityDensity(x,t,[C1,C2,C3,0$97]);
```

11. Plotting the probability density

Command.

```
ProbabilityDensityIsPlotted
```

Calling sequences.

```
p:-ProbabilityDensityIsPlotted(t,ListOfCoeffs)

p:-ProbabilityDensityIsPlotted(t1..t2,ListOfCoeffs)

p:-ProbabilityDensityIsPlotted(t1..t2,ListOfCoeffs,NumFrames)
```

Here, `p` is a problem-module (see section 2 of this manual).

Parameters.

- `t`, real constant representing a single instant of time.
- `t1` and `t2`, real constants representing the initial and final instants of the time interval $[t1, t2]$. Used for animations of the probability density.
- `ListOfCoeffs`, a list of N complex numbers representing the coefficients C_i of the wave function (14). The wave function is normalized automatically.
- `NumFrames` (optional), a positive integer that defines the number of frames in the animation. The default value is `NumFrames=10`. The corresponding time step is

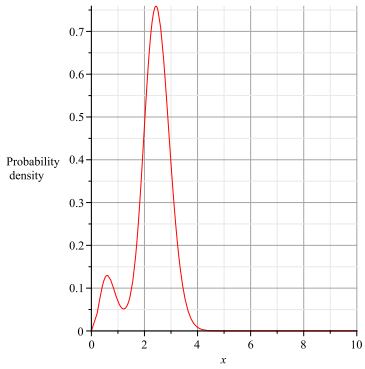
$$\Delta t = (t2 - t1)/NumFrames$$

.

Description.
If a single instant of time t is given, the plot of the probability density versus position for that instant of time is returned. If a range of time t is given, then an animated plot of the probability density versus position within that interval of time is shown. The option `NumFrames` controls the number of frames in the animation. The spatial interval to be shown is defined by the problem-module.

Examples.
In what follows, we consider an equiprobable superposition (i.e., a superposition with equal coefficients, all set to unity) of the three lowest levels. The probability density plot at $t = 0$ is obtained as follows.

```
> p:-ProbabilityDensityIsPlotted(0,[1,1,1,0$97]);
```



Animated plots can be obtained by entering a time range, rather than a time instant. For $t \in [0, 20]$, for instance, we would have

```
> p:-ProbabilityDensityIsPlotted(0..20,[1,1,1,0$97]);
```

By default, the number of frames in animations is set to 10; an optional 3rd argument may define it otherwise. Animations with 20 frames, for instance, can be obtained as follows.

```
> p:-ProbabilityDensityIsPlotted(0..20,[1,1,1,0$97],20);
```

Moreover, `ProbabilityDensityIsPlotted` accepts other `plot` options as extra arguments.

12. Expected position and uncertainty

Commands.

`ExpectedPosition`

`PositionUncertainty`

Calling sequences.

`p:-ExpectedPosition(ListOfCoeffs)`

`p:-PositionUncertainty(ListOfCoeffs)`

Parameters.

– `ListOfCoeffs`, a list of N complex numbers representing the coefficients C_i of the wave function (14). The wave function is normalized automatically.

Here, `p` is a problem-module (see section 2 of this manual).

Description.
For a given wave function $\Psi(x,t)$ (represented by its list of coefficients `ListOfCoeffs`) the expected position

$$\langle x \rangle (t) = \int_0^\infty dx\ \Psi^*(x,t)\ w(x)\ x\ \Psi(x,t), \tag{18}$$

as a function of time is returned by `ExpectedPosition` in the form of a procedure. Likewise, `PositionUncertainty` returns a procedure for the uncertainty on the position

$$\begin{aligned} \sigma(t) &= \sqrt{\langle x^2 \rangle - \langle x \rangle^2} \\ &= \left[\int_0^\infty dx\ \Psi^*(x,t)\ w(x)\ x^2\ \Psi(x,t) \right. \\ &\quad \left. - \left(\int_0^\infty dx\ \Psi^*(x,t)\ w(x)\ x\ \Psi(x,t) \right)^2 \right]^{1/2}. \end{aligned} \tag{19}$$

Examples.

```
> av := p:-ExpectedPosition([1$3,0$97]):
#Superposition of the 3 lowest levels.
> av(0); #Expected value at t t=0.
```

```
                    2.30450998041029
```

```
> un := p:-PositionUncertainty([1$3,0$97]):
> un(0); #Uncertainty at t=0.
```

```
                    0.704586627830134
```

13. Calculating the expected value and uncertainty of position and saving data to a file

Commands.

`ExpectedPositionAndUncertaintyAreCalculated`

`ExpectedPositionIsCalculated`

Calling sequences.

```
p:-ExpectedPositionAndUncertaintyAreCalculated(t1..t2,
ListOfCoeffs)

p:-ExpectedPositionAndUncertaintyAreCalculated(t1..t2,
ListOfCoeffs, NumSubintervals)

p:-ExpectedPositionIsCalculated(t1..t2, ListOfCoeffs)

p:-ExpectedPositionIsCalculated(t1..t2, ListOfCoeffs,
NumSubintervals)
```

Here, `p` is a problem-module (see section 2 of this manual).

Parameters.

- `t1` and `t2`, real constants representing the initial and final instants of the time interval $[t1, t2]$.
- `ListOfCoeffs`, a list of N complex numbers representing the coefficients C_i of the linear combination (14).
- `NumSubintervals` (optional), a positive integer that defines the number of sub-intervals of time to be used. The corresponding time step is $\Delta t = (t2 - t1)/NumSubintervals$. The default value is `NumSubintervals=10`.

Description.
The procedure `ExpectedPositionAndUncertaintyAreCalculated` calculates the values of the expected position and its uncertainty for a given wave function, represented here by the list of coefficients `ListOfCoeffs`, for a discrete set of time instants within a given time interval $[t1, t2]$, defined by the number of time subintervals `NumSubIntervals` (its default value is 10). For each instant of time, the expected value and uncertainty of the position are calculated and saved in a human-readable text file, the name of which is a concatenation of the root file name (see the `Label` option in `Problem`, section 2 of this manual) and the suffix `"_PositionUncertainty"`.

The procedure `ExpectedPositionIsCalculated` does a similar job, but it only calculates the expected position. The file thus produced has the suffix `"_Position"`, instead.

Example.
In what follows, the expected position and uncertainty for a equiprobable superposition of the three lowest-energy states are calculated within the time interval $[0, 10]$, for 100 time subintervals, and then saved to the file `HalfOscillator_Position_Uncertainty`

```
> p:-ExpectedPositionAndUncertaintyAreCalculated(0..10,[1$3,0$97], 100);
            "Expected position and uncertainty data stored in file:", "HalfOscillator_PositionUncertainty"
```

The ouput file has the typical structure shown below.

```
# Filename :  HalfOscillator_PositionUncertainty
# Program version :Spectral_1.0
# Start:  Sat May 19 15:39:10 2012
# (1) time, (2) expected position, (3) expected position-uncertainty, (4) expected position+uncertainty
#=============================================================================
 0    0.849253985121159 0.162552373918615  1.53595555963237
 0.1  0.927519660226608 0.210144260043562  1.64489506040965
 0.2  1.14061480364454  0.371491808303762  1.90973779898532
 ..............<deleted output>..............
 10   2.13841615739507  1.45581704621637   2.82101526857377
```

The file heading is composed of a few Maple comment lines (having # as their first character), in which the user can find the name of the file, the program version that created it, the starting date and time of computation, and also a brief reminder of the meaning of the subsequent columns of data, in the remaining of the file. The first column shows the instants of time; the second column shows the expected value of position $< x >$; the third and fourth column show, respectively, that expected position plus and minus the uncertainty σ. Columns are separated by the ASCII TAB character.

14. Plotting the expected value and uncertainty of position

Commands.

`ExpectedPositionIsPlotted`

`ExpectedPositionAndUncertaintyArePlotted`

Calling sequences.

`p:-ExpectedPositionIsPlotted()`

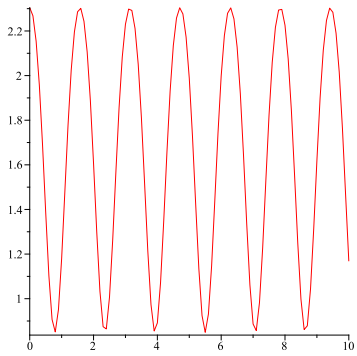
`p:-ExpectedPositionAndUncertaintyArePlotted()`

Here, `p` is a problem-module (see section 2 of this manual).

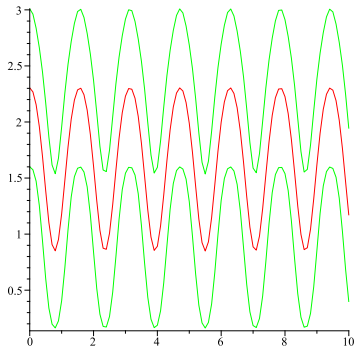
Description.
The procedure `ExpectedPositionIsPlotted` reads the data file created by the commands `ExpectedPositionIsCalculated` or `ExpectedPositionAndUncertaintyAreCalculated` (see section 13 of this manual) and then plots the expected value $\langle x \rangle$ as a function of time. In its turn, the procedure `ExpectedPositionAndUncertaintyArePlotted` plots the expected value $\langle x(t) \rangle$ curve (in red) as well as the as the curves $\langle x(t) \rangle \pm \sigma(t)$ (in green), in which $\sigma(t)$ is the uncertainty as a function of time. The spatial interval is determined by the problem-module.

Examples.

```
> p:-ExpectedPositionIsPlotted();
```




```
> p:-ExpectedPositionAndUncertaintyArePlotted();
```



15. Calculating the norm of the wave function

Command.

`NormOfWaveFunction`

Calling sequence.

```
p:-NormOfWaveFunction(ListOfCoeffs)
```

Here, `p` is a problem-module (see section 2 of this manual).

Parameters.

- `ListOfCoeffs`, a list of N complex numbers representing the coefficients C_i of the linear combination (14).

Description.
A procedure for calculating the norm of the wave function, represented by the list of numeric coefficients `ListOfCoeffs`, is returned. Such procedure accepts a single argument, that represents the time variable. Generally, the conservation of the norm is used as a check of the accuracy of numerical results. The wave function is automatically normalized.

Examples.
In what follows, an equiprobable superposition of the three lowest-energy states is considered. A procedure that represents its norm as a function of time is generated and assigned to n, and then evaluated for several instants of time.

```
> n := p:-NormOfWaveFunction([1,1,1,0$97]):
> seq(n(t),t=0..10);
```

1.00000000000001, 1.00000000000001, 1.00000000000000, 1.00000000000000,
1.00000000000001, 1.00000000000001, 1.00000000000001, 1.00000000000001, 1.00000000000000, 1.00000000000001, 1.00000000000001

16. Checking solutions

Command.

```
CheckSolutions:-Numerically
```

Calling sequence.

```
p:-CheckSolutions:-Numerically()
```

Description.
Tests on the eigenvalue spectra an on the orthonormality of eigenfunctions, described in detail in our paper, can performed by the procedure `CheckSolutions:-Numerically`. The results of these checkings are saved in three report files, in the working directory. The names of those files are generated by concatenation of the root file name (defined by `Problem` through the `Label` option, as shown in section 2 of this manual) with the suffixes `_Check_Equations`, `_Check_Normality` and `_Check_Orthogonality`. The aforementioned files are human-readable text files; please refer to the examples below. Moreover, they can be read by the procedure `CheckSolutions:-Graphically` so that their data be displayed in graphical form, as described in section 17 of this manual.

Examples.

```
> p:-CheckSolutions:-Numerically();
```

```
                    "Reporting errors in equations in file:  ",HalfOscillator_Check_Equations
                "Reporting errors in eigenvector normality in file:  ", HalfOscillator_Check_Normality
            "Reporting errors in eigenvectors orthogonality in file:  ", HalfOscillator_Check_Orthogonality
                                                    "Done."
```

The file `HalfOscillator_Check_Equations` will exhibit the following structure.

```
# Maximum absolute error obtained after substitution of (v,E) in the eigenvector equation M(E)v=0
# Filename:  HalfOscillator_Check_Equations
# Program version :Spectral_1.0
# Started in:  Mon Jul 23 19:24:28 2012
# Digits used:  15
#============================================================================
# (1) En.Level (2) Error
#============================================================================

1    3.41197032936222e-13
2    2.92147163334710e-13
..............<deleted output>..............
100  1.88649096344307e-12
```

The file heading is composed of a few Maple comment lines (having # as their first character), in which the user can find the name of the file, the program version, the starting date and time of computation, and also a brief reminder of the meaning of the subsequent columns of data. The first column shows the energy level indices; the second column shows the error obtained upon substituting the corresponding eigenvalue/eigenvector in the equation. (Please refer to section 3 of the paper.) Columns are separated by the ASCII TAB character.

The files `HalfOscillator_Check_Normality` has a similar layout.

```
# Checking deviation from normality of eigenvectors (dot product).
# Filename:  HalfOscillator_Check_Normality
# Program version :Spectral_1.0
# Started in:  Mon Jul 23 19:24:29 2012
# Digits used:  15
#============================================================================
# (1) En.  Level (2) Error
#============================================================================

1    0
2    1e-15
..............<deleted output>..............
100  1e-15
```

At last, the file `HalfOscillator_Check_Orthogonality` shows the following layout.

```
# Checking deviation from orthogonality of eigenvectors (dot product).
# Filename:  HalfOscillator_Check_Orthogonality
# Program version :Spectral_1.0
# Started in:  Mon Jul 23 19:24:31 2012
# Digits used:  15
#============================================================================
# (1) En.Level i, (2)En.Level j, (3) Error
#============================================================================

1       2    -6.95076372144661e-15
1       3    5.87617925506016e-16
..............<deleted output>..............
1       99   1.92408206983684e-16
1       100  1.3115711060297e-17
2       3    -9.98643591392239e-15
2       4    6.03226683672062e-15
..............<deleted output>..............
98      100  7.83159763e-15
99      100  2.94231e-18
```

The first and second columns shows the energy level indices (i, j), whereas the third column shows the dot product (9) of the i-th and j-th eigenvectors.

17. Graphically displaying error checkings.

Command.

`CheckSolutions:-Graphically`

Calling sequence.

```
p:-CheckSolutions:-Graphically()
```

Here, `p` is a problem-module (see section 2 of this manual).

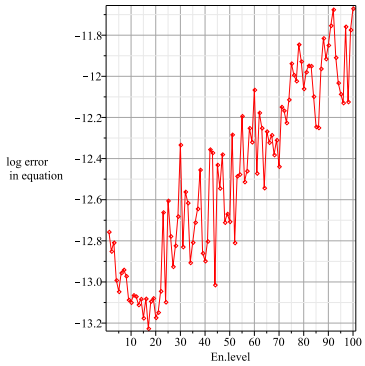
Description.
The procedure `CheckSolutions:-Graphically` reads the files created by `CheckSolutions:-Numerically` (see section 16 of this manual) and presents errors in graphical form.

Examples.
Let us explore the output corresponding to the following input line.

```
> p:-CheckSolutions:-Graphically();
```

The first group of data to be displayed is that of the error obtained when substituting eigenvalues and eigenvectors back into the matricial eigenvalue equation, in the same precision employed in calculations, as described in section 3 of the paper. The errors are displayed in logarithmic scale.

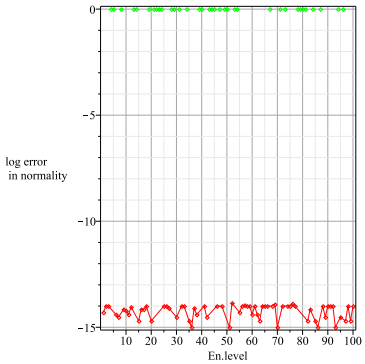
```
Checking the energy spectrum and eigenvectors using , 15, Digits.
Maximum absolute error obtained after substitution of (v,E) in the
eigenvector equation M(E)v=0
(Green dots indicate zero deviation.)
Reading data from file , HalfOscillator_Check_Equations
```



In the example shown, the error ranges from 10^{-14} to 10^{-11}, approximately. "Zero" errors (within the number of digits used) cannot be represented ordinarily in logarithmic scale; notwithstanding, they would be shown on the horizontal axis as green dots.

The next checking is related to the normality of eigenvectors, the weight function having been taken into account.

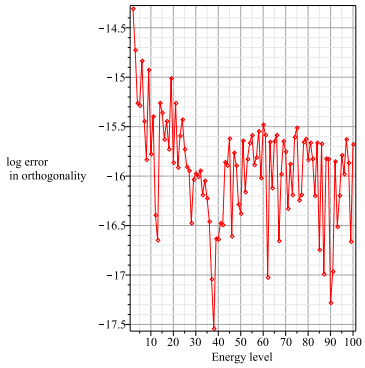
```
Checking absolute deviation from normality of eigenvectors (dot product).
(Green dots indicate zero deviation.)
Reading data from file , HalfOscillator_Check_Normality
```



Deviation from normality ranges from 10^{-15} to 10^{-14}, approximately.

The last checking is that of the orthogonality of eigenvectors. The i-th eigenvector ($i \in \{1, 2, 3, \ldots\}$) is compared to the j-th eigenvector ($j \in \{i+1, i+2, i+3, \ldots\}$). Here, we show the lowest energy eigenvector compared to the remaining ones.

```
Checking absolute deviation from orthogonality of eigenvectors (dot product).
(Green dots indicate zero deviation.)
Reading data from file , HalfOscillator_Check_Orthogonality
Comparing eigenstate at level , 1, to upper levels.
```



Deviation from orthogonality ranges from 10^{-18} to 10^{-14}, approximately.

18. Comparison of solutions

Command.

`IsComparedTo:-Numerically`

`IsComparedTo:-Graphically`

Calling sequences.

`p:-IsComparedTo:-Numerically(s)`

`s:-IsComparedTo:-Numerically(p)`

`p:-IsComparedTo:-Graphically(s)`

`s:-IsComparedTo:-Graphically(p)`

Here `p` and `q` are the problem-modules under comparison.

Parameters.

– `p` and `s` are the problem-modules under comparison (see section 2 of this manual).

Description.
The procedure `IsComparedTo:-Numerically` compares the eigenvalues of two (solved) problem-modules, saving results in a file the name of which is the concatenation of the root file name of the caller problem (defined by the option `Label` in `Problem`; see section 2 of this manual) and the suffix `_Comparison_Eigenvalues`.

Examples.
Let us compare the spectrum of the problem defined in section 2 of this manual, assigned to the variable `p`, to that of a slightly different version of it, to be assigned to the variable `s`.

```
> s := Problem(Ffunction = x^2, Gfunction=1, Variable = x, Length=10,
NumberOfLevels = 200, Weigth=1):
```

The problems `p` and `s` differ by the number of energy levels to be used, now raised to 200. Let us solve `s`.

```
> s:-IsSolvedByOctave();
```

```
                                        Time elapsed:  , 27.176, seconds
The eigenvalue/eigenvector problem has been completely solved with 15 digits used.
```

Let us call `p` the *primary* problem, and `s` the *secondary* problem; we will compare their spectra by invoking the exported procedure `IsComparedTo:-Numerically` of the primary problem, passing the secondary problem as a parameter, so that comparison data be saved in a file.

```
> p:-IsComparedTo:-Numerically(s);
```

```
                        Comparison of spectra has been written in the file ,
                              "HalfOscillator_Comparison_Eigenvalues"
                                              Done
```

The output file `HalfOscillator_Comparison_Eigenvalues` has the following layout.

```
# Filename:  HalfOscillator_Comparison_Eigenvalues
# Comparison of the eigenvalues of two different problems.
# Program version :Spectral_1.0
# Start:  Mon Jul 23 19:28:40 2012
#=============================================================================
# Description of the primary problem
# Number of energy levels:  100
# Interval amplitude (as entered):  10
# Interval amplitude (as float):  10.
# f function:  x^2
# g function:  1
# w (weight) function:  1
# The energy eigenvalues and eigenvectors have been calculated with 15 digits.
#=============================================================================
# Description of the secondary problem
# Number of energy levels:  200
# Interval amplitude (as entered):  10
# Interval amplitude (as float):  10.
# f function:  x^2
# g function:  1
# w (weight) function:  1
# The energy eigenvalues and eigenvectors have been calculated with 15 digits.
#=============================================================================
# Primary problem has 100 eigenvalues, calculated with 15 Digits
# Secondary problem has 200 eigenvalues, calculated with 15 Digits
#=============================================================================
# Absolute and percent variations of the lowest 100 energy levels:
# Columns:  (1) level, (2) eigenvalue in primary problem, (3) eigenvalue in secondary problem,
# (4) absolute variation (secondary - primary), (5) percent variation
#=============================================================================
1    3.00000000000036    3.00000000000206    1.70e-12          5.66666666666599e-11
2    6.99999999999981    7.00000000000106    1.25e-12          1.78571428571433e-11
.............<deleted output>.............
100  1039.6515762369     1020.52062690548    -19.13094933142    -1.84013084466875
```

The file heading (with Maple comment lines, begun by the character `#`) contains information on the filename, program version and date of start, as well as the description of the problems under comparison. Also, a brief description of the columns of data that follow. Each line of data is associated to an energy level. Notice that only the 100 lowest levels (shared by the two problems) are compared. The 1st column of data indexes the levels. The 2nd and 3rd columns show the eigenvalues of the primary and secondary problems, respectively. The 4th column is obtained by subtracting the 2nd from the 3rd, thus corresponding to the variation of the eigenvalue from the primary to the secondary problem. The 5th column shows the percent variation of the eigenvalue, i.e., the value in the 4th column divided by the value in the 2nd column, multiplied by 100.

Alternatively, we could switch the roles of the primary and secondary problems.

```
> s:-IsComparedTo:-Numerically(p);
```

There would be obvious changes in the name and heading of the output file, the reversal of its 2nd and 3rd columns, as well as the change of sign in its 4th. The 5th column might change altogether, due to the new denominator in the 2nd column.

Comparison data may also be presented in graphical form. No output files are generated.
`p:-IsComparedTo:-Graphically(q);`

The first part of the ouput is a description of the problems under comparison.

```
Description of the primary problem
```

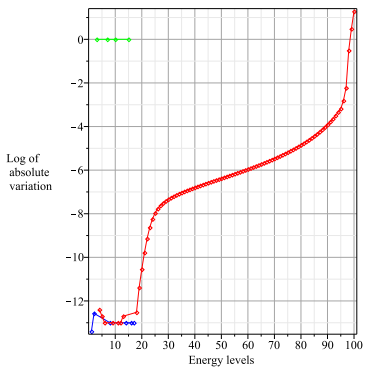
```
Number of energy levels:, 100
Interval amplitude:  , 10, (float:  , 10., .  )
f function:, x^2
g function:, 1
w (weight) function, 1
The energy eigenvalues and eigenvectors have been calculated with , 15, Digits.
```

```
Description of the secondary problem
```

```
Number of energy levels:, 200
Interval amplitude:  , 10, (float:  , 10., .  )
f function:, x^2
g function:, 1
w (weight) function, 1
The energy eigenvalues and eigenvectors have been calculated with , 15, Digits.
```

The second part describes the variation of eigenvalues in logarithmic scale. Positive variations are colored blue, negative variations are colored red; null variations (within the numeric precision indicated) are shown as green dots.

```
Log10 of absolute variations (secondary - primary) of the lowest , 100, energy levels.
```

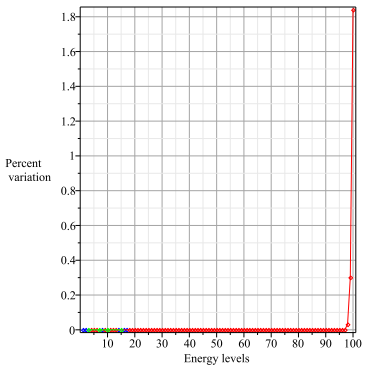



```
(Variations color code:  blue=positive, red=negative, green=null.)
```

The variation in energy levels ranges from 10^{-14} to 10^{1}, approximately.

The third and final part shows the percent variation of eigenvalues, in normal scale. The same color coding is used.

```
‘Percent variations [(secondary- primary)/primary] of the lowest ‘, 100, ‘ energy levels.‘
```



(Variations color code: blue=positive, red=negative, green=null.)

The percental variation in the energy levels ranges from 0 to 1.8%, approximately.

---

[1] D.J. Griffiths. *Introduction to Quantum Mechanics.* New Jersey, USA: Prentice Hall, 1995.

[2] M.B. Monagan *et al. Maple 13 Advanced Programming Guide*. Waterloo, Canada: Maplesoft, 2009. Available at <http://www.maplesoft.com/products/maple/history/ documentation.aspx> (accessed on 07/24/2012).

[3] GNU Octave. Available at <http://www.gnu.org/software/octave/> (accessed on 07/24/2012). GNU Octave, as part of the GNU Project, it is free software under the terms of the GNU General Public License (GPL).