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(f_1 = f_1(q), f_2 = f_2(q), f_3 = f_3(q)):


Wavefunction
alias(Υ_1 = Υ_1(q, p, t), Υ_2 = Υ_2(q, p, t)):

Υ := Vector([Υ_1, Υ_2]):

Pauli matrices
alias(σ_1 = Library:-RewriteInMatrixForm(Psigma[1]), σ_2 = Library:-
    RewriteInMatrixForm(Psigma[2]), σ_3 = Library:-RewriteInMatrixForm(Psigma[3])):
```

# 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 of Hybrid density matrix

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

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

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

Classical density

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

Density matrix for the quantum subsystem

$$\rho_{quant} := map\left(x\rightarrow Int(x,\,[q=-\infty..+\infty, p=-\infty..+\infty]), \begin{bmatrix} D_{1,1} & D_{1,2} \\ conjugate\left(D_{1,2}\right) & D_{2,2} \end{bmatrix}\right):$$

## Example 1: free particle classical Boltzmann state

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

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

Get the hybrid density matrix

$$simplify\left(eval\left(\left[\begin{array}{cc} D_{1,1} & D_{1,2} \\ conjugate\left(D_{1,2}\right) & D_{2,2} \end{array}\right], ExampleFreeParticle\right)\right) \text{ assuming } p :: \mathbb{R},\ q :: \mathbb{R},\ \hbar > 0, \beta > 0$$

$$\left[\begin{array}{cc} \frac{\sqrt{2}\,\sqrt{\beta}\,e^{-\frac{\beta p^2}{2}}}{2\sqrt{\pi}} & 0 \\ 0 & 0 \end{array}\right] \qquad \textbf{(3.1.2)}$$

## Example 2: harmonic oscillator classical Boltzmann state

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

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

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

$$\#HarmonicOscSolution := pdsolve(HarmonicOscEq)$$

# Analytically derived exact solution

Defining the unitary matrix U

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

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

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

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

Check that U diagonalize the coupling

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

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

$$\lambda := \alpha \cdot ArrayTools:-Diagonal(\lambda)$$

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

Introducing notations

$$\Omega_1 := \sqrt{\omega^2 + \frac{\lambda_1}{m}} :$$

$$\Omega_2 := \sqrt{\omega^2 + \frac{\lambda_2}{m}} :$$

$$\Omega_3 := \sqrt{\omega^2 + \frac{\lambda_1 + \lambda_2}{2 \cdot m}} :$$

Introducing new (rotated) variables

$$NewPQ := (k) \rightarrow \left\{ q = q \cdot \cos\left(\Omega_k \cdot t\right) - \frac{p \cdot \sin\left(\Omega_k \cdot t\right)}{m \cdot \Omega_k}, p = p \cdot \cos\left(\Omega_k \cdot t\right) + m \cdot \Omega_k \cdot q \cdot \sin\left(\Omega_k \cdot t\right) \right\} :$$

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

In the following exact solution, $F_1:$ and $F_2:$ denotes for the initial condition for $\Upsilon$ :

$\left(subs\left(t=0,\ \Upsilon_1\right)=F_1(q,p),\ subs\left(t=0,\ \Upsilon_2\right)=F_2(q,p)\right):$

**Here is the sought exact solution**

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

Verifying the initial conditions

$$simplify\left(eval\left(eval\left(\Upsilon_1, ExSolution\right), t=0\right) - F_1(q,p)\right)$$
$$0 \qquad \textbf{(4.1.5)}$$
$$simplify\left(eval\left(eval\left(\Upsilon_2, ExSolution\right), t=0\right) - F_2(q,p)\right)$$
$$0 \qquad \textbf{(4.1.6)}$$

Verifying the exact solution

$$simplify\left(pdetest(ExSolution, HarmonicOscEq)\right)$$
$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \textbf{(4.1.7)}$$

## Exact solution of Aleksandrov–Gerasimenko equation

Defyning equation

$$PoissonBracket := (f,g) \to \frac{\partial}{\partial q}(f) \cdot \frac{\partial}{\partial p}(g) - \frac{\partial}{\partial p}(f) \cdot \frac{\partial}{\partial q}(g):$$

$$AleksandrovGerasimenkoEq := (DM, H) \to -\frac{\partial}{\partial t}(DM) - \frac{I}{\hbar} \cdot (H \bullet DM - DM \bullet H) + \frac{1}{2} \cdot PoissonBracket(H, DM) - \frac{1}{2} \cdot PoissonBracket(DM, H):$$

Initial condition

$$InitDM := Matrix(2, 2, (k,j) \to c[k,j](q,p)):$$

$$d := simplify(U \cdot InitDM \cdot U^*):$$

$$\varphi := \frac{\lambda_1 - \lambda_2}{2\cdot\hbar\cdot(2\cdot m)^2\cdot\Omega_3^3}\cdot\left(\left(p^2 - \left(m\cdot\Omega_3\cdot q\right)^2\right)\cdot\sin\left(2\cdot\Omega_3\cdot t\right) - 2\cdot\Omega_3\cdot\left(t\cdot\left(p^2 + \left(m\cdot\Omega_3\cdot q\right)^2\right)\right.\right.$$
$$\left.\left. + m\cdot p\cdot q\cdot\left(\cos\left(2\cdot\Omega_3\cdot t\right) - 1\right)\right)\right):$$

$$DM := simplify\left(expand\left(U^* \cdot \begin{bmatrix} eval(d[1,1], NewPQ(1)) & e^{I\cdot\varphi}\cdot eval(d[1,2], NewPQ(3)) \\ e^{-I\cdot\varphi}\cdot eval(d[2,1], NewPQ(3)) & eval(d[2,2], NewPQ(2)) \end{bmatrix}\cdot U\right)\right):$$

Testing initial condition

$$eval(DM, t=0) - InitDM$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \textbf{(4.2.1)}$$

Testing the full solution

$$simplify\left(expand\left(AleksandrovGerasimenkoEq\left(DM, \frac{p^2}{2\cdot m} + \frac{m\cdot\omega^2\cdot q^2}{2} + \alpha\cdot\frac{\sigma_1\cdot q^2}{2}\right)\right)\right)$$
assuming $m > 0, \omega > 0$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \textbf{(4.2.2)}$$

# Code generation for illustration

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

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

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

$ParamsToPlot := \{\hbar = 1, m = 1, \}$ :

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

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

$pdetest(SolutionToPlot, eval(HarmonicOscEq, ParamsToPlot))$ assuming $\beta > 0, \omega > 0$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \textbf{(5.1)}$$

$CodeGeneration:-Python(eval(eval(D_{1,1}, SolutionToPlot), ParamsToPlot), optimize, resultname = "\_D11")$ assuming $p :: \mathbb{R}, q :: \mathbb{R}, \beta > 0, \omega > 0$

```
Warning, type signature [complex] for function sqrt is not
recognized
Warning, type signature [complex] for function exp is not
recognized
Warning, the function names {Im, Re} are not recognized in the
target language
t5 = math.sqrt(0.1e1 / math.pi * omega / beta)
t6 = omega ** 2
t8 = math.sqrt(t6 + alpha)
t9 = t * t8
t10 = math.cos(t9)
t13 = math.sin(t9)
t15 = t13 * q * t8 + t10 * p
t16 = t15 ** 2
t19 = 0.1e1 / t8
t21 = -t19 * t13 * p + t10 * q
t22 = t21 ** 2
t24 = t22 * t6 + t16
t25 = t24 / 2
t26 = t25 * beta
t27 = 1 + t26
t28 = math.exp(-t26)
t29 = t28 * t27
t31 = math.sqrt(1 - t29)
t33 = 0.1e1 / t24
t36 = math.sqrt(t6 - alpha)
t37 = t * t36
t38 = math.cos(t37)
t41 = math.sin(t37)
t43 = t41 * q * t36 + t38 * p
t44 = t43 ** 2
t47 = 0.1e1 / t36
t49 = -t47 * t41 * p + t38 * q
t50 = t49 ** 2
t52 = t50 * t6 + t44
t53 = t52 / 2
t54 = t53 * beta
t55 = 1 + t54
t56 = math.exp(-t54)
t57 = t56 * t55
t59 = math.sqrt(1 - t57)
t61 = 0.1e1 / t52
t64 = abs(t33 * t31 * t5 + t61 * t59 * t5)
t65 = t64 ** 2
```

```
t66 = math.sqrt(2)
t67 = t5 * t66
t73 = t33 * t31 * t67 / 2 + t61 * t59 * t67 / 2
t75 = (t24).conjugate
t76 = t75 ** 2
t78 = (t31).conjugate
t79 = t78 / t76
t80 = (t15).conjugate
t82 = (t13 * t8).conjugate
t84 = (t21).conjugate
t85 = t84 * t6
t86 = (t9).conjugate
t87 = cmath.cos(t86)
t89 = t82 * t80 + t87 * t85
t93 = (t29).conjugate
t95 = sqrt(1 - t93)
t97 = 0.1e1 / t95 / t75
t99 = (t25).conjugate
t100 = t99 * beta
t101 = exp(-t100)
t104 = beta * (1 + t100)
t112 = (t57).conjugate
t114 = sqrt(1 - t112)
t115 = 0.1e1 / t114
t116 = (t43).conjugate
t118 = (t41 * t36).conjugate
t120 = (t49).conjugate
t121 = t120 * t6
t122 = (t37).conjugate
t123 = cmath.cos(t122)
t125 = t118 * t116 + t123 * t121
t127 = (t53).conjugate
t128 = t127 * beta
t129 = exp(-t128)
t132 = beta * (1 + t128)
t137 = (t52).conjugate
t138 = 0.1e1 / t137
t141 = (t59).conjugate
t142 = t137 ** 2
t144 = 0.1e1 / t142 * t141
t155 = (t19 * t13).conjugate
t157 = -t155 * t85 + t87 * t80
t171 = (t47 * t41).conjugate
t173 = t123 * t116 - t171 * t121
t187 = (-2 * t157 * t79 + (-t101 * t157 * beta + t101 * t157 *
t104) * t97 / 2) * t67 / 2 + (t138 * (-t129 * t173 * beta + t129
* t173 * t132) * t115 / 2 - 2 * t173 * t144) * t67 / 2
t190 = Re(((-2 * t89 * t79 + (-t101 * t89 * beta + t101 * t89 *
t104) * t97 / 2) * t67 / 2 + (t138 * (-t129 * t125 * beta + t129
* t125 * t132) * t115 / 2 - 2 * t125 * t144) * t67 / 2) * q * t73
+ t187 * p * t73)
t197 = t10 * t21 * t6 + t13 * t8 * t15
t207 = t24 ** 2
t220 = t41 * t36 * t43 + t38 * t49 * t6
t230 = t52 ** 2
t239 = Im(t187 * ((t28 * t197 * beta * t27 - t28 * t197 * beta) *
t33 / t31 * t67 / 4 - t197 / t207 * t31 * t67 + (t56 * t220 *
beta * t55 - t56 * t220 * beta) * t61 / t59 * t67 / 4 - t220 /
t230 * t59 * t67))
_D11 = t65 + t190 - 2 * t239
```

$CodeGeneration:\text{-}Python\left(eval\left(eval\left(D_{1,2}, SolutionToPlot\right), ParamsToPlot\right), optimize, resultname = "\_D12"\right)$ assuming $p :: \mathbb{R}, q :: \mathbb{R}, \beta > 0, \omega > 0$

Warning, type signature [complex] for function sqrt is not recognized
Warning, type signature [complex] for function exp is not recognized

```
t1 = math.sqrt(2)
t6 = math.sqrt(0.1e1 / math.pi * omega / beta)
t7 = t6 * t1
t8 = omega ** 2
t10 = math.sqrt(t8 + alpha)
t11 = t * t10
t12 = math.cos(t11)
t15 = math.sin(t11)
t17 = t15 * q * t10 + t12 * p
t18 = t17 ** 2
t21 = 0.1e1 / t10
t23 = -t21 * t15 * p + t12 * q
t24 = t23 ** 2
t26 = t24 * t8 + t18
t27 = t26 / 2
t28 = t27 * beta
t29 = 1 + t28
t30 = math.exp(-t28)
t31 = t30 * t29
t33 = math.sqrt(1 - t31)
t34 = 0.1e1 / t26
t36 = t34 * t33 * t7
t38 = math.sqrt(t8 - alpha)
t39 = t * t38
t40 = math.cos(t39)
t43 = math.sin(t39)
t45 = t43 * q * t38 + t40 * p
t46 = t45 ** 2
t49 = 0.1e1 / t38
t51 = -t49 * t43 * p + t40 * q
t52 = t51 ** 2
t54 = t52 * t8 + t46
t55 = t54 / 2
t56 = t55 * beta
t57 = 1 + t56
t58 = math.exp(-t56)
t59 = t58 * t57
t61 = math.sqrt(1 - t59)
t62 = 0.1e1 / t54
t64 = t62 * t61 * t7
t66 = t36 / 2 + t64 / 2
t69 = (-t64 / 2 + t36 / 2).conjugate
t73 = t34 / t33
t76 = t23 * t8
t78 = t15 * t10 * t17 + t12 * t76
t81 = beta * t29
t88 = t26 ** 2
t90 = 0.1e1 / t88 * t33
t96 = t62 / t61
t99 = t51 * t8
t101 = t43 * t38 * t45 + t40 * t99
t104 = beta * t57
t111 = t54 ** 2
t113 = 0.1e1 / t111 * t61
```

```
t118 = (-t30 * t78 * beta + t30 * t78 * t81) * t73 * t7 / 4 - t78
* t90 * t7 + (-t58 * t101 * beta + t58 * t101 * t104) * t96 * t7
/ 4 - t101 * t113 * t7
t119 = (t59).conjugate
t121 = sqrt(1 - t119)
t122 = 0.1e1 / t121
t123 = (t45).conjugate
t124 = (t39).conjugate
t125 = cmath.cos(t124)
t127 = (t51).conjugate
t128 = t127 * t8
t129 = t49 * t43
t130 = (t129).conjugate
t132 = t125 * t123 - t130 * t128
t134 = (t55).conjugate
t135 = t134 * beta
t136 = exp(-t135)
t139 = beta * (1 + t135)
t144 = (t54).conjugate
t145 = 0.1e1 / t144
t148 = (t61).conjugate
t149 = t144 ** 2
t151 = 0.1e1 / t149 * t148
t156 = (t26).conjugate
t157 = t156 ** 2
t159 = (t33).conjugate
t160 = t159 / t157
t161 = (t17).conjugate
t162 = (t11).conjugate
t163 = cmath.cos(t162)
t165 = (t23).conjugate
t166 = t165 * t8
t167 = t21 * t15
t168 = (t167).conjugate
t170 = t163 * t161 - t168 * t166
t174 = (t31).conjugate
t176 = sqrt(1 - t174)
t178 = 0.1e1 / t176 / t156
t180 = (t27).conjugate
t181 = t180 * beta
t182 = exp(-t181)
t185 = beta * (1 + t181)
t194 = -(t145 * (-t136 * t132 * beta + t136 * t132 * t139) * t122
/ 2 - 2 * t132 * t151) * t7 / 2 + (-2 * t170 * t160 + (-t182 *
t170 * beta + t182 * t170 * t185) * t178 / 2) * t7 / 2
t197 = (t43 * t38).conjugate
t200 = t197 * t123 + t125 * t128
t214 = (t15 * t10).conjugate
t217 = t214 * t161 + t163 * t166
t230 = -(t145 * (-t136 * t200 * beta + t136 * t200 * t139) * t122
/ 2 - 2 * t200 * t151) * t7 / 2 + (-2 * t217 * t160 + (-t182 *
t217 * beta + t182 * t217 * t185) * t178 / 2) * t7 / 2
t233 = t12 * t17 - t167 * t76
t248 = -t129 * t99 + t40 * t45
t261 = (-t30 * t233 * beta + t30 * t233 * t81) * t73 * t7 / 4 -
t233 * t90 * t7 + (-t58 * t248 * beta + t58 * t248 * t104) * t96
* t7 / 4 - t248 * t113 * t7
_D12 = 2 * t69 * t66 + complex(0, 1) * (t194 * t118 - t261 *
t230) + (t194 * p + t230 * q) * t66 / 2 + (t261 * p + t118 * q) *
t69 / 2
```

$$\mathit{CodeGeneration}\text{:-}\mathit{Python}\left(\mathit{eval}\left(\mathit{eval}\left(\mathrm{D}_{2,\,2},\ \mathit{SolutionToPlot}\right),\ \mathit{ParamsToPlot}\right),\ \mathit{optimize},\ \mathit{resultname} = \text{"\_D22"}\right) \text{ assuming } p :: \mathbb{R},\ q :: \mathbb{R},\ \beta > 0,\ \omega > 0$$

Warning, type signature [complex] for function sqrt is not recognized
Warning, type signature [complex] for function exp is not recognized
Warning, the function names {Im, Re} are not recognized in the target language

```
t5 = math.sqrt(0.1e1 / math.pi * omega / beta)
t6 = omega ** 2
t8 = math.sqrt(t6 - alpha)
t9 = t * t8
t10 = math.cos(t9)
t13 = math.sin(t9)
t15 = t13 * q * t8 + t10 * p
t16 = t15 ** 2
t19 = 0.1e1 / t8
t21 = -t19 * t13 * p + t10 * q
t22 = t21 ** 2
t24 = t22 * t6 + t16
t25 = t24 / 2
t26 = t25 * beta
t27 = 1 + t26
t28 = math.exp(-t26)
t29 = t28 * t27
t31 = math.sqrt(1 - t29)
t33 = 0.1e1 / t24
t36 = math.sqrt(t6 + alpha)
t37 = t * t36
t38 = math.cos(t37)
t41 = math.sin(t37)
t43 = t41 * q * t36 + t38 * p
t44 = t43 ** 2
t47 = 0.1e1 / t36
t49 = -t47 * t41 * p + t38 * q
t50 = t49 ** 2
t52 = t50 * t6 + t44
t53 = t52 / 2
t54 = t53 * beta
t55 = 1 + t54
t56 = math.exp(-t54)
t57 = t56 * t55
t59 = math.sqrt(1 - t57)
t61 = 0.1e1 / t52
t64 = abs(-t33 * t31 * t5 + t61 * t59 * t5)
t65 = t64 ** 2
t66 = math.sqrt(2)
t67 = t5 * t66
t73 = -t33 * t31 * t67 / 2 + t61 * t59 * t67 / 2
t75 = (t29).conjugate
t77 = sqrt(1 - t75)
t78 = 0.1e1 / t77
t79 = (t15).conjugate
t81 = (t13 * t8).conjugate
t83 = (t21).conjugate
t84 = t83 * t6
t85 = (t9).conjugate
t86 = cmath.cos(t85)
t88 = t81 * t79 + t86 * t84
t90 = (t25).conjugate
```

```
t91 = t90 * beta
t92 = exp(-t91)
t95 = beta * (1 + t91)
t100 = (t24).conjugate
t101 = 0.1e1 / t100
t104 = (t31).conjugate
t105 = t100 ** 2
t107 = 0.1e1 / t105 * t104
t112 = (t52).conjugate
t113 = t112 ** 2
t115 = (t59).conjugate
t116 = t115 / t113
t117 = (t43).conjugate
t119 = (t41 * t36).conjugate
t121 = (t49).conjugate
t122 = t121 * t6
t123 = (t37).conjugate
t124 = cmath.cos(t123)
t126 = t119 * t117 + t124 * t122
t130 = (t57).conjugate
t132 = sqrt(1 - t130)
t134 = 0.1e1 / t132 / t112
t136 = (t53).conjugate
t137 = t136 * beta
t138 = exp(-t137)
t141 = beta * (1 + t137)
t155 = (t19 * t13).conjugate
t157 = -t155 * t84 + t86 * t79
t172 = (t47 * t41).conjugate
t174 = t124 * t117 - t172 * t122
t187 = -(t101 * (-t92 * t157 * beta + t92 * t157 * t95) * t78 / 2
- 2 * t157 * t107) * t67 / 2 + (-2 * t174 * t116 + (-t138 * t174
* beta + t138 * t174 * t141) * t134 / 2) * t67 / 2
t190 = Re((-(t101 * (-t92 * t88 * beta + t92 * t88 * t95) * t78 /
2 - 2 * t88 * t107) * t67 / 2 + (-2 * t126 * t116 + (-t138 * t126
* beta + t138 * t126 * t141) * t134 / 2) * t67 / 2) * q * t73 +
t187 * p * t73)
t197 = t10 * t21 * t6 + t13 * t8 * t15
t207 = t24 ** 2
t220 = t41 * t36 * t43 + t38 * t49 * t6
t230 = t52 ** 2
t239 = Im(t187 * (-(t28 * t197 * beta * t27 - t28 * t197 * beta)
* t33 / t31 * t67 / 4 + t197 / t207 * t31 * t67 + (t56 * t220 *
beta * t55 - t56 * t220 * beta) * t61 / t59 * t67 / 4 - t220 /
t230 * t59 * t67))
_D22 = t65 + t190 - 2 * t239
```

# Code for plotting the solution for Aleksandrov–Gerasimenko equation

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

$$c_{1,1}(q,p) := \frac{\omega\cdot\beta}{2\cdot\pi}\cdot\exp\left(-\beta\cdot\left(\frac{p^2}{2\cdot m}+\frac{m\cdot\omega^2\cdot q^2}{2}\right)\right):$$
$$c_{1,2}(q,p) := 0:$$
$$c_{2,1}(q,p) := 0:$$
$$c_{2,2}(q,p) := 0:$$

Check the normalization

$$int\left(c_{1,1}(q,p),\,[q=-\infty..+\infty,\,p=-\infty..+\infty]\right) \text{ assuming } \beta>0,\,m>0,\,\omega>0$$
$$1 \qquad \textbf{(6.1)}$$

*CodeGeneration*:-*Python*(*simplify*(*eval*(*DM*[1, 1], *ParamsToPlot*)), *optimize*, *resultname* = "_D11") assuming $p :: \mathbb{R}$, $q :: \mathbb{R}$, $\beta > 0$, $\omega > 0$

```
Warning, type signature [complex] for function exp is not
recognized
Warning, type signature [complex] for function sqrt is not
recognized
t3 = omega ** 2
t4 = q ** 2
t5 = t4 * t3
t7 = p ** 2
t9 = complex(0, -1) * t5 * alpha + complex(0, 1) * t7 * alpha
t11 = 2 * omega * t
t12 = math.sin(t11)
t14 = p * q
t15 = math.cos(t11)
t17 = t15 * alpha * t14
t18 = t3 ** 2
t20 = t4 * t18 * beta
t21 = t4 * alpha
t22 = t * t21
t23 = t7 * beta
t29 = alpha * (p * t - q) * p
t35 = 1 / t3 / omega
t38 = exp(t35 * (t12 * t9 - 2 * (complex(0, 1) * t17 + t20 + t3 *
(complex(0, 1) * t22 + t23) + complex(0, 1) * t29) * omega) / 4)
t49 = exp(t35 * (-t12 * t9 + 2 * (complex(0, 1) * t17 - t20 + t3
* (complex(0, 1) * t22 - t23) + complex(0, 1) * t29) * omega) /
4)
t52 = t3 - alpha
t53 = sqrt(t52)
t54 = t * t53
t55 = math.cos(t54)
t56 = t55 ** 2
t58 = math.sin(t54)
t64 = t4 * t18
t67 = 2 * t4 * t3 * alpha
t68 = t7 * t3
t69 = alpha ** 2
t70 = t4 * t69
t77 = exp(0.1e1 / t52 * beta * (t56 * (-t5 + t21 + t7) * alpha +
2 * t53 * t14 * alpha * t58 * t55 - t64 + t67 - t68 - t70) / 2)
t80 = t3 + alpha
t81 = sqrt(t80)
```

```
t82 = t * t81
t83 = math.cos(t82)
t84 = t83 ** 2
t86 = math.sin(t82)
t97 = exp(-0.1e1 / t80 * (t84 * (-t5 - t21 + t7) * alpha + 2 *
t81 * t14 * alpha * t86 * t83 + t64 + t67 + t68 + t70) * beta /
2)
_D11 = 0.1e1 / math.pi * (t38 + t49 + t77 + t97) * omega * beta /
8
```

*CodeGeneration*:-*Python*(*simplify*(*eval*(*DM*[1, 2], *ParamsToPlot*)), *optimize*, *resultname* = "_D12") assuming $p :: \mathbb{R}$, $q :: \mathbb{R}$, $\beta > 0$, $\omega > 0$

```
Warning, type signature [complex] for function exp is not
recognized
Warning, type signature [complex] for function sqrt is not
recognized
t3 = omega ** 2
t4 = q ** 2
t5 = t4 * t3
t7 = p ** 2
t9 = complex(0, -1) * t5 * alpha + complex(0, 1) * t7 * alpha
t11 = 2 * omega * t
t12 = math.sin(t11)
t14 = p * q
t15 = math.cos(t11)
t17 = t15 * alpha * t14
t18 = t3 ** 2
t20 = t4 * t18 * beta
t21 = t4 * alpha
t22 = t * t21
t23 = t7 * beta
t29 = alpha * (p * t - q) * p
t35 = 1 / t3 / omega
t38 = exp(t35 * (t12 * t9 - 2 * (complex(0, 1) * t17 + t20 + t3 *
(complex(0, 1) * t22 + t23) + complex(0, 1) * t29) * omega) / 4)
t49 = exp(t35 * (-t12 * t9 + 2 * (complex(0, 1) * t17 - t20 + t3
* (complex(0, 1) * t22 - t23) + complex(0, 1) * t29) * omega) /
4)
t52 = t3 - alpha
t53 = sqrt(t52)
t54 = t * t53
t55 = math.cos(t54)
t56 = t55 ** 2
t58 = math.sin(t54)
t64 = t4 * t18
t67 = 2 * t4 * t3 * alpha
t68 = t7 * t3
t69 = alpha ** 2
t70 = t4 * t69
t77 = exp(0.1e1 / t52 * beta * (t56 * (-t5 + t21 + t7) * alpha +
2 * t53 * t14 * alpha * t58 * t55 - t64 + t67 - t68 - t70) / 2)
t80 = t3 + alpha
t81 = sqrt(t80)
t82 = t * t81
t83 = math.cos(t82)
t84 = t83 ** 2
t86 = math.sin(t82)
t97 = exp(-0.1e1 / t80 * (t84 * (-t5 - t21 + t7) * alpha + 2 *
t81 * t14 * alpha * t86 * t83 + t64 + t67 + t68 + t70) * beta /
2)
```

```
 D12 = -0.1e1 / math.pi * (t38 - t49 + t77 - t97) * omega * beta
7 8
```

*CodeGeneration*:-*Python*(*simplify*(*eval*(*DM*[2, 2], *ParamsToPlot*)), *optimize*, *resultname* = "_D22") assuming $p :: \mathbb{R}$, $q :: \mathbb{R}$, $\beta > 0$, $\omega > 0$

```
Warning, type signature [complex] for function exp is not
recognized
Warning, type signature [complex] for function sqrt is not
recognized
t3 = omega ** 2
t4 = q ** 2
t5 = t4 * t3
t7 = p ** 2
t9 = complex(0, -1) * t5 * alpha + complex(0, 1) * t7 * alpha
t11 = 2 * omega * t
t12 = math.sin(t11)
t14 = p * q
t15 = math.cos(t11)
t17 = t15 * alpha * t14
t18 = t3 ** 2
t20 = t4 * t18 * beta
t21 = t4 * alpha
t22 = t * t21
t23 = t7 * beta
t29 = alpha * (p * t - q) * p
t35 = 1 / t3 / omega
t38 = exp(t35 * (t12 * t9 - 2 * (complex(0, 1) * t17 + t20 + t3 *
(complex(0, 1) * t22 + t23) + complex(0, 1) * t29) * omega) / 4)
t49 = exp(t35 * (-t12 * t9 + 2 * (complex(0, 1) * t17 - t20 + t3
* (complex(0, 1) * t22 - t23) + complex(0, 1) * t29) * omega) /
4)
t52 = t3 - alpha
t53 = sqrt(t52)
t54 = t * t53
t55 = math.cos(t54)
t56 = t55 ** 2
t58 = math.sin(t54)
t64 = t4 * t18
t67 = 2 * t4 * t3 * alpha
t68 = t7 * t3
t69 = alpha ** 2
t70 = t4 * t69
t77 = exp(0.1e1 / t52 * beta * (t56 * (-t5 + t21 + t7) * alpha +
2 * t53 * t14 * alpha * t58 * t55 - t64 + t67 - t68 - t70) / 2)
t80 = t3 + alpha
t81 = sqrt(t80)
t82 = t * t81
t83 = math.cos(t82)
t84 = t83 ** 2
t86 = math.sin(t82)
t97 = exp(-0.1e1 / t80 * (t84 * (-t5 - t21 + t7) * alpha + 2 *
t81 * t14 * alpha * t86 * t83 + t64 + t67 + t68 + t70) * beta /
2)
 D22 = -0.1e1 / math.pi * (t38 + t49 - t77 - t97) * omega * beta
7 8
```