Exact solution for quantum-classical hybrid mode of a classical harmonic oscillator quadratically coupled to a degenerate two-level quantum system

Settings

$restart;$
$with(Physics):$

Hamiltonian
$alias(H = H(q, p)):$

$alias\left(f_1 = f_1(q), f_2 = f_2(q), f_3 = f_3(q)\right):$

Wavefunction
$alias\left(\Upsilon_1 = \Upsilon_1(q, p, t), \Upsilon_2 = \Upsilon_2(q, p, t)\right):$

$\Upsilon := Vector\left(\left[\Upsilon_1, \Upsilon_2\right]\right):$

Pauli matrices
$alias\left(\sigma_1 = Matrix(Psigma[1]), \sigma_2 = Matrix(Psigma[2]), \sigma_3 = Matrix(Psigma[3])\right):$

Quanutm-classical hybrid equation of motion for the Koopman wavefunction Υ

Hybrid equation of motion

$$HybridEq := -I\cdot\hbar\cdot\frac{\partial}{\partial t}\Upsilon - I\cdot\hbar\cdot\left(\frac{\partial}{\partial p}H\cdot\frac{\partial}{\partial q}\Upsilon - \frac{\partial}{\partial q}H\cdot\frac{\partial}{\partial p}\Upsilon - \left(\frac{\partial}{\partial q}f_1\cdot\sigma_1 + \frac{\partial}{\partial q}f_2\cdot\sigma_2 + \frac{\partial}{\partial q}f_3\cdot\sigma_3\right).\frac{\partial}{\partial p}\Upsilon\right) - \frac{1}{2}\cdot\left(q\cdot\frac{\partial}{\partial q}H\cdot\Upsilon + q\cdot\left(\frac{\partial}{\partial q}f_1\cdot\sigma_1 + \frac{\partial}{\partial q}f_2\cdot\sigma_2 + \frac{\partial}{\partial q}f_3\cdot\sigma_3\right).\Upsilon + p\cdot\frac{\partial}{\partial p}H\cdot\Upsilon\right) + H\cdot\Upsilon + \left(f_1\cdot\sigma_1 + f_2\cdot\sigma_2 + f_3\cdot\sigma_3\right).\Upsilon:$$

Hybrid density matrix expressed through the Koopman wave function

Define the matrix components

$$D_{1,1} := 2\cdot|\Upsilon_1|^2 + \Re\left(q\cdot\Upsilon_1\cdot\frac{\partial}{\partial q}conjugate(\Upsilon_1) + p\cdot\Upsilon_1\cdot\frac{\partial}{\partial p}conjugate(\Upsilon_1) + 2\cdot I\cdot\hbar\cdot\frac{\partial}{\partial q}\Upsilon_1\cdot\frac{\partial}{\partial p}conjugate(\Upsilon_1)\right):$$

$$D_{1,2} := 2\cdot\Upsilon_1\cdot conjugate(\Upsilon_2) + I\cdot\hbar\cdot\left(\frac{\partial}{\partial q}\Upsilon_1\cdot\frac{\partial}{\partial p}conjugate(\Upsilon_2) - \frac{\partial}{\partial q}conjugate(\Upsilon_2)\cdot\frac{\partial}{\partial p}\Upsilon_1\right) + \frac{\Upsilon_1}{2}\cdot\left(q\cdot\frac{\partial}{\partial q}conjugate(\Upsilon_2) + p\cdot\frac{\partial}{\partial p}conjugate(\Upsilon_2)\right) + \frac{conjugate(\Upsilon_2)}{2}\cdot\left(q\cdot\frac{\partial}{\partial q}\Upsilon_1 + p\cdot\frac{\partial}{\partial p}\Upsilon_1\right):$$

$$D_{2,2} := 2\cdot|\Upsilon_2|^2 + \Re\left(q\cdot\Upsilon_2\cdot\frac{\partial}{\partial q}conjugate(\Upsilon_2) + p\cdot\Upsilon_2\cdot\frac{\partial}{\partial p}conjugate(\Upsilon_2) + 2\cdot I\cdot\hbar\cdot\frac{\partial}{\partial q}\Upsilon_2\cdot\frac{\partial}{\partial p}conjugate(\Upsilon_2)\right):$$

Hybrid density matrix

$$d := \begin{bmatrix} D_{1,1} & D_{1,2} \\ conjugate(D_{1,2}) & D_{2,2} \end{bmatrix}:$$

Classical density

$$\rho_{classical} := D_{1,1} + D_{2,2}:$$

Density matrix for the quantum subsystem

$$\rho_{quant} := map(x \rightarrow Int(x, [q = -\infty..+\infty, p = -\infty..+\infty]), d):$$

Example 1: free particle classical Boltzmann state

$$ExampleFreeParticle := \left\{\Upsilon_1 = \frac{1}{\sqrt{2}\cdot\sqrt[4]{2\cdot \mathrm{Pi}\cdot kT}}\cdot\exp\left(-\frac{p^2}{4\cdot kT} + \frac{I\cdot p\cdot q}{2\cdot\hbar}\right), \Upsilon_2 = 0\right\}$$

$$ExampleFreeParticle := \left\{Upsi_1 = \frac{2^{1/4}\,e^{-\frac{p^2}{4\,kT}+\frac{\mathrm{I}\,p\,q}{2\,\hbar}}}{2\,(\pi\,kT)^{1/4}}, Upsi_2 = 0\right\} \tag{3.1.1}$$

Get the hybrid density matrix

$simplify(subs(ExampleFreeParticle, d))$ assuming $p :: \mathbb{R}, q :: \mathbb{R}, \hbar > 0, kT > 0$

$$\begin{bmatrix} \frac{\sqrt{2}\, e^{-\frac{p^2}{2\,kT}}}{2\,\sqrt{\pi}\,\sqrt{kT}} & 0 \\ 0 & 0 \end{bmatrix} \tag{3.1.2}$$

Example 2: harmonic oscillator classical Boltzmann state

$$ExampleHarmonicOscillator := \left\{\Upsilon_1 = subs\left(H = \frac{p^2}{2\cdot m} + \frac{m\cdot\omega^2\cdot q^2}{2}, \frac{\sqrt{\frac{kT\cdot\omega}{2\cdot \mathrm{Pi}}}}{H} \cdot\sqrt{1-\left(1+\frac{H}{kT}\right)\cdot \exp\left(-\frac{H}{kT}\right)}\right), \Upsilon_2 = 0\right\}:$$

Solving the special case of a classical harmonic oscillator quadratically coupled to a degenerate two-level quantum system

$$HarmonicOscEq := subs\left(H = \frac{p^2}{2\cdot m} + \frac{m\cdot\omega^2\cdot q^2}{2}, f_1 = \frac{\beta\cdot q^2}{2}, f_2 = 0, f_3 = 0, HybridEq\right):$$

#HarmonicOscSolution:=pdsolve(HarmonicOscEq)

Analytically derived exact solution

Defining the unitary matrix U

$$\_, U := LinearAlgebra:\text{-}Eigenvectors(\sigma_1)$$

$$\_, U := \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \tag{4.1.1}$$

$$U := \frac{Dagger(U)}{\sqrt{2}}$$

$$U := \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \tag{4.1.2}$$

Check that U diagonalize the coupling

$\lambda := U.(\sigma_1).Dagger(U)$

$$\lambda := \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.1.3}$$

$\lambda := ArrayTools{:}\text{-}Diagonal(\lambda)$

$$\lambda := \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{4.1.4}$$

Introducing notations

$r_1 := \sqrt{p^2 + m \cdot \varkappa_1 \cdot q^2}$; $r_2 := \sqrt{p^2 + m \cdot \varkappa_2 \cdot q^2}$;

$$r_1 := \sqrt{m\, kappav_1\, q^2 + p^2}$$

$$r_2 := \sqrt{m\, kappav_2\, q^2 + p^2} \tag{4.1.5}$$

$\phi_1 := \arctan\left(\sqrt{m \cdot \varkappa_1} \cdot q, p\right)$; $\phi_2 := \arctan\left(\sqrt{m \cdot \varkappa_2} \cdot q, p\right)$;

$$\phi_1 := \arctan\left(\sqrt{m\, kappav_1}\, q, p\right)$$

$$\phi_2 := \arctan\left(\sqrt{m\, kappav_2}\, q, p\right) \tag{4.1.6}$$

$\varkappa_1 := m \cdot \omega^2 + \lambda[1] \cdot \beta$; $\varkappa_2 := m \cdot \omega^2 + \lambda[2] \cdot \beta$

$$kappav_1 := m\, \omega^2 - \beta$$

$$kappav_2 := m\, \omega^2 + \beta \tag{4.1.7}$$

$pq2polar1 := \left\{ q = \frac{r_1 \cdot \sin\left(\phi_1 - t \cdot \sqrt{\frac{\varkappa_1}{m}}\right)}{\sqrt{m \cdot \varkappa_1}}, p = r_1 \cdot \cos\left(\phi_1 - t \cdot \sqrt{\frac{\varkappa_1}{m}}\right) \right\}$:

$pq2polar2 := \left\{ q = \frac{r_2 \cdot \sin\left(\phi_2 - t \cdot \sqrt{\frac{\varkappa_2}{m}}\right)}{\sqrt{m \cdot \varkappa_2}}, p = r_2 \cdot \cos\left(\phi_2 - t \cdot \sqrt{\frac{\varkappa_2}{m}}\right) \right\}$:

$Y := \{$
$y_1(1) = eval\left(F_1(q,p), pq2polar1\right), y_1(2) = eval\left(F_1(q,p), pq2polar2\right),$
$y_2(1) = eval\left(F_2(q,p), pq2polar1\right), y_2(2) = eval\left(F_2(q,p), pq2polar2\right),$
$\}:$

In the following exact solution, $F_1:$ and $F_2:$ denotes for the initial condition for Υ:
$\left(subs\left(t=0, \Upsilon_1\right) = F_1(q,p), subs\left(t=0, \Upsilon_2\right) = F_2(q,p)\right):$

Here is the sought exact solution

$ExSolution := \{$
$\Upsilon_1 = eval\Big(y_1(2) + \operatorname{abs}(U[1,1])^2 \cdot \left(y_1(1) - y_1(2)\right) + conjugate(U[1,1]) \cdot U[1,2] \cdot \big(y_2(1) - y_2(2)\big), Y\Big),$
$\Upsilon_2 = eval\Big(y_2(1) + \operatorname{abs}(U[2,2])^2 \cdot \left(y_2(2) - y_2(1)\right) + conjugate(U[1,2]) \cdot U[1,1] \cdot \big(y_1(1) - y_1(2)\big), Y\Big)$
$\}:$

Verifying the initial conditions

$simplify\left(eval\left(eval\left(\Upsilon_1, ExSolution\right), t=0\right) - F_1(q,p)\right)$

$$0 \tag{4.1.8}$$

$simplify\left(eval\left(eval\left(\Upsilon_2, ExSolution\right), t=0\right) - F_2(q,p)\right)$

$$0 \tag{4.1.9}$$

Verifying the exact solution

$simplify(pdetest(ExSolution, HarmonicOscEq))$ assuming $m > 0$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.1.10}$$

Code generation for illustration

Specify the initial condition - the same as in Example 2: harmonic oscillator classical Boltzmann state

$$F_1 := (q,p) \rightarrow \frac{\sqrt{2}\,\sqrt{\frac{kT\,\omega}{\pi}}\,\sqrt{1-\left(1+\frac{\frac{p^2}{2\,m}+\frac{m\,\omega^2\,q^2}{2}}{kT}\right)\mathrm{e}^{-\frac{\frac{p^2}{2\,m}+\frac{m\,\omega^2\,q^2}{2}}{kT}}}}{2\left(\frac{p^2}{2\,m}+\frac{m\,\omega^2\,q^2}{2}\right)}:$$

$$F_2 := (q,p) \rightarrow 0:$$

$$ParamsToPlot := \left\{\hbar = 1, m = 1, \omega = 1, \beta = \frac{1}{4}, kT = \frac{1}{10000}\right\}:$$

$$SolutionToPlot := eval(ExSolution, ParamsToPlot):$$

One more (redundant) check that the obtain solution satisfies the equation

$$pdetest(SolutionToPlot, eval(HarmonicOscEq, ParamsToPlot))$$

$$\left[\begin{array}{c} 0 \\ 0 \end{array}\right] \qquad \textbf{(5.1)}$$

$$CodeGeneration{:-}Python\left(eval\left(eval\left(\mathrm{D}_{1,1}, SolutionToPlot\right), ParamsToPlot\right), optimize, resultname = \text{"D11"}\right) \text{ assuming } q :: \mathbb{R}, p :: \mathbb{R}, t :: \mathbb{R}$$

```
Warning, the function names {Re} are not recognized in the 
target language
t1 = 0.1e1 / math.pi
t2 = q ** 2
t4 = p ** 2
t5 = 0.5e1 / 0.4e1 * t2 + t4
t6 = math.sqrt(5)
t7 = math.sqrt(4)
t8 = t7 * t6
t11 = math.atan2(q * t8 / 4, p)
t15 = -t11 + t7 * t6 * t / 4
t16 = math.cos(t15)
t17 = t16 ** 2
t18 = t17 * t5
t19 = math.sin(t15)
t20 = t19 ** 2
t21 = t20 * t5
t23 = t18 + 0.4e1 / 0.5e1 * t21
t24 = 0.1e1 / t23
t25 = 5000 * t18
t26 = 4000 * t21
t27 = 1 + t25 + t26
t29 = math.exp(-t25 - t26)
t32 = math.sqrt(-t29 * t27 + 1)
t35 = 0.3e1 / 0.4e1 * t2 + t4
t36 = math.sqrt(3)
t37 = t7 * t36
t40 = math.atan2(q * t37 / 4, p)
t44 = -t40 + t7 * t36 * t / 4
t45 = math.cos(t44)
t46 = t45 ** 2
```

```
t47 = t46 * t35
t48 = 5000 * t47
t49 = math.sin(t44)
t50 = t49 ** 2
t51 = t50 * t35
t52 = 0.20000e5 / 0.3e1 * t51
t53 = 1 + t48 + t52
t55 = math.exp(-t48 - t52)
t58 = math.sqrt(-t55 * t53 + 1)
t60 = t47 + 0.4e1 / 0.3e1 * t51
t61 = 0.1e1 / t60
t64 = abs(t32 * t24 + t61 * t58)
t65 = t64 ** 2
t68 = math.sqrt(2)
t69 = math.sqrt(10000)
t70 = t69 * t68
t71 = math.sqrt(t1)
t79 = t24 * t32 * t71 * t70 / 20000 + t61 * t58 * t71 * t70 /
20000
t81 = t71 * t70
t82 = t23 ** 2
t83 = 0.1e1 / t82
t84 = (t32).conjugate
t85 = t84 * t83
t86 = t17 * q
t88 = t16 * t5
t91 = 0.1e1 / p * t7
t92 = 0.1e1 / t4
t93 = t92 * t2
t97 = t19 / (1 + 0.5e1 / 0.4e1 * t93)
t99 = t97 * t91 * t6 * t88
t101 = t20 * q
t103 = 0.5e1 / 0.2e1 * t86 + t99 / 10 + 2 * t101
t108 = 0.1e1 / t32 * t24
t112 = 12500 * t86 + 500 * t99 + 10000 * t101
t120 = (t29 * t112 * t27 - t29 * t112) * t108 * t81 / 40000
t121 = t60 ** 2
t122 = 0.1e1 / t121
t123 = (t58).conjugate
t124 = t123 * t122
t125 = t46 * q
t127 = t45 * t35
t132 = t49 / (1 + 0.3e1 / 0.4e1 * t93)
t134 = t132 * t91 * t36 * t127
t136 = t50 * q
t138 = 0.3e1 / 0.2e1 * t125 - t134 / 6 + 2 * t136
t143 = 0.1e1 / t58 * t61
t147 = 7500 * t125 - 0.2500e4 / 0.3e1 * t134 + 10000 * t136
t155 = (t55 * t147 * t53 - t55 * t147) * t143 * t81 / 40000
t159 = t17 * p
t162 = t92 * q
t164 = t97 * t162 * t8 * t88
t166 = t20 * p
t175 = 10000 * t159 - 500 * t164 + 8000 * t166
t184 = t46 * p
t188 = t132 * t162 * t37 * t127
t190 = t50 * p
t199 = 10000 * t184 + 0.2500e4 / 0.3e1 * t188 + 0.40000e5 / 0.3e1
* t190
t208 = -(2 * t159 - t164 / 10 + 0.8e1 / 0.5e1 * t166) * t85 * t81
/ 20000 + (t29 * t175 * t27 - t29 * t175) * t108 * t81 / 40000 -
(2 * t184 + t188 / 6 + 0.8e1 / 0.3e1 * t190) * t124 * t81 / 20000
```

```
+ (t55 * t199 * t53 - t55 * t199) * t143 * t81 / 40000
t222 = Re((-t103 * t85 * t81 / 20000 + t120 - t138 * t124 * t81 /
20000 + t155) * q * t79 + t208 * p * t79 + complex(0, 2) * t208 *
(t120 - t103 * t83 * t32 * t81 / 20000 + t155 - t138 * t122 * t58
* t81 / 20000))
D11 = t65 * t1 / 10000 + t222
```

CodeGeneration:-*Python*$\left(eval\left(eval\left(\mathrm{D}_{1,\,2}, SolutionToPlot\right), ParamsToPlot\right), optimize, resultname = \text{"D12"}\right)$ assuming $q :: \mathbb{R}, p :: \mathbb{R}, t :: \mathbb{R}$

```
t1 = math.sqrt(2)
t2 = math.sqrt(10000)
t3 = t2 * t1
t5 = math.sqrt(0.1e1 / math.pi)
t6 = q ** 2
t8 = p ** 2
t9 = 0.5e1 / 0.4e1 * t6 + t8
t10 = math.sqrt(5)
t11 = math.sqrt(4)
t12 = t11 * t10
t15 = math.atan2(q * t12 / 4, p)
t19 = -t15 + t11 * t10 * t / 4
t20 = math.cos(t19)
t21 = t20 ** 2
t22 = t21 * t9
t23 = 5000 * t22
t24 = math.sin(t19)
t25 = t24 ** 2
t26 = t25 * t9
t27 = 4000 * t26
t28 = 1 + t23 + t27
t30 = math.exp(-t23 - t27)
t33 = math.sqrt(-t30 * t28 + 1)
t36 = t22 + 0.4e1 / 0.5e1 * t26
t37 = 0.1e1 / t36
t39 = t37 * t33 * t5 * t3
t41 = 0.3e1 / 0.4e1 * t6 + t8
t42 = math.sqrt(3)
t43 = t11 * t42
t46 = math.atan2(q * t43 / 4, p)
t50 = -t46 + t11 * t42 * t / 4
t51 = math.cos(t50)
t52 = t51 ** 2
t53 = t52 * t41
t54 = 5000 * t53
t55 = math.sin(t50)
t56 = t55 ** 2
t57 = t56 * t41
t58 = 0.20000e5 / 0.3e1 * t57
t59 = 1 + t54 + t58
t61 = math.exp(-t54 - t58)
t64 = math.sqrt(-t61 * t59 + 1)
t67 = t53 + 0.4e1 / 0.3e1 * t57
t68 = 0.1e1 / t67
t70 = t68 * t64 * t5 * t3
t72 = t39 / 20000 + t70 / 20000
t75 = (-t70 / 20000 + t39 / 20000).conjugate
t78 = t5 * t3
t80 = 0.1e1 / t33 * t37
t81 = t21 * q
t83 = t20 * t9
```

```
t86 = 0.1e1 / p * t11
t87 = 0.1e1 / t8
t88 = t87 * t6
t92 = t24 / (1 + 0.5e1 / 0.4e1 * t88)
t94 = t92 * t86 * t10 * t83
t96 = t25 * q
t98 = 12500 * t81 + 500 * t94 + 10000 * t96
t106 = (t30 * t98 * t28 - t30 * t98) * t80 * t78 / 40000
t107 = t36 ** 2
t108 = 0.1e1 / t107
t109 = t108 * t33
t113 = 0.5e1 / 0.2e1 * t81 + t94 / 10 + 2 * t96
t118 = 0.1e1 / t64 * t68
t119 = t52 * q
t121 = t51 * t41
t126 = t55 / (1 + 0.3e1 / 0.4e1 * t88)
t128 = t126 * t86 * t42 * t121
t130 = t56 * q
t132 = 7500 * t119 - 0.2500e4 / 0.3e1 * t128 + 10000 * t130
t140 = (t61 * t132 * t59 - t61 * t132) * t118 * t78 / 40000
t141 = t67 ** 2
t142 = 0.1e1 / t141
t143 = t142 * t64
t147 = 0.3e1 / 0.2e1 * t119 - t128 / 6 + 2 * t130
t151 = t106 - t113 * t109 * t78 / 20000 + t140 - t147 * t143 *
t78 / 20000
t152 = (t64).conjugate
t153 = t152 * t142
t154 = t52 * p
t157 = t87 * q
t159 = t126 * t157 * t43 * t121
t161 = t56 * p
t163 = 2 * t154 + t159 / 6 + 0.8e1 / 0.3e1 * t161
t170 = 10000 * t154 + 0.2500e4 / 0.3e1 * t159 + 0.40000e5 / 0.3e1
* t161
t178 = (t61 * t170 * t59 - t61 * t170) * t118 * t78 / 40000
t179 = (t33).conjugate
t180 = t179 * t108
t181 = t21 * p
t185 = t92 * t157 * t12 * t83
t187 = t25 * p
t189 = 2 * t181 - t185 / 10 + 0.8e1 / 0.5e1 * t187
t196 = 10000 * t181 - 500 * t185 + 8000 * t187
t204 = (t30 * t196 * t28 - t30 * t196) * t80 * t78 / 40000
t205 = t163 * t153 * t78 / 20000 - t178 - t189 * t180 * t78 /
20000 + t204
t213 = t147 * t153 * t78 / 20000 - t140 - t113 * t180 * t78 /
20000 + t106
t220 = t204 - t189 * t109 * t78 / 20000 + t178 - t163 * t143 *
t78 / 20000
D12 = 2 * t75 * t72 + complex(0, 1) * (t205 * t151 - t220 * t213)
+ (t205 * p + t213 * q) * t72 / 2 + (t220 * p + t151 * q) * t75 /
2
```

$CodeGeneration{:}\text{-}Python\left(eval\left(eval\left(\mathrm{D}_{2,2}, SolutionToPlot\right), ParamsToPlot\right), optimize, resultname = \text{"D22"}\right) \text{ assuming } q :: \mathbb{R}, p :: \mathbb{R}, t :: \mathbb{R}$

```
Warning, the function names {Re} are not recognized in the
target language
t1 = 0.1e1 / math.pi
t2 = q ** 2
```

```
t4 = p ** 2
t5 = 0.3e1 / 0.4e1 * t2 + t4
t6 = math.sqrt(3)
t7 = math.sqrt(4)
t8 = t7 * t6
t11 = math.atan2(q * t8 / 4, p)
t15 = -t11 + t7 * t6 * t / 4
t16 = math.cos(t15)
t17 = t16 ** 2
t18 = t17 * t5
t19 = 5000 * t18
t20 = math.sin(t15)
t21 = t20 ** 2
t22 = t21 * t5
t23 = 0.20000e5 / 0.3e1 * t22
t24 = 1 + t19 + t23
t26 = math.exp(-t19 - t23)
t29 = math.sqrt(-t26 * t24 + 1)
t31 = t18 + 0.4e1 / 0.3e1 * t22
t32 = 0.1e1 / t31
t35 = 0.5e1 / 0.4e1 * t2 + t4
t36 = math.sqrt(5)
t37 = t7 * t36
t40 = math.atan2(q * t37 / 4, p)
t44 = -t40 + t7 * t36 * t / 4
t45 = math.cos(t44)
t46 = t45 ** 2
t47 = t46 * t35
t48 = math.sin(t44)
t49 = t48 ** 2
t50 = t49 * t35
t52 = t47 + 0.4e1 / 0.5e1 * t50
t53 = 0.1e1 / t52
t54 = 5000 * t47
t55 = 4000 * t50
t56 = 1 + t54 + t55
t58 = math.exp(-t54 - t55)
t61 = math.sqrt(-t58 * t56 + 1)
t64 = abs(-t32 * t29 + t61 * t53)
t65 = t64 ** 2
t68 = math.sqrt(2)
t69 = math.sqrt(10000)
t70 = t69 * t68
t71 = math.sqrt(t1)
t79 = -t32 * t29 * t71 * t70 / 20000 + t53 * t61 * t71 * t70 /
20000
t81 = t71 * t70
t82 = t31 ** 2
t83 = 0.1e1 / t82
t84 = (t29).conjugate
t85 = t84 * t83
t86 = t17 * q
t88 = t16 * t5
t91 = 0.1e1 / p * t7
t92 = 0.1e1 / t4
t93 = t92 * t2
t97 = t20 / (1 + 0.3e1 / 0.4e1 * t93)
t99 = t97 * t91 * t6 * t88
t101 = t21 * q
t103 = 0.3e1 / 0.2e1 * t86 - t99 / 6 + 2 * t101
t108 = 0.1e1 / t29 * t32
t112 = 7500 * t86 - 0.2500e4 / 0.3e1 * t99 + 10000 * t101
```

```
t120 = (t26 * t112 * t24 - t26 * t112) * t108 * t81 / 40000
t121 = t52 ** 2
t122 = 0.1e1 / t121
t123 = (t61).conjugate
t124 = t123 * t122
t125 = t46 * q
t127 = t45 * t35
t132 = t48 / (1 + 0.5e1 / 0.4e1 * t93)
t134 = t132 * t91 * t36 * t127
t136 = t49 * q
t138 = 0.5e1 / 0.2e1 * t125 + t134 / 10 + 2 * t136
t143 = 0.1e1 / t61 * t53
t147 = 12500 * t125 + 500 * t134 + 10000 * t136
t155 = (t58 * t147 * t56 - t58 * t147) * t143 * t81 / 40000
t159 = t17 * p
t162 = t92 * q
t164 = t97 * t162 * t8 * t88
t166 = t21 * p
t175 = 10000 * t159 + 0.2500e4 / 0.3e1 * t164 + 0.40000e5 / 0.3e1
* t166
t184 = t46 * p
t188 = t132 * t162 * t37 * t127
t190 = t49 * p
t199 = 10000 * t184 - 500 * t188 + 8000 * t190
t208 = (2 * t159 + t164 / 6 + 0.8e1 / 0.3e1 * t166) * t85 * t81 /
20000 - (t26 * t175 * t24 - t26 * t175) * t108 * t81 / 40000 - (2
* t184 - t188 / 10 + 0.8e1 / 0.5e1 * t190) * t124 * t81 / 20000 +
(t58 * t199 * t56 - t58 * t199) * t143 * t81 / 40000
t222 = Re((t103 * t85 * t81 / 20000 - t120 - t138 * t124 * t81 /
20000 + t155) * q * t79 + t208 * p * t79 + complex(0, 2) * t208 *
(-t120 + t103 * t83 * t29 * t81 / 20000 + t155 - t138 * t122 *
t61 * t81 / 20000))
D22 = t65 * t1 / 10000 + t222
```