QHD-2 Tunneling Escape from a Metastable State

by José Luis Gómez-Muñoz
http://homepage.cem.itesm.mx/lgomez/quantum/
jose.luis.gomez@itesm.mx

Introduction

Second Order Quantized Hamilton Dynamics (QHD-2) is applied to the **tunneling escape** from a cubic potential. QHD-2 gives an approximation to the Heisenberg Equations of Motion (EOM). The QHD commands used in this document are included in QUANTUM, which is a free *Mathematica* add-on that can be downloaded from
http://homepage.cem.itesm.mx/lgomez/quantum/
This tutorial shows how to use the QUANTUM *Mathematica* add-on to reproduce the results and graphs from Prezhdo and Pereverzev in J. Chem. Phys. Vol 113 No. 16, October 2000, Pages 6557-6565
http://homepage.cem.itesm.mx/lgomez/quantum/QHD2000.pdf
.

Load the Package

First load the Quantum`QHD` package. Write:
Needs["Quantum`QHD`"]
then press at the same time the keys SHIFT-ENTER to evaluate. *Mathematica* will load the package and print a welcome message:

```
Needs["Quantum`QHD`"]
```

```
Quantum`QHD`
A Mathematica package for Quantized Hamilton
  Dynamics approximation to Heisenberg Equations of Motion
by José Luis Gómez-Muñoz
based on the original idea of Kirill Igumenshchev

This add-on does NOT work properly with the debugger turned on. Therefore
  the debugger must NOT be checked in the Evaluation menu of Mathematica.

Execute SetQHDAliases[] in order to use the keyboard to enter QHD objects
SetQHDAliases[] must be executed again in each new notebook that is created
```

In order to use the keyboard to enter quantum objects write:
SetQHDAliases[ ];
then press at the same time the keys SHIFT-ENTER to evaluate. Remember that SetQuantumAliases[ ] must be evaluated again in each

new notebook:

```
SetQHDAliases[]
```

```
ALIASES:
[ESC]on[ESC]      · Quantum concatenation symbol
[ESC]time[ESC]    t Time symbol
[ESC]hb[ESC]      ℏ Reduced Planck's constant (h bar)
[ESC]ii[ESC]      ⅈ Imaginary I symbol
[ESC]inf[ESC]     ∞ Infinity symbol
[ESC]->[ESC]      → Option (Rule) symbol
[ESC]ave[ESC]     ⟨□⟩ Quantum average template
[ESC]expec[ESC]   ⟨□⟩ Quantum average template
[ESC]symm[ESC]    (□·□)s Symmetrized quantum product template
[ESC]comm[ESC]    ⟦□,□⟧- Commutator template
[ESC]po[ESC]      (□)^□ Power template
[ESC]su[ESC]      □_□ Subscripted variable template
[ESC]posu[ESC]    □_□^□ Power of a subscripted variable template
                  □
[ESC]fra[ESC]     — Fraction template
                  □
[ESC]eva[ESC]     □/.{□→□,□→□} Evaluation (ReplaceAll) template

SetQHDAliases[] must be executed again in each
  new notebook that is created, only one time per notebook.
```

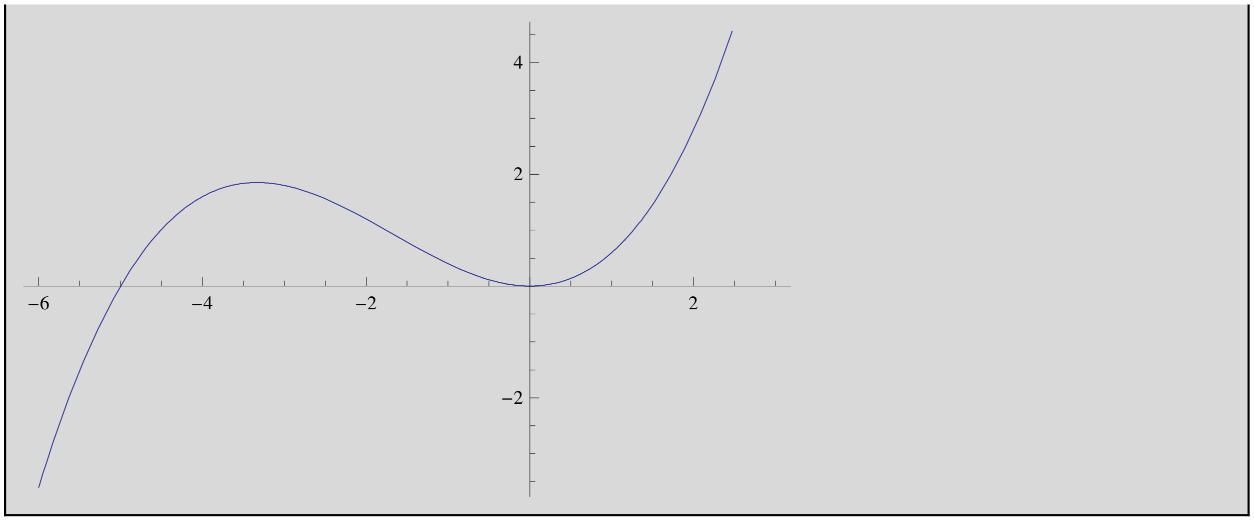
Potential Energy

The potential energy that will be used in this document is

$V = \frac{q^2}{2} + a\, q^3, \quad \text{with} \quad a = 0.1$

It is used by Prezhdo and Pereverzev in J. Chem. Phys. Vol 113 No. 16, October 2000, Pages 6557-6565
 http://homepage.cem.itesm.mx/lgomez/quantum/QHD2000.pdf
This potential energy is used to study tunneling escape with QHD, see the graph below:

```
Plot[q^2/2 + 0.1 * q^3, {q, -6, 3}]
```



Commutation Relationships and Hamiltonian

Here we define Hamiltoninan (and store it in the variable *th*) and commutation relationships that will be used in this document. In order to enter the templates and symbols $\frac{\square}{\square}$, ⟦□, □⟧₋ and *i* you can either use the QHD palette (toolbar) or press the keys [ESC]fra[ESC], [ESC]comm[ESC] and [ESC]ii[ESC]

```
Clear[a, p, q];
SetQuantumObject[q, p];
⟦q, p⟧₋ = ⅈ;
th = p^2/2 + q^2/2 + a * q^3
```

$$\frac{p^2}{2} + \frac{q^2}{2} + a\,q^3$$

Heisenberg Equations of Motion (EOM)

The evolution of the expected value of an observable *A* in the Heisenberg representation is given by the equation of motion (EOM):

$$i\hbar \frac{d}{dt} \langle A \rangle = \langle [A, H] \rangle$$

The Heisenberg EOM can be calculated using the QHD-*Mathematica* command QHDEOM, as shown below. The first argument specifies the QHD-order, which is set below to ∞ so that no QHD approximation is made in this first example. The second argument is the observable, which is set to *p* in the example below. The third argument is the Hamiltonian, remember that it was stored in variable *th* above in this document. Finally options can be included, for example QHDHBar→1 specifies that reduced Planck's constant (Dirac constant) will take the value of 1 (The → symbol can be entered by pressing the keys [ESC]->[ESC], that is [ESC][MINUS][GREATERTHAN][ESC]). The output of QHDEOM can be stored in a variable; it is stored in *teom* below:

```
teom = QHDEOM[∞, p, th, QHDHBar → 1]
```

```
{{QHDLabel, No closure was applied}, {⟨p⟩, -⟨q⟩ - 3 a ⟨q^2⟩}}
```

The output of QHDEOM can be shown in a nicer format using QHDForm. The output *teom* that was stored above is formated that way below:

```
QHDForm[teom]
```

```
No closure was applied
```

$$\frac{d\langle p\rangle}{dt} = -\langle q\rangle - 3\,a\,\langle q^2\rangle$$

On the other hand, the command QHDDifferentialEquations formats the output of QHDEOM as a standard *Mathematica* equation, as those that can be part of the input of standard *Mathematica* commands like DSolve and NDSolve. The output *teom* that was stored above is formated that way below:

```
QHDDifferentialEquations[teom]
```

```
{⟨p⟩'[t] == -⟨q⟩[t] - 3 a ⟨q^2⟩[t]}
```

A different symbol for time can be specified. The operator → can be entered pressing the keys [ESC][MINUS][GREATERTHAN][ESC], see the equations below:

```
QHDDifferentialEquations[teom, QHDSymbolForTime → z]
```

```
{⟨p⟩'[z] == -⟨q⟩[z] - 3 a ⟨q^2⟩[z]}
```

The EOM for $\langle p\rangle$ depends on $\langle q\rangle$ and $\langle q^2\rangle$ (please remember that $\langle A^2\rangle$ is NOT the same as $\langle A\rangle^2$). Next, the EOMs of $\langle q\rangle$ and $\langle q^2\rangle$ can generate new expectation values, such as $\langle p\cdot q\rangle_s = \left\langle \frac{p\cdot q+q\cdot p}{2}\right\rangle$. Regarding each expectation value as a dynamical variable, we can generate a hierarchy of EOMs, which in general is an infinite hierarchy. We can obtain some of the EOM's of that infinite hierarchy using the QHD-*Mathematica* command QHDHierarchy, which has the same syntax as the command QHDEOM that was used above. Likewise, the output of QHDHierarchy can be stored in a variable (*tinfhier* in the example below) and shown using the command QHDForm below:

```
tinfhier = QHDHierarchy[∞, p, th, QHDBar → 1];
QHDForm[ tinfhier ]
```

Calculations were stopped at order QHDMaxOrder→2

$$\frac{d\langle p\rangle}{dt} = -\langle q\rangle - 3\,a\,\langle q^2\rangle$$

$$\frac{d\langle q\rangle}{dt} = \langle p\rangle$$

$$\frac{d\langle q^2\rangle}{dt} = 2\,\langle pq\rangle_s$$

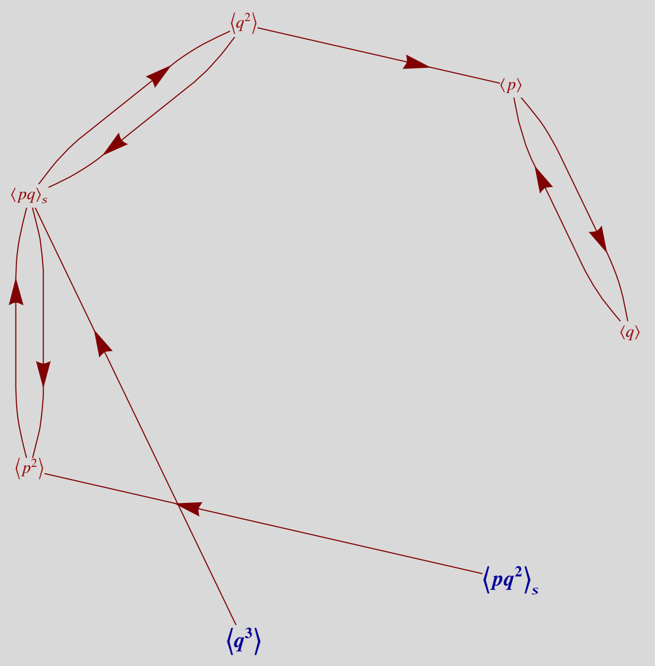
$$\frac{d\langle pq\rangle_s}{dt} = \langle p^2\rangle - \langle q^2\rangle - 3\,a\,\langle q^3\rangle$$

$$\frac{d\langle p^2\rangle}{dt} = -2\,\langle pq\rangle_s - 6\,a\,\langle pq^2\rangle_s$$

A EOM hierarchy can be shown in a graph using the command QHDGraphPlot on the output of QHDHieracthy. Arrows point from a first dynamical variable to a second dynamical variable that includes the first one in its EOM; please compare the table above with the graph below:

```
QHDGraphPlot[ tinfhier ]
```

Calculations were stopped at order QHDMaxOrder→2



QHD-2 Closure Approximation

QHDHierarchy stops calculating infinite hierarchies as specified by its option QHDMaxOrder, which takes a default value of 2. Some of the dynamical variables (expectation values) in the truncated hierarchy will not have their equation of motion (EOM) included, as it was shown above by the colors in the ouput of QHDGraphPlot. A closure procedure has to be applied in order to approximate those dynamical variables in terms of those other variables that do have their EOMs included in the hierarchy. For instance, the approximation

$\langle ABC\rangle \approx \langle AB\rangle\langle C\rangle+\langle AC\rangle\langle B\rangle+\langle BC\rangle\langle A\rangle-2\langle A\rangle\langle B\rangle\langle C\rangle$

is used to approximate the third-order dynamical variables $\langle q^3\rangle$ and $\langle pq^2\rangle_s$ in terms of the first and second order variables $\langle p\rangle$, $\langle q\rangle$, $\langle p^2\rangle$, $\langle q^2\rangle$ and $\langle p\cdot q\rangle_s = \left\langle \frac{p\cdot q+q\cdot p}{2}\right\rangle$, thus we obtain a second order QHD finite hierarchy of equations, QHD-2. The QHD-2 hierarchy is obtained as the output of the command QHDHierarchy with the first argument set to 2. Same as above, the second argument is the first variable in the hierarchy, the third argument is the hamiltonian, and optional arguments can be given after that. The output of QHDHierarchy is stored in the variable *thier*, and it is shown using the QHDForm command below:

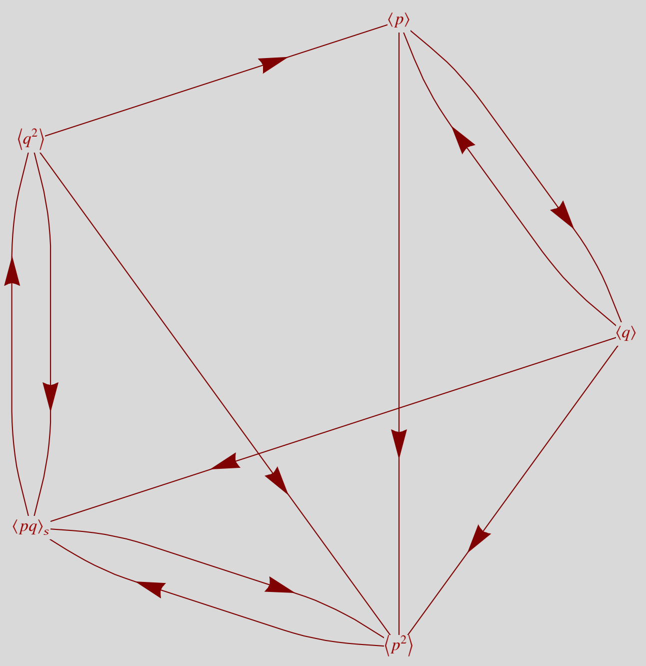
```
thier = QHDHierarchy[2, p, th, QHDHBar → 1];
QHDForm[ thier ]
```

Closure procedure was applied to order 2
$\frac{d\langle p\rangle}{dt} = -\langle q\rangle - 3\,a\,\langle q^2\rangle$
$\frac{d\langle q\rangle}{dt} = \langle p\rangle$
$\frac{d\langle q^2\rangle}{dt} = 2\,\langle pq\rangle_s$
$\frac{d\langle pq\rangle_s}{dt} = \langle p^2\rangle + 6\,a\,\langle q\rangle^3 - \langle q^2\rangle - 9\,a\,\langle q\rangle\,\langle q^2\rangle$
$\frac{d\langle p^2\rangle}{dt} = 12\,a\,\langle p\rangle\,\langle q\rangle^2 - 6\,a\,\langle p\rangle\,\langle q^2\rangle - 2\,\langle pq\rangle_s - 12\,a\,\langle q\rangle\,\langle pq\rangle_s$

The closed QHD-2 hierarchy can be shown using the QHDGraphPlot command, compare the table above and the graph below:

```
QHDGraphPlot[thier]
```

```
Closure procedure was applied to order 2
```



The command QHDDifferentialEquations formats the output of QHDHierarchy as standard *Mathematica* equations, as those that can be part of the input of standard *Mathematica* commands like DSolve and NDSolve. The output *thier* that was stored above is formated that way below:

```
QHDDifferentialEquations[thier]
```

$$\{\langle p\rangle'[t] == -\langle q\rangle[t] - 3\,a\,\langle q^2\rangle[t],\ \langle q\rangle'[t] == \langle p\rangle[t],\ \langle q^2\rangle'[t] == 2\,\langle (p\cdot q)_s\rangle[t],$$
$$\langle (p\cdot q)_s\rangle'[t] == \langle p^2\rangle[t] + 6\,a\,\langle q\rangle[t]^3 - \langle q^2\rangle[t] - 9\,a\,\langle q\rangle[t]\,\langle q^2\rangle[t],\ \langle p^2\rangle'[t] ==$$
$$12\,a\,\langle p\rangle[t]\,\langle q\rangle[t]^2 - 6\,a\,\langle p\rangle[t]\,\langle q^2\rangle[t] - 2\,\langle (p\cdot q)_s\rangle[t] - 12\,a\,\langle q\rangle[t]\,\langle (p\cdot q)_s\rangle[t]\}$$

Solution of the QHD-2 Equations: Tunneling

Next we evaluate the hierarchy when the parameter *a* takes the value of 0.1, and the result of that evaluation is stored in the variable *tnumhier*:

```
tnumhier = thier /. {a → 0.1};
QHDForm[tnumhier]
```

Closure procedure was applied to order 2
$\frac{d\langle p\rangle}{dt} = -\langle q\rangle - 0.3\,\langle q^2\rangle$
$\frac{d\langle q\rangle}{dt} = \langle p\rangle$
$\frac{d\langle q^2\rangle}{dt} = 2\,\langle pq\rangle_s$
$\frac{d\langle pq\rangle_s}{dt} = \langle p^2\rangle + 0.6\,\langle q\rangle^3 - \langle q^2\rangle - 0.9\,\langle q\rangle\,\langle q^2\rangle$
$\frac{d\langle p^2\rangle}{dt} = 1.2\,\langle p\rangle\,\langle q\rangle^2 - 0.6\,\langle p\rangle\,\langle q^2\rangle - 2\,\langle pq\rangle_s - 1.2\,\langle q\rangle\,\langle pq\rangle_s$

The command QHDInitialConditionsTemplate generates an initial conditions template for the hierarchy:

```
QHDInitialConditionsTemplate[tnumhier , 0]
```

$\{\langle p\rangle[0] == \blacksquare,\ \langle q\rangle[0] == \blacksquare,\ \langle q^2\rangle[0] == \blacksquare,\ \langle (p\cdot q)_s\rangle[0] == \blacksquare,\ \langle p^2\rangle[0] == \blacksquare\}$

Copy-paste the output of the previous command as input in the next one. Fill in the placeholders (■) with the appropiate initial values. Those initial values can be numbers. On the other hand, in the calculation below they are symbols like *po*, and then these symbols are evaluated at the desired numerical values. These initial conditions correspond to a Gaussian wavepacket, with its center one unit to the left of the minimum of the potential energy, and its momentum pointing to the right hand side, see J. Chem. Phys. Vol 113 No. 16, October 2000, Pages 6557-6565
http://homepage.cem.itesm.mx/lgomez/quantum/QHD2000.pdf
The initial conditions are stored in the variable *tinicond* below:

$$\text{tinicond} = \Big\{\langle p\rangle[0] == \text{po},\ \langle q\rangle[0] == \text{qo},\ \langle q^2\rangle[0] == \text{qo}^2 + \frac{1}{2},\ \langle (p\cdot q)_s\rangle[0] == \text{po} * \text{qo},\ \langle p^2\rangle[0] == \text{po}^2 + \frac{1}{2}\Big\} \,/.\, \{\text{po} \to 1.0,\ \text{qo} \to -1.0\}$$

$\{\langle p\rangle[0] == 1.,\ \langle q\rangle[0] == -1.,\ \langle q^2\rangle[0] == 1.5,\ \langle (p\cdot q)_s\rangle[0] == -1.,\ \langle p^2\rangle[0] == 1.5\}$

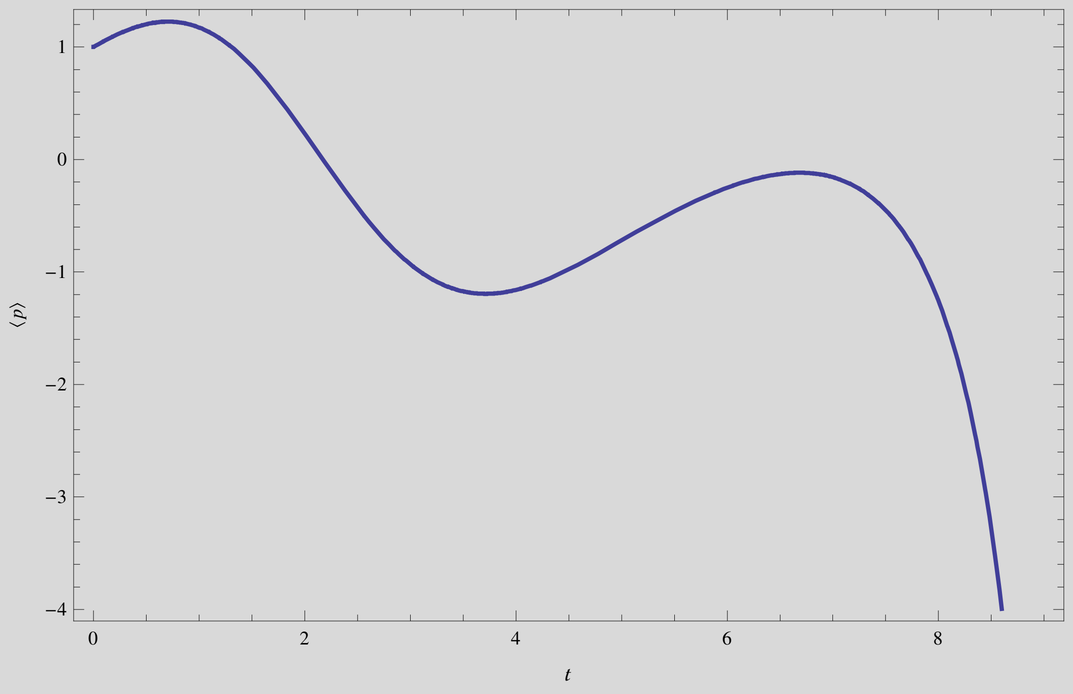
The command QHDNDSolve takes as arguments the numerical version of the hierarchy (which was stored in the variable *tnumhier*), the initial conditions (which were stored in *tinicond*), the initial time and the final time. The output of this command is the numerical solution of the differential equations, in the form of InterpolatingFunction objects. This output can be used to plot (graph) the dynamical variables as functions of time, as it will be shown below in this document. The output is stored in the variable *tsol* in the calculation below:

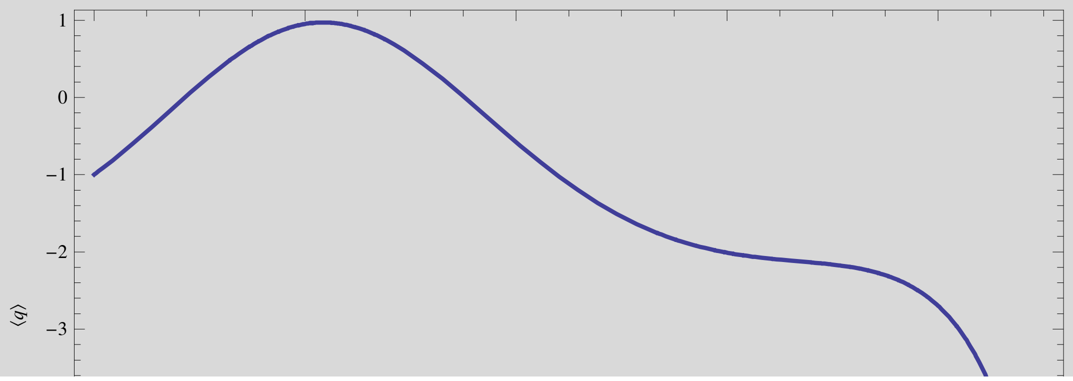
```
tsol = QHDNDSolve[tnumhier, tinicond, 0, 9]
```

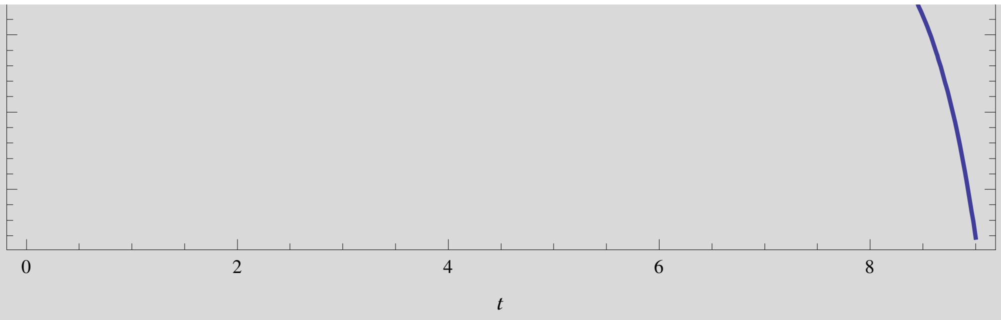
```
{{⟨p⟩ → InterpolatingFunction[{{0., 9.}}, <>],
  ⟨q⟩ → InterpolatingFunction[{{0., 9.}}, <>],
  ⟨q^2⟩ → InterpolatingFunction[{{0., 9.}}, <>],
  ⟨(p · q)_s⟩ → InterpolatingFunction[{{0., 9.}}, <>],
  ⟨p^2⟩ → InterpolatingFunction[{{0., 9.}}, <>]}}
```

The command command QHDPlot takes as its second argument the output of QHDNDSolve, which was stored in the variable *tsol*. The first argument specifies the dynamical variables or expression that we want to plot (graph) as a function of time. If we want to plot all the dynamical variables, the first argument must be the word *All*, see the five plots below:

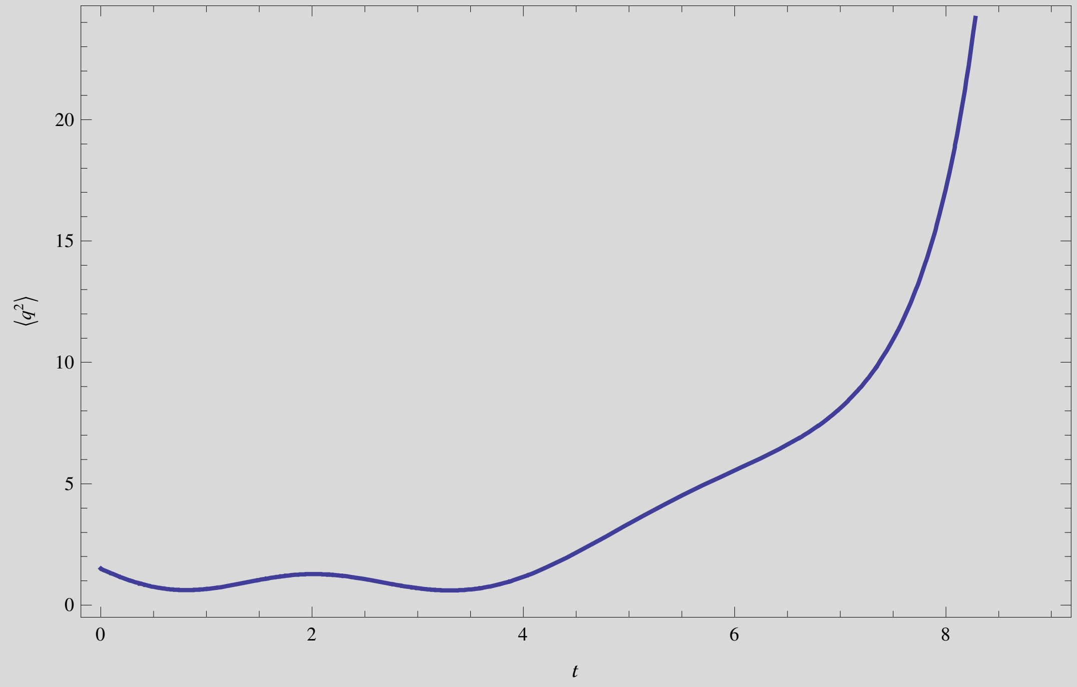
```
QHDPlot[All, tsol]
```



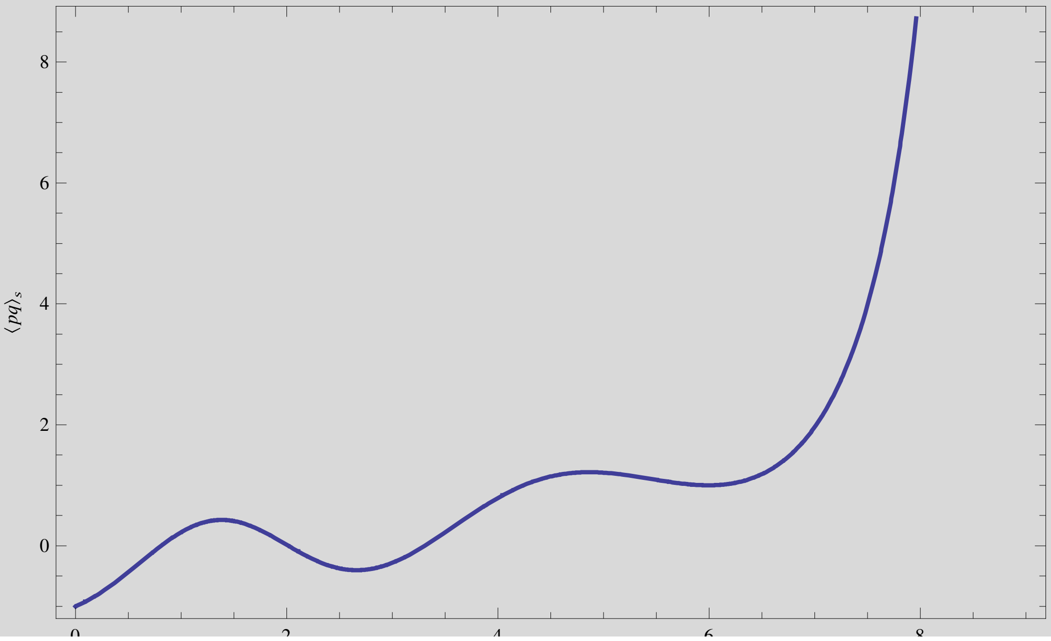




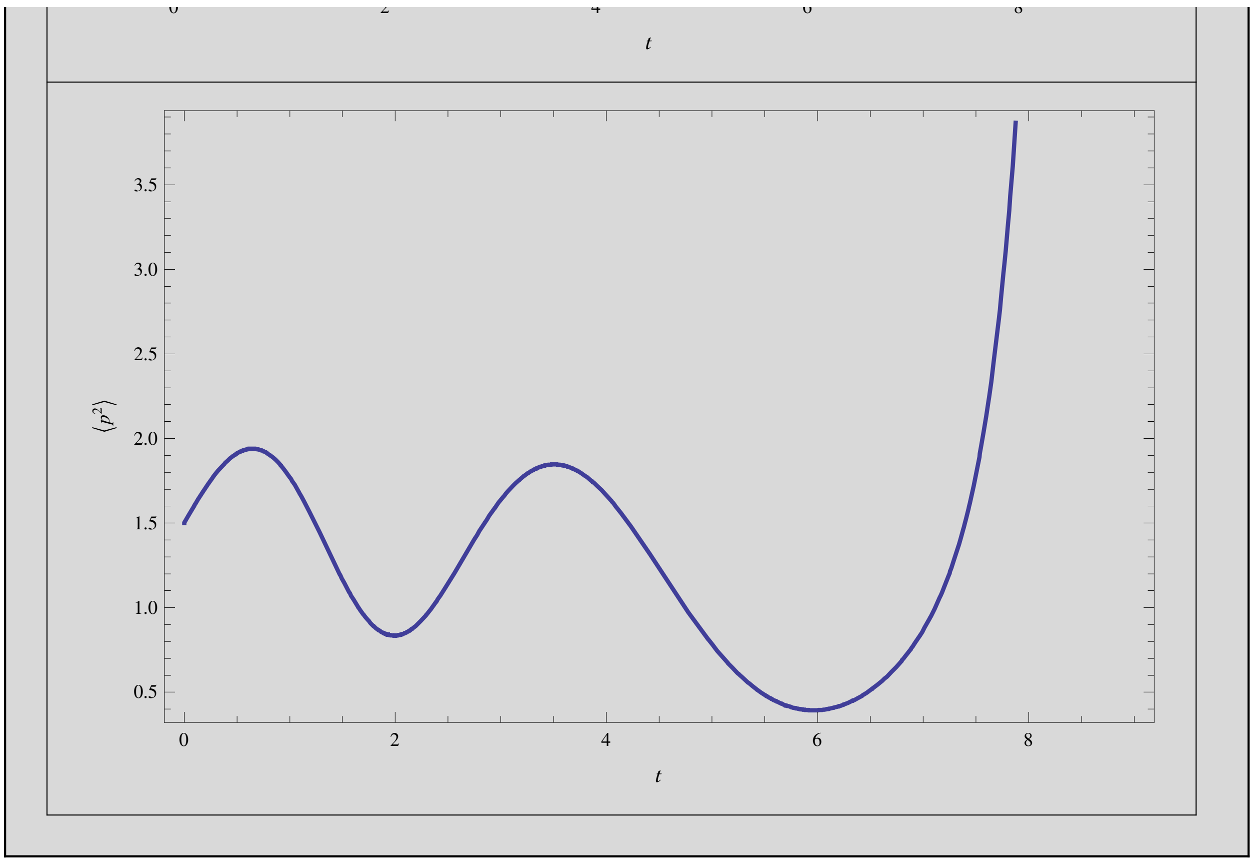
−4
−5
−6
0
2
4
6
8
t



$\langle q^2 \rangle$
0
5
10
15
20
0
2
4
6
8
t

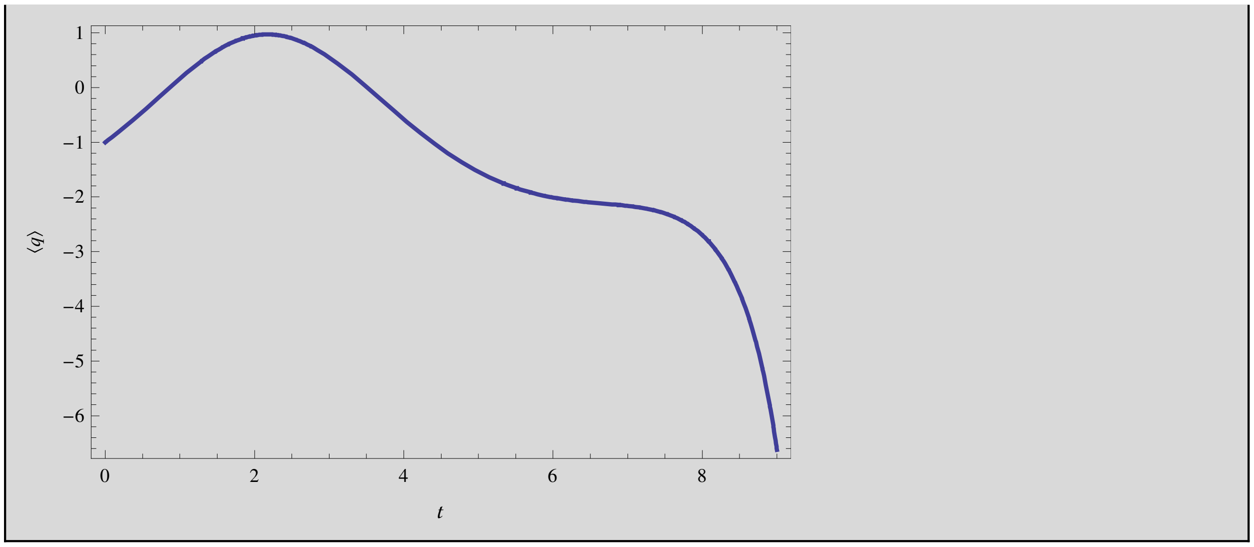


$\langle pq \rangle_s$
0
2
4
6
8



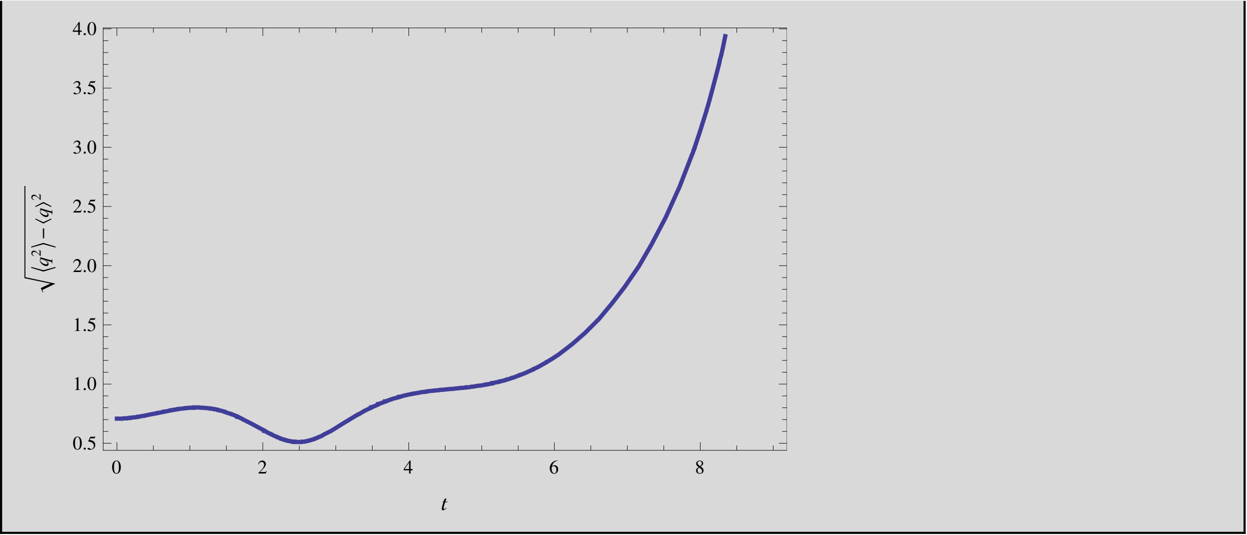
Next command generates the plot of $\langle q \rangle$ as a function of time and stores it in the variable *tpos*. This plot stored in the variable *tpos* will be used again at the end of this document.
This evolution of $\langle q \rangle$ as a function of time shows a tunneling event, because $\langle q \rangle$ leaves the basin of the local minimum of the potential. This will be clearly shown at the end of the document, where classical dynamics is compared with this QHD-2 dynamics.

```
tpos = QHDPlot[q, tsol]
```



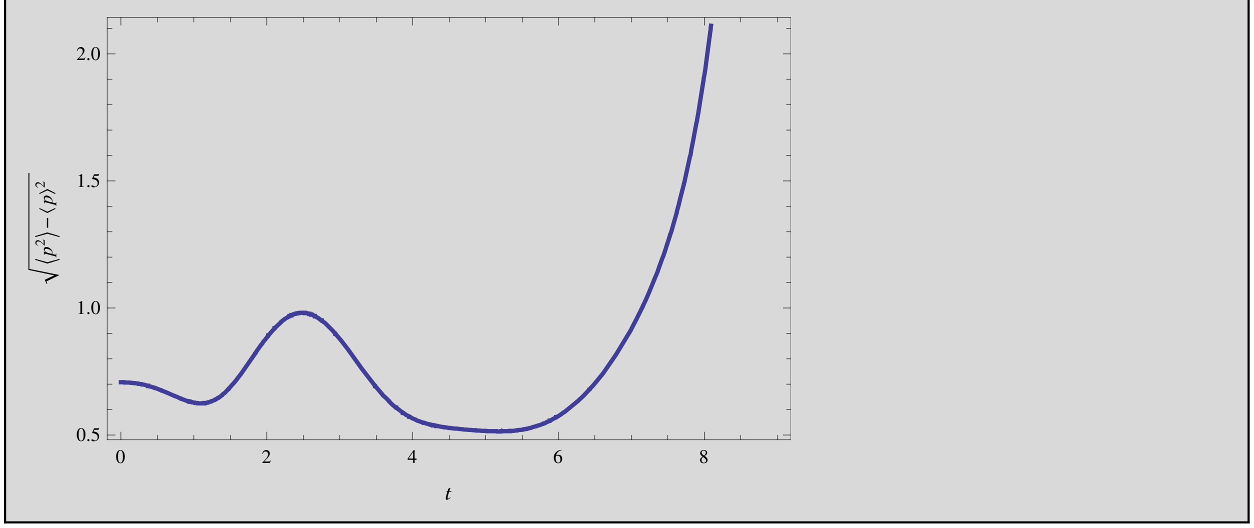
QHDPlot can be used to plot functions of the dynamical variables, for example the “spread” of the wavepacket $\sqrt{\langle q^2 \rangle - \langle q \rangle^2}$. The $\langle \square \rangle$ template can be entered using the QHD palette (toolbar), or pressing the keys [ESC]expec[ESC], or pressing the keys [ESC]ave[ESC]. The interpretation of the plot below is that the wavepacket mantains its width till the tunneling event, which roughly begins at t=4, as can be seen below:

```
QHDPlot[√(⟨q²⟩ - ⟨q⟩²), tsol]
```



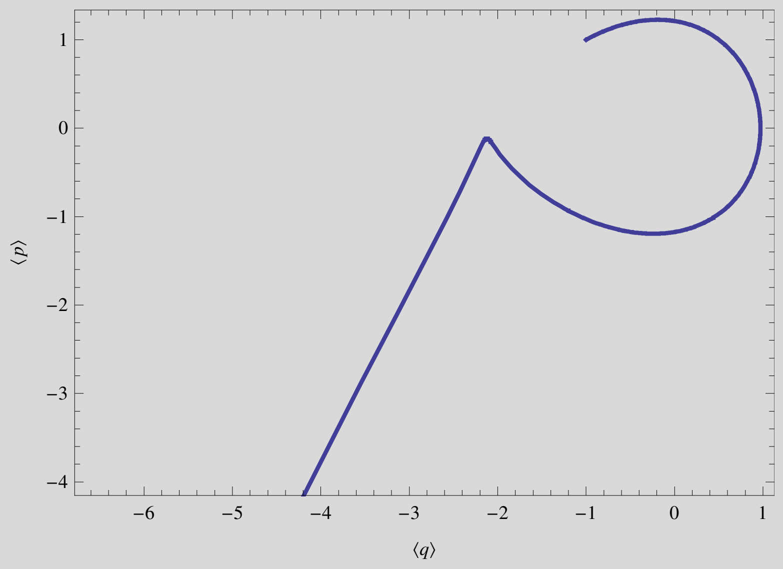
On the other hand, the “spread” in momentum $\sqrt{\langle p^2\rangle - \langle p\rangle^2}$ is very small during tunneling, roughly for 4<*t*<6, see the graph below:

```
QHDPlot[√(⟨p²⟩ - ⟨p⟩²), tsol]
```



QHDParametricPlot can be used to generate phase-space plots of couples of dynamical variables. The plot (graph) below is stored in the variable *tphase*, in order to be used again at the end of this document:

```
tphase = QHDParametricPlot[{q, p}, tsol]
```



Next the command QHDClosure is used to obtain the QHD-2 expression for the energy of the system, and it is stored in the variable *tenergy* below:

$$\texttt{tenergy = QHDClosure}\left[2,\ \frac{p^2}{2} + \frac{q^2}{2} + a * q^3\right]$$

$$\frac{\langle p^2\rangle}{2} - 2\,a\,\langle q\rangle^3 + \frac{\langle q^2\rangle}{2} + 3\,a\,\langle q\rangle\,\langle q^2\rangle$$

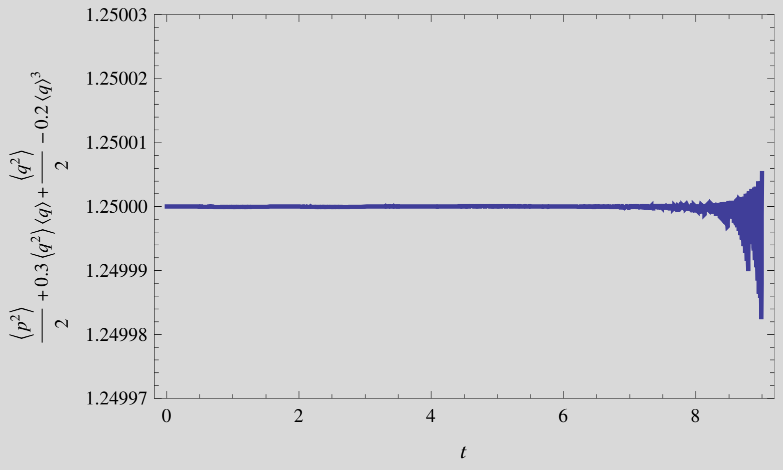
This is a numerical version of the QHD-2 expresson for the energy. It is stored in the variables *tnumenergy* below:

```
tnumenergy = tenergy /. {a → 0.1}
```

$$\frac{\langle p^2\rangle}{2} - 0.2\,\langle q\rangle^3 + \frac{\langle q^2\rangle}{2} + 0.3\,\langle q\rangle\,\langle q^2\rangle$$

QHDPlot is used to plot the QHD-2 energy. The numerical solution of the differential equations is very good at keeping the energy constant before the tunneling event. Even after the tunneling the variation of the QHD-2 energy is very small, see the scale in the left hand side of the frame below:

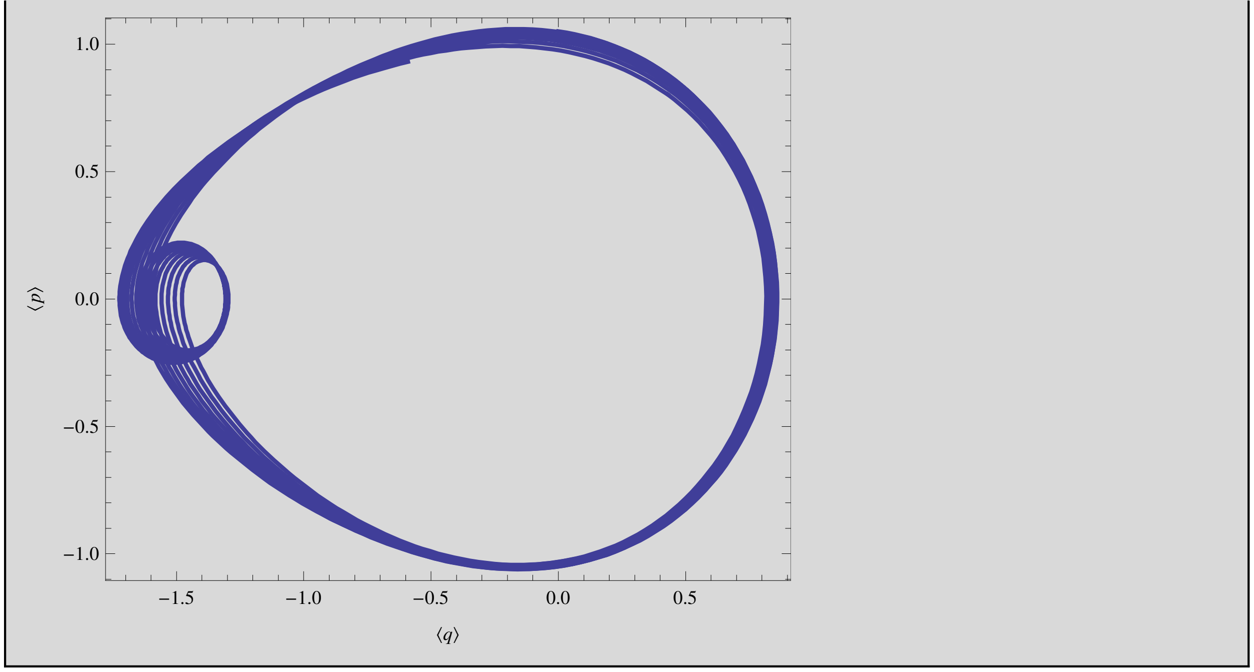
```
QHDPlot[tnumenergy, tsol, PlotRange → {1.24997, 1.25003}]
```



QHD-2 for Lower Energies

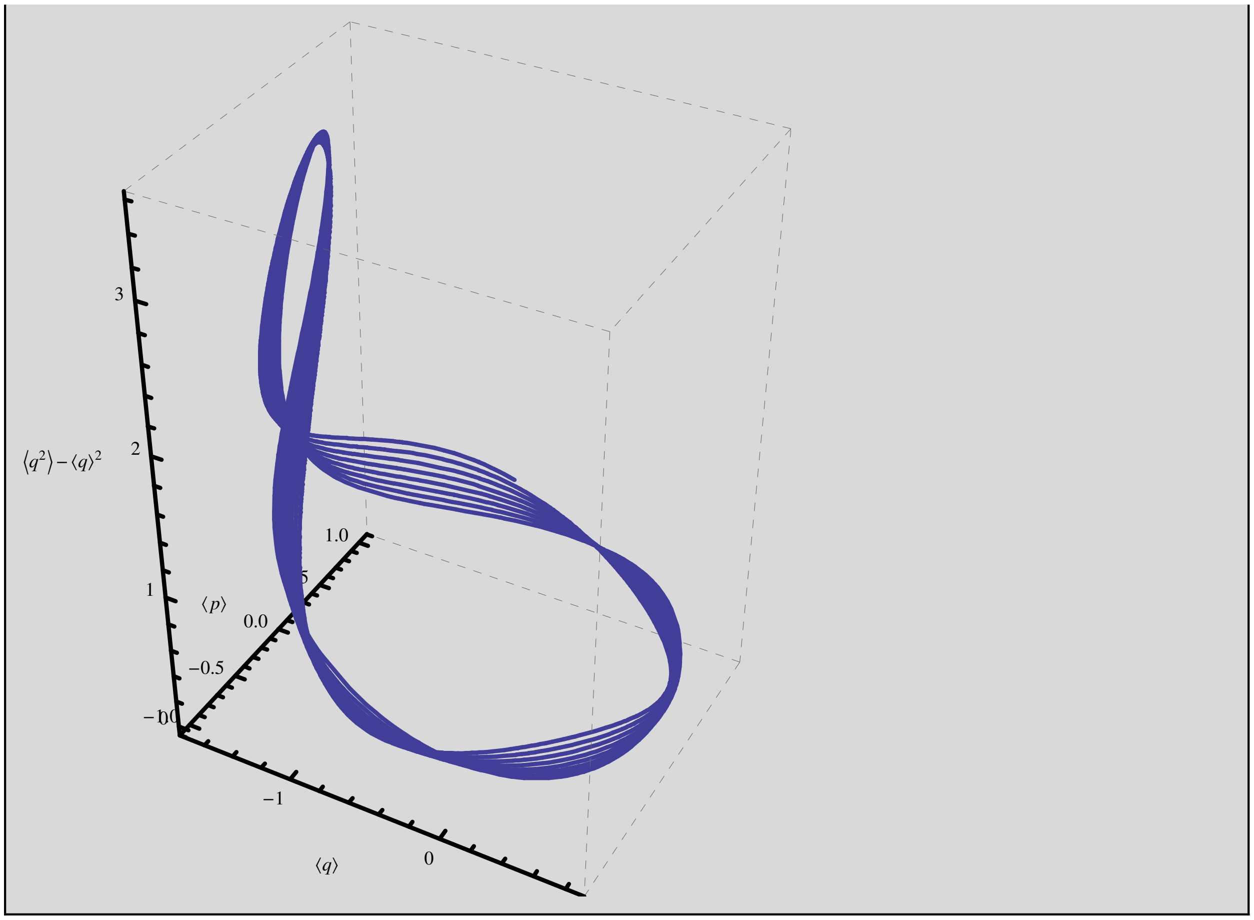
Here the numerical solution of the hierarchy (system of equations) stored in the variable *tnumhier* is obtained for a different set of initial conditions. Again they represent a Gaussian wavepacket, but this time the initial position is at the minimum of the potential energy. Notice that the solution with these initial conditions is stored in the variables *tsol2* below. The phase-space plot produced by QHDParametricPlot shows that this time there is no tunneling event, with the wavepacket only oscillating about the minimum of the potential energy, see the graph below:

```
tinicond2 = {
     ⟨p⟩[0] == po,
     ⟨q⟩[0] == qo,
     ⟨p²⟩[0] == po² + 1/2,
     ⟨(p · q)ₛ⟩[0] == po * qo,
     ⟨q²⟩[0] == qo² + 1/2} /.
   { po → 1.05, qo → 0};

tsol2 = QHDNDSolve[tnumhier, tinicond2, 0, 100];

QHDParametricPlot[{q, p}, tsol2]
```



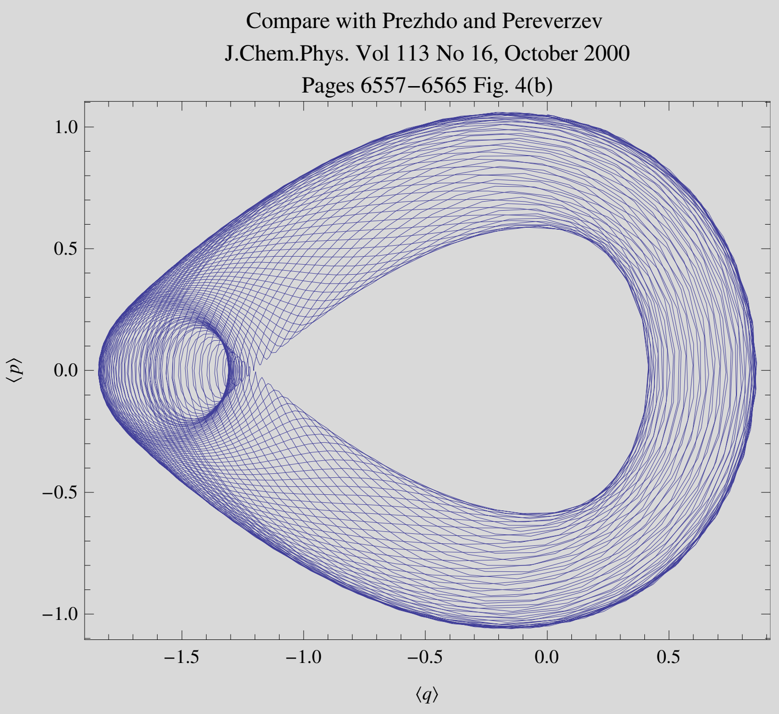
The command QHDParametricPlot**3D** can be used to generate 3D phase plots. Here the position $\langle q \rangle$, momentum $\langle p \rangle$ and "spread" $\langle q^2 \rangle - \langle q \rangle^2$ are shown for the numerical solution that was stored in *tsol2*. The $\langle \square \rangle$ template can be entered using the QHD palette (toolbar), or pressing the keys [ESC]expec[ESC], or pressing the keys [ESC]ave[ESC].

```
QHDParametricPlot3D[{q, p, ⟨q²⟩ - ⟨q⟩²}, tsol2]
```



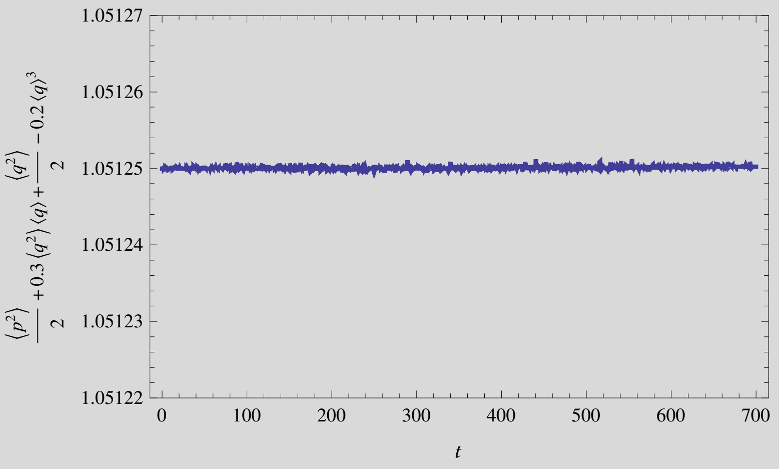
Here the hierarchy (system of equations) that was stored in *tnumhier* and the initial conditions that were stored in *tinicond2* are used to generate a graph that can be compared with the corresponding graph in the paper by Prezhdo and Pereverzev. Notice that QHDParametricPlot accepts the same options as the standard *Mathematica* command ParametricPlot, see the graph below:

```
tsol3 = QHDNDSolve[tnumhier, tinicond2, 0, 700];

QHDParametricPlot[{q, p}, tsol3, PlotStyle → Thin,
 PlotLabel → "Compare with Prezhdo and Pereverzev \nJ.Chem.Phys.
    Vol 113 No 16, October 2000\nPages 6557-6565 Fig. 4(b)"]
```



Here the expression for the QHD-2 energy, which was stored in the variable *tnumenergy* and the numerical solution that was stored in the variable *tsol3* are used to show that the numerical solution of the differential equations is very good at keeping the energy constant, see the graph below:

```
QHDPlot[tnumenergy, tsol3, PlotRange → {1.05122, 1.05127}]
```



Classical Dynamics (QHD-1) for the Same Hamiltonian

The **first-order** QHD equations are actually the same as those of classical dynamics for a single particle. Please remember that the Hamiltonian was stored in the variable *th*. The QHD-**1** hierarchy (made only of two EOMs, for position and momentum) is stored in the variable *classicthier* below:

```
classicthier = QHDHierarchy[1, p, th, QHDHBar → 1];
QHDForm[ classicthier ]
```

```
Closure procedure was applied to order 1
```
$$\frac{d\langle p\rangle}{dt} = -\langle q\rangle - 3\,a\,\langle q\rangle^2$$
$$\frac{d\langle q\rangle}{dt} = \langle p\rangle$$

Next we evaluate the hierarchy when the parameter *a* takes the value of 0.1, and the result of that evaluation is stored in the variable *numclassichier* below:

```
numclassicthier = classicthier /. {a → 0.1};
QHDForm[ numclassicthier ]
```

```
Closure procedure was applied to order 1
```
$$\frac{d\langle p\rangle}{dt} = -\langle q\rangle - 0.3\,\langle q\rangle^2$$
$$\frac{d\langle q\rangle}{dt} = \langle p\rangle$$

The command QHDInitialConditionsTemplate generates a template for the initial conditions for the hierarchy:

```
QHDInitialConditionsTemplate[numclassicthier, 0]
```

```
{⟨p⟩[0] == ■, ⟨q⟩[0] == ■}
```

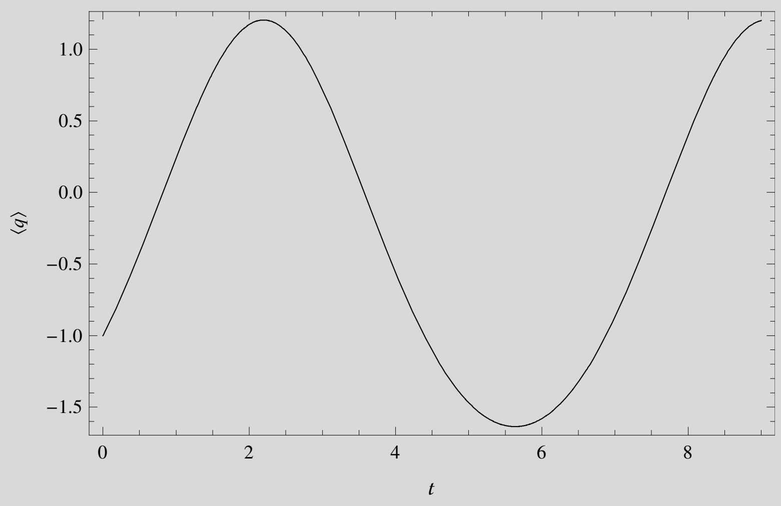
Copy-paste the output of the previous command as input in the next one. Fill in the placeholders (■) with the appropiate initial values. The initial conditions are stored in the variable *tiniclassical* and used as the second argument of the command QHDND-Solve below. The first argument is the hierarchy *numclassichier*, and the numerical solution is stored in the variable *tclassicsol* below:

```
tiniclassical = {⟨p⟩[0] == 1.0, ⟨q⟩[0] == -1.0};
tclassicsol = QHDNDSolve[numclassicthier, tiniclassical, 0, 9]
```

```
{{⟨p⟩ → InterpolatingFunction[{{0., 9.}}, <>],
  ⟨q⟩ → InterpolatingFunction[{{0., 9.}}, <>]}}
```

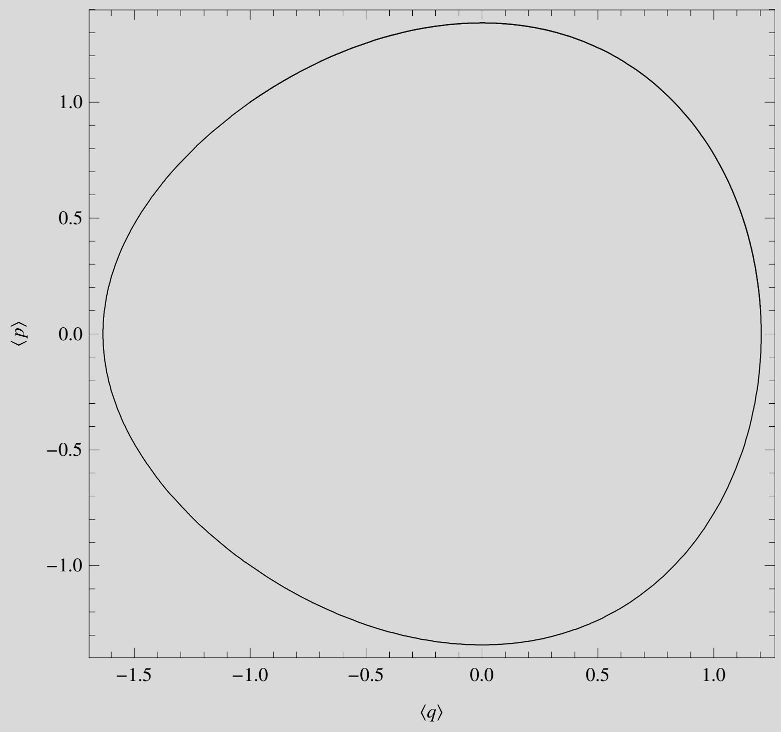
The plot of the position variable q for the classical solution *tclassicsol* is stored in the variable *tposcla*, it will be used again at the end of this document:

```
tposcla = QHDPlot[q, tclassicsol, PlotStyle → Black]
```



The phase-space plot for the classical solution *tclassicsol* is stored in the variable *tphasecla*, it will be used again at the end of this document:

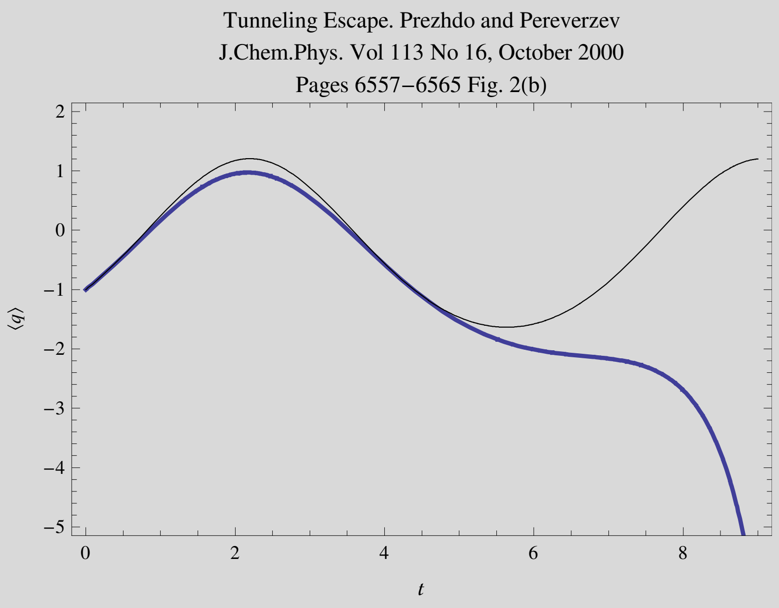
```
tphasecla = QHDParametricPlot[{q, p}, tclassicsol, PlotStyle → Black]
```



Compare Classical Dynamics (QHD-1) with QHD-2

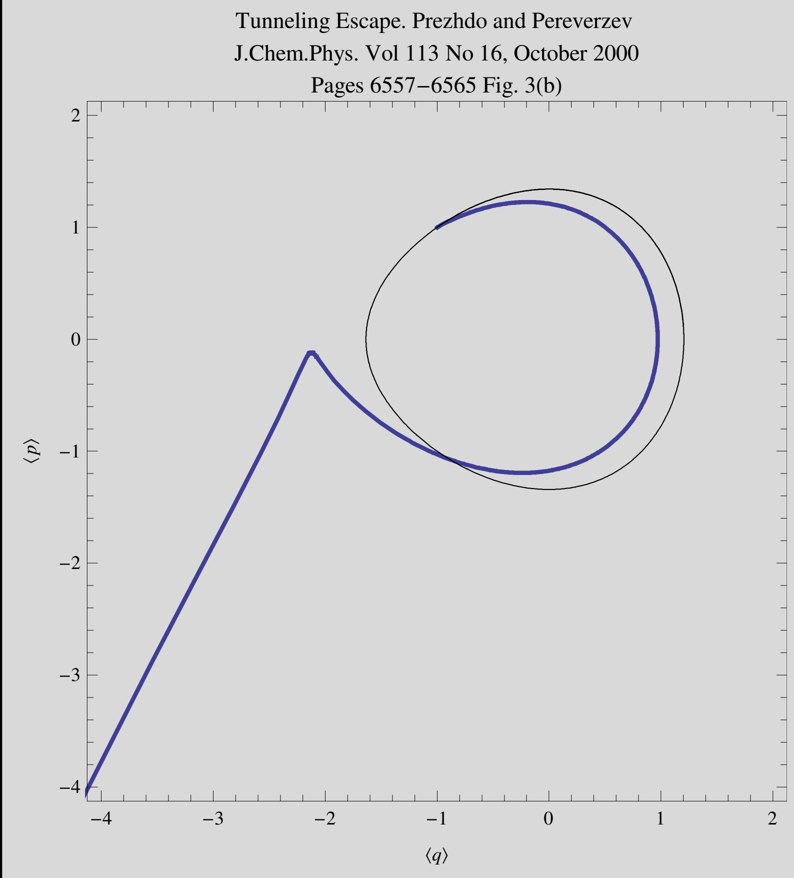
Here we compare the classical dynamics solution (*tposcla*) with the QHD-2 solution (*tpos*). They begin oscillating together, however about t=5 the QHD-2 solution shows a tunneling event, while the classical solution continues oscillating, see the graph below:

```
Show[tpos, tposcla,
 PlotRange → {-5, 2},
 PlotLabel →
  "Tunneling Escape. Prezhdo and Pereverzev\nJ.Chem.Phys. Vol 113 No
    16, October 2000\nPages 6557-6565 Fig. 2(b)"]
```



Here we compare the classical dynamics solution (*tposcla*) with the QHD-2 solution (*tpos*). They begin oscillating together, however the QHD-2 solution shows a tunneling event, while the classical solution continues oscillating, see the graph below:

```
Show[tphase, tphasecla,
 PlotRange → {{-4, 2}, {-4, 2}},
 PlotLabel →
  "Tunneling Escape. Prezhdo and Pereverzev \nJ.Chem.Phys. Vol 113
    No 16, October 2000\nPages 6557-6565 Fig. 3(b)"]
```



by José Luis Gómez-Muñoz
http://homepage.cem.itesm.mx/lgomez/quantum/
jose.luis.gomez@itesm.mx