Quantum harmonic oscillator with time-dependent frequency

Modules

The following modules refer to the algebra

$\mathcal{L}_n = \{h_1, \ldots, h_n\}$,

that fulfils the commutation rule

$[h_i, h_j] = i\,\hbar \sum_{k=1}^{n} c_{i,j,k}\, h_k$,

characterized by the structure constants $c_{i,k,j.}$

LieGetMa

`LieGetMa[c, J, vα]` generates the M transformations of the for $M_k = e^{-Q_k}$ for $k = 1, \ldots,$
n and $M_{n+1} = M_1 \ldots M_n$ is an $n \times n$ matrix and M is an $n \times n \times n$ tensor corresponding to the structure constants c.

- **c** : $n \times n \times n$ tensor containing the structure constants,
- **J** : $n \times n$ matrix, $h' = Jh$ is a new representation of the $\mathcal{L}_\text{n}$,

- **vα** : dimension n list $v\alpha$ = $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ containing the transformation parameters for the U_A transformation.

```
LieGetMa[c_, J_, vα_] := Module[{dim, k, Q, M},
 dim = Dimensions[c][[1]];
 k = Dimensions[J][[1]];
 M = Table[0, {k1, 1, k + 1}, {k2, 1, dim},
    {k3, 1, dim}];
 M[[k + 1]] = IdentityMatrix[k];
 Do[

   Q =
    Table[Sum[c[[k2, k3, k4]] J[[k1, k2]] vα[[k1]],
      {k2, 1, dim}], {k3, 1, dim}, {k4, 1, dim}];
  M[[k1]] = MatrixExp[-Q];
  M[[k + 1]] = M[[k + 1]].M[[k1]];
  , {k1, 1, k}];
 M
 ]
```

LieGetNu

LieGetNu[M,J] generates the v matrix where v is an n×n matrix.

- **M** : $n \times n \times n$ tensor containing the transformation matrices,
- **J** : $n \times n$ matrix, $h' = Jh$ is a new representation of the $\mathcal{L}_n$.

```
LieGetNu[M_, J_] := Module[{dim, Mk, Ik, νt, νt1},
  dim = Dimensions[M[[1]]][[1]];
  Ik = Normal[SparseArray[{{1, 1} → 1}, dim]];
  νt1 = Ik.J;
  Do[
   Mk = M[[k1]];
   Ik = Normal[SparseArray[{{k1, k1} → 1}, dim]];
   νt = νt1.Mk + Ik.J;
   νt1 = νt;
   , {k1, 2, dim}];
  Transpose[νt]
 ]
```

LieTrans

`LieTrans[M, J, va, vα, k, t]` transforms the coeficients I into va' under the k'th transformation U_k corresponding to the M_k matrix. Under this transformation the original Floquet operator $H - p_t =$ $va^T h - p_t$ is transformed into $H' - p_t =$ $U_k (H - p_t) U_k = va^T M_k h - \dot{\alpha}_k h_k - p_t =$ $(va')^T h - p_t$ where va' is the new set of coefficients.

- **M** : $n \times n \times n$ tensor containing the transformation matrices,
- **J** : $n \times n$ matrix, $h' = Jh$ is a new representation of the $\mathcal{L}_n$,
- **va** : dimension n list $va =$ $\{a_1, a_2, \ldots, a_n\}$ containing the coefficients of the original Floquet operator,

- **vα** : dimension n list $v\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ containing the transformation parameters for the U_A transformation,
- **k** : integer that tags the number of transformation to be used,
- **t** : time parameter.

```
LieTrans[M_, J_, va_, vα_, k_, t_] :=
 va.M[[k]] - D[vα[[k]], t] J[[k]]
```

LieGetu

LieGetu[M, J, va, vα, t] transforms the original coefficients va into u under the complete transformation U_A corresponding to $M_a = M_1 M_2 \ldots M_n$. Under this transformation the original Floquet operator $H - p_t = va.h - p_t$ is transformed into $H' - p_t = U_A (H - p_t) U_A = va^T M_a h - \dot{\alpha}^T v^T h_k - p_t = u^T h - p_t$.

- **M** : is an $n \times n \times n$ tensor containing the transformation matrices,
- **J** : $n \times n$ matrix, $h' = Jh$ is a new representation of the $\mathcal{L}_n$,
- **va** : dimenison n list va = $\{a_1, a_2, \ldots, a_n\}$ containing the coefficients of the original Floquet operator,
- **vα** : dimension n list containing the transformation parameters for the U_A transformation. The α parameters must be functions of the time parameter t.
- t : time parameter.

```
LieGetu[M_, J_, va_, vα_, t_] := Module[{k, vu, vw},
  k = Dimensions[vα][[1]];
  vu = va;
  Do[
   vw = LieTrans[M, J, vu, vα, k1, t];
   vu = vw;
   , {k1, 1, k}];
  vu
 ]
```

LieGetDifEqLambda

LieGetDifEqLambda[J,*vi*,v*α*,v*β*,*λ*,ci] calculates a list containing the differential equations with respect to the auxiliary parameter λ that connects the α and β parameters.

- **J** : $n \times n$ matrix, $h' = J.h$ is a new representation of the $\mathcal{L}_n$,
- ***vi*** : inverse of the n×n matrix v calculated with LieGetNu[M,J],
- **v*α*** : dimension n list $v\alpha =$
  $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ containing the transformation parameters
  for the U_A. The α parameters must be functions of the
  auxiliary parameter λ,
- **v*β*** : dimension n list $v\beta =$
  $\{\beta_1, \beta_2, \ldots, \beta_n\}$ containing the transformation parameters
  for the U_B. The β parameters are NOT functions of the
  auxiliary parameter λ,
- ***λ*** : is the auxiliary parameter that helps relate the α and β transformation parameters.

ci : is a Boolean variable. If ci == True,
the initial conditions $\alpha_1[0] == 0$, $\alpha_2[0] == 0$, ...,
$\alpha_n[0] == 0$ is appended to the list of differential equations. If on ci == False then the output is just the list of differential equations,

- **λ** : is the auxiliary parameter that helps relate the α and β transformation parameters.

```
LieGetDifEqLambda[J_, νi_, vα_, vβ_, λ_, ci_] :=
 Module[{dim, v},
  dim = Dimensions[vα][[1]];
  v = νi.vβ;
  If[ci == True,
   Join[Table[D[vα[[k1]], λ] == v[[k1]], {k1, 1, dim}],
    Table[(vα[[k1]] /. λ → 0) == 0, {k1, 1, dim}]],
   Table[D[vα[[k1]], λ] == v[[k1]], {k1, 1, dim}]
  ]
 ]
```

LieGetNAlpha

LieGetNAlpha[J,*νi*,v*β*,m] numerically calculates the α parameters using Eq. (16).

- **J** : n × n matrix, h' = J.h is a new representation of the $\mathcal{L}_n$,
- ***νi*** : inverse of the n×n matrix ν calculated with LieGetNu[M,J], νi must be a function of vα.
- **vα** : dimension n list vα = $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ containing the transformation parameters for the U_B.

- **vβ** : dimension n list vβ = $\{\beta_1, \beta_2, \ldots, \beta_n\}$ containing the transformation parameters for the U_B.

- **m** : number of iterations.

```
LieGetNAlpha[J_, νi_, να_, νβ_, m_] :=
 Module[{dim, να0, να1, cond, νi0},
  dim = Dimensions[νβ][[1]];
  να0 = Table[0, {k1, 1, dim}];
  Do[
   cond = Table[να[[k1]] → να0[[k1]], {k1, 1, dim}];
   νi0 = νi /. cond;
   να1 = 1/m νi0.νβ + να0;
   να0 = να1;
   , {m1, 0, m - 1}];
  να1
 ]
```

Main Program

Definition of the structure constants $c_{i,j,k}$

The elements of this algebra are given by the operators $h_1 = x^2$, $h_2 = x\,p + p\,x$, $h_3 = p^2$. The following lines define the algebra dimension and the structure constants.

```
n = 3;
d = Table[0, {k1, 1, n}, {k2, 1, n}, {k3, 1, n}];
d[[1, 2, 3]] = 2;
d[[2, 1, 3]] = -2;
d[[1, 3, 1]] = 4;
d[[3, 1, 1]] = -4;
d[[2, 3, 2]] = -4;
d[[3, 2, 2]] = 4;
R = {{1, 0, 0}, {0, 0, 1}, {0, 1, 0}};
Ri = Inverse[R];
c =
  Table[Sum[R[[k1, m1]] R[[k2, m2]] Ri[[m3, k3]]
     d[[m1, m2, m3]], {m1, 1, n}, {m2, 1, n},
    {m3, 1, n}], {k1, 1, n}, {k2, 1, n}, {k3, 1, n}];
```

Since J=$\mathcal{I}$, the structure of the algebra elements is preserved.

```
J = IdentityMatrix[n];
```

Derivation of the time differential equations for α_i(t)

We calculate u using Eq. (13). Notice that in this case vα = {$\alpha_1(t)$, ..., α_n(t)} is a function of time.

```
vα = Table[Subscript[α, k1][t], {k1, 1, n}];
va = Table[Subscript[a, k1], {k1, 1, n}];
Ma = LieGetMa[c, J, vα];
vu = LieGetu[Ma, J, va, vα, t];
Simplify[vu]
```

$$\left\{e^{4\,\alpha_2[t]}\left(a_1 - 4\,a_2\,\alpha_1[t] + 4\,a_3\,\alpha_1[t]^2 - \alpha_1'[t]\right),\; a_2 - 2\,a_3\,\alpha_1[t] + 2\,e^{4\,\alpha_2[t]}\,\alpha_3[t]\left(a_1 - 4\,a_2\,\alpha_1[t] + 4\,a_3\,\alpha_1[t]^2 - \alpha_1'[t]\right) - \alpha_2'[t],\; e^{-4\,\alpha_2[t]}\,a_3 + 4\,e^{4\,\alpha_2[t]}\,\alpha_3[t]^2\left(a_1 - 4\,a_2\,\alpha_1[t] + 4\,a_3\,\alpha_1[t]^2 - \alpha_1'[t]\right) + 4\,\alpha_3[t]\left(a_2 - 2\,a_3\,\alpha_1[t] - \alpha_2'[t]\right) - \alpha_3'[t]\right\}$$

Compare these results with the ones in Eqs. (36)-(38).

The simplified differential equations for $\alpha_i(t)$ are obtained from Eq. (15)

```
ν = LieGetNu[Ma, J];
νi = Inverse[ν];
ε = Simplify[νi.vu];
difeqst = Join[Table[ε[[k1]] == 0, {k1, 1, n}],
   Table[(vα[[k1]] /. {t → 0}) == 0, {k1, 1 n}]];
MatrixForm[difeqst]
```

$$\begin{pmatrix} a_1 - 4\,a_2\,\alpha_1[t] + 4\,a_3\,\alpha_1[t]^2 - \alpha_1'[t] == 0 \\ a_2 - 2\,a_3\,\alpha_1[t] - \alpha_2'[t] == 0 \\ e^{-4\,\alpha_2[t]}\,a_3 - \alpha_3'[t] == 0 \\ \alpha_1[0] == 0 \\ \alpha_2[0] == 0 \\ \alpha_3[0] == 0 \end{pmatrix}$$

Compare these results with the ones in Eqs. (40)-(42).

Relation between α(t) and β(t) via the solution of the

λ differential equations

Using Eq. (17) we workout the λ differential equations for the $\alpha_i(\lambda,t)$ parameters. Note that in this case v$\alpha = \{\alpha_1(\lambda), \ldots, \alpha_n(\lambda)\}$ is a function of λ therefore, M_a and ν have to be recalculated. In **LieGetDifEqLamda** the condition **ci** is set to **False** in order to avoid setting the initial conditions.

```
vα = Table[Subscript[α, k1][λ], {k1, 1, n}];
Ma = LieGetMa[c, J, vα];
ν = LieGetNu[Ma, J];
νi = Inverse[ν];
vβ = Table[Subscript[β, k1], {k1, 1, n}];
difeqsλ =
  Simplify[
   ExpToTrig[LieGetDifEqLambda[J, νi, vα, vβ, λ,
     False]]];
MatrixForm[difeqsλ]
```

$$\begin{pmatrix} (\mathrm{Cosh}[4\,\alpha_2[\lambda]] - \mathrm{Sinh}[4\,\alpha_2[\lambda]])\,\beta_1 == \alpha_1'[\lambda] \\ \beta_2 == 2\,\beta_1\,\alpha_3[\lambda] + \alpha_2'[\lambda] \\ \beta_3 + 4\,\beta_1\,\alpha_3[\lambda]^2 == 4\,\beta_2\,\alpha_3[\lambda] + \alpha_3'[\lambda] \end{pmatrix}$$

Compare these results with Eqs. (26)-(28).

These equations are simple enough that we can attempt to solve them with **DSolve**, however, as mentioned above it is easier to solve the system without initial conditions.

```
sol = Simplify[DSolve[difeqsλ, vα, λ][[1]]]
```

$$\left\{\alpha_3[\lambda] \to \frac{\beta_2 + \sqrt{-\beta_2^2 + \beta_1 \beta_3}\ \mathrm{Tan}\left[2\ (\lambda + C[1])\ \sqrt{-\beta_2^2 + \beta_1 \beta_3}\right]}{2\ \beta_1},\right.$$
$$\alpha_2[\lambda] \to C[2] + \frac{1}{2}\mathrm{Log}\left[\mathrm{Cos}\left[2\ (\lambda + C[1])\ \sqrt{-\beta_2^2 + \beta_1 \beta_3}\right]\right],$$
$$\alpha_1[\lambda] \to C[3] + \frac{1}{2\sqrt{-\beta_2^2 + \beta_1 \beta_3}}\ (\mathrm{Cosh}[4\,C[2]] - \mathrm{Sinh}[4\,C[2]])$$
$$\left.\beta_1\ \mathrm{Tan}\left[2\ (\lambda + C[1])\ \sqrt{-\beta_2^2 + \beta_1 \beta_3}\right]\right\}$$

Setting λ=1 we can obtain a relation between α(t) and β(t) of the form (18).

```
vα1 = Table[Subscript[α, k1], {k1, 1, n}];
eqs = Table[vα1[[k1]] == ((vα[[k1]] /. sol) /. {λ → 1}),
  {k1, 1, n}]
```

$$\left\{\alpha_1 == C[3] + \frac{1}{2\sqrt{-\beta_2^2 + \beta_1 \beta_3}}\ (\mathrm{Cosh}[4\,C[2]] - \mathrm{Sinh}[4\,C[2]])\right.$$
$$\beta_1\ \mathrm{Tan}\left[2\ (1 + C[1])\ \sqrt{-\beta_2^2 + \beta_1 \beta_3}\right],$$
$$\alpha_2 == C[2] + \frac{1}{2}\mathrm{Log}\left[\mathrm{Cos}\left[2\ (1 + C[1])\ \sqrt{-\beta_2^2 + \beta_1 \beta_3}\right]\right],$$
$$\left.\alpha_3 == \frac{\beta_2 + \sqrt{-\beta_2^2 + \beta_1 \beta_3}\ \mathrm{Tan}\left[2\ (1 + C[1])\ \sqrt{-\beta_2^2 + \beta_1 \beta_3}\right]}{2\ \beta_1}\right\}$$

In principle, one could obtain the inverse relations of the form (18), however, it is easier to resort to the eigenvalue one eigenvectors of M_a^{T}.

Relation between α(t) and β(t) via the eigenvalue one

eigenvectors of M_α^T

Working out the inverse relation between α and β might be difficult in this case given the complexity of the previous equations. Therefore, it is convenient to obtain a relation between α(t) and β(t) by obtaining the eigenvalue one eigenvectors of M_α^T. There is one eigenvalue one eigenvector.

```
vα = Table[Subscript[α, k1], {k1, 1, n}];
Ma = LieGetMa[c, J, vα];
Mat = Transpose[Ma[[n + 1]]];
eval = Simplify[Eigenvalues[Mat]];
evec = Simplify[Eigenvectors[Mat]];
eval[[1]]
ρ1 = Simplify[evec[[1]]]
β1 = γ1 ρ1
```

1

$$\left\{\frac{\alpha_1}{\alpha_3}, \frac{1 - e^{-4\alpha_2} + 4\alpha_1\alpha_3}{4\alpha_3}, 1\right\}$$

$$\left\{\frac{\alpha_1\gamma_1}{\alpha_3}, \frac{\left(1 - e^{-4\alpha_2} + 4\alpha_1\alpha_3\right)\gamma_1}{4\alpha_3}, \gamma_1\right\}$$

Compare this last result with the one in Eq. (30).

Exact Solution

Here we plot the main results concerning the exact solution. Here we used Eqs. (24)-(26), (30) and (32).

```
MC[ϕ1_, ϕ0_, t_] := MathieuC[4 ϕ1^2, -2 ϕ0^2, t]
pMC[ϕ1_, ϕ0_, t_] := MathieuCPrime[4 ϕ1^2, -2 ϕ0^2, t]
iMC[ϕ1_, ϕ0_, t_] :=
 NIntegrate[1/MC[ϕ1, ϕ0, s]^2, {s, 0, t}]

ni = 100;
qmax = 0.5;
qmin = 0.0005;
dataβ1 = {};
dataβ2 = {};
dataβ3 = {};
dataM = {};
dataΩ = {};
Do[
 q = qmin + (qmax - qmin) i / ni;
 nC0 = MC[0, q, 0];
 nC = MC[0, q, π];
 npC = pMC[0, q, π];
 niC = iMC[0, q, π];
 nγ =
  ArcTan[
    (Sqrt[(-(4 nC0^2 npC niC)/nC - (-(nC0^2 npC niC)/nC - nC0^2/nC^2 + 1)^2)])/
     (((nC0^2 npC niC)/nC + 1) + nC0^2/nC^2)]/
   (Sqrt[(-(4 nC0^2 npC niC)/nC - (-(nC0^2 npC niC)/nC - nC0^2/nC^2 + 1)^2)]);
 β1 = -1/2 npC/nC nγ;
```

$$\beta 2 = \frac{1}{2}\left(-\frac{nC0^2\ npC\ niC}{nC} - \frac{nC0^2}{nC^2} + 1\right) n\gamma;$$

$$\beta 3 = 2\ nC0^2\ niC\ n\gamma;$$

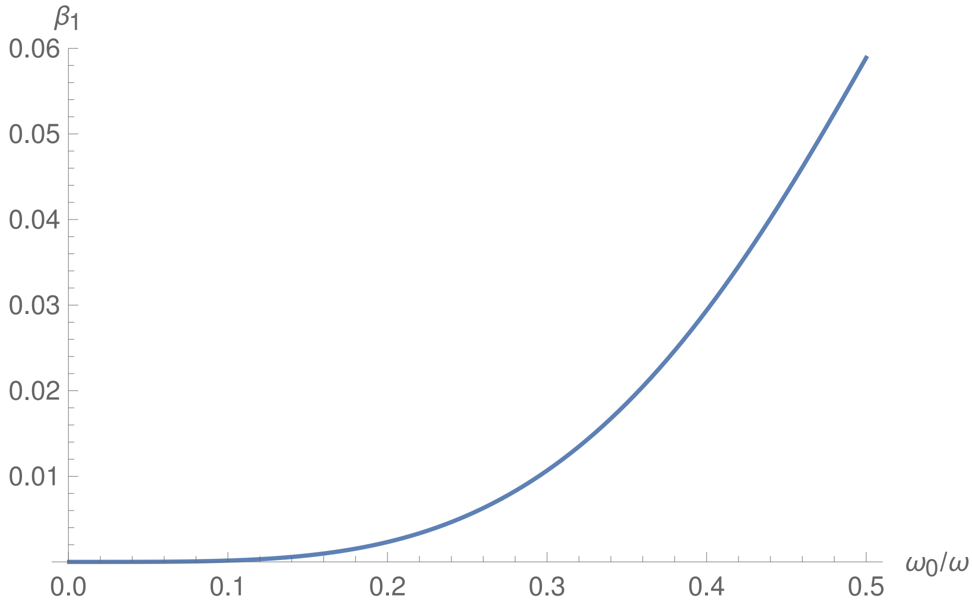
```
AppendTo[dataβ1, {q, β1}];
AppendTo[dataβ2, {q, β2}];
AppendTo[dataβ3, {q, β3}];
```

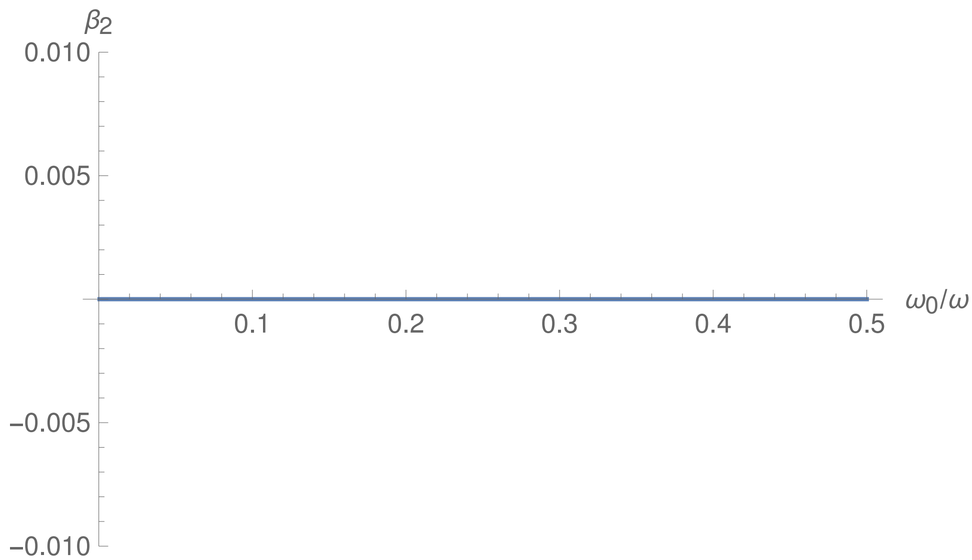
$$\text{AppendTo}\left[\text{dataM},\ \left\{q,\ \frac{\pi}{\beta 3}\right\}\right];$$

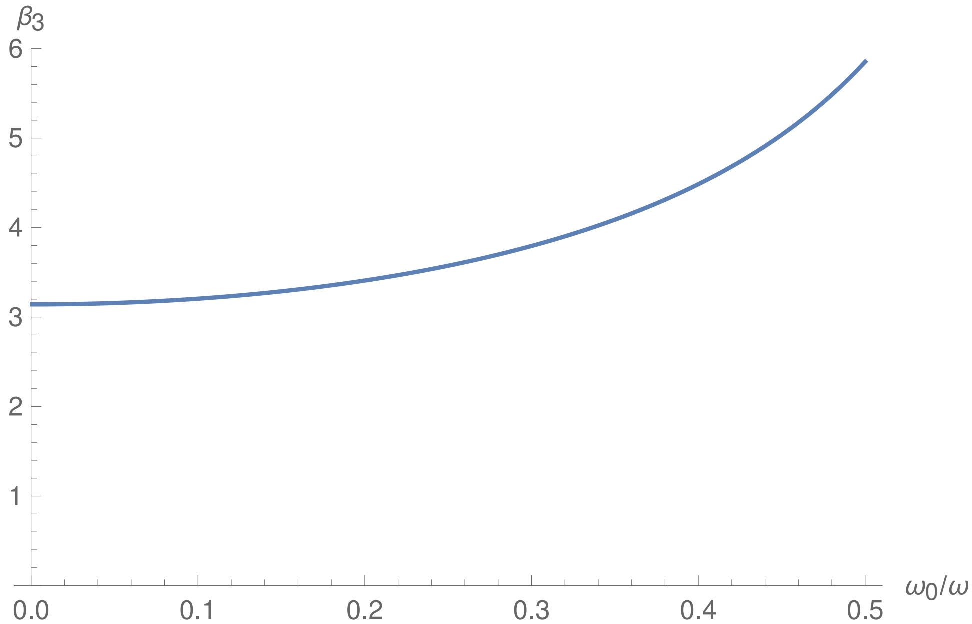
$$\text{AppendTo}\left[\text{data}\Omega,\ \left\{q,\ \sqrt{\frac{\beta 3}{\pi^2}\left(\beta 1 - \frac{\beta 2^2}{\beta 3}\right)}\ \right\}\right];$$

```
, {i, 0, ni}]
```

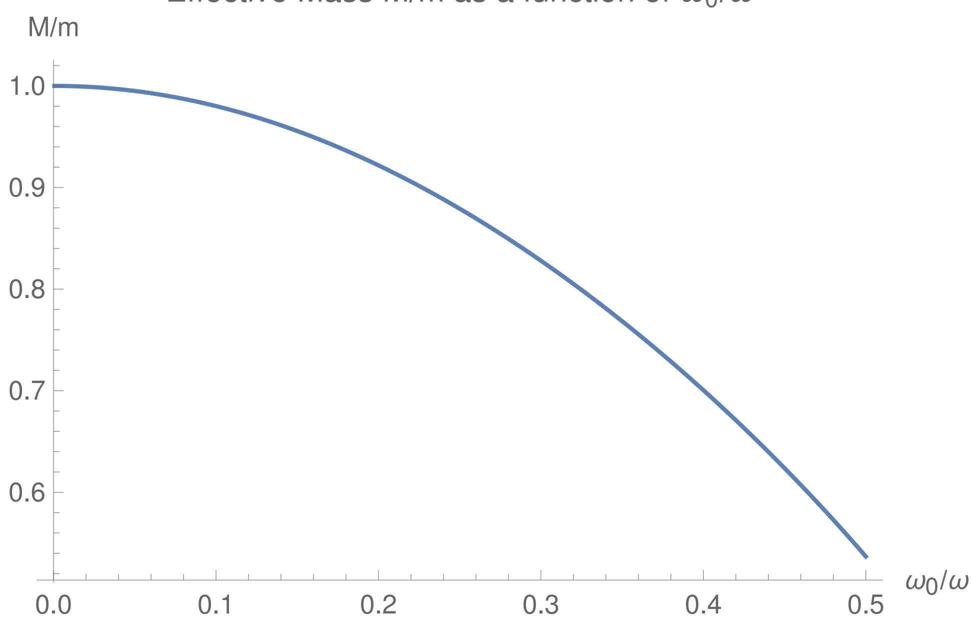
```
ListPlot[dataβ1, AxesLabel → {"ω0/ω", "β1"},
 PlotRange → {0.0, 0.06}, Joined → True]
ListPlot[dataβ2, AxesLabel → {"ω0/ω", "β2"},
 PlotRange → {-0.01, 0.01}, Joined → True]
ListPlot[dataβ3, AxesLabel → {"ω0/ω", "β3"},
 PlotRange → {0, 6.0}, Joined → True]
ListPlot[dataM, AxesLabel → {"ω0/ω", "M/m"},
 PlotLabel →
  "Effective Mass M/m as a function of ω0/ω",
 Joined → True]
ListPlot[dataΩ, AxesLabel → {"ω0/ω", "Ω/ω"},
 PlotLabel →
  "Effective angular frequency Ω/ω as a function
    of ω0/ω", Joined → True]
```

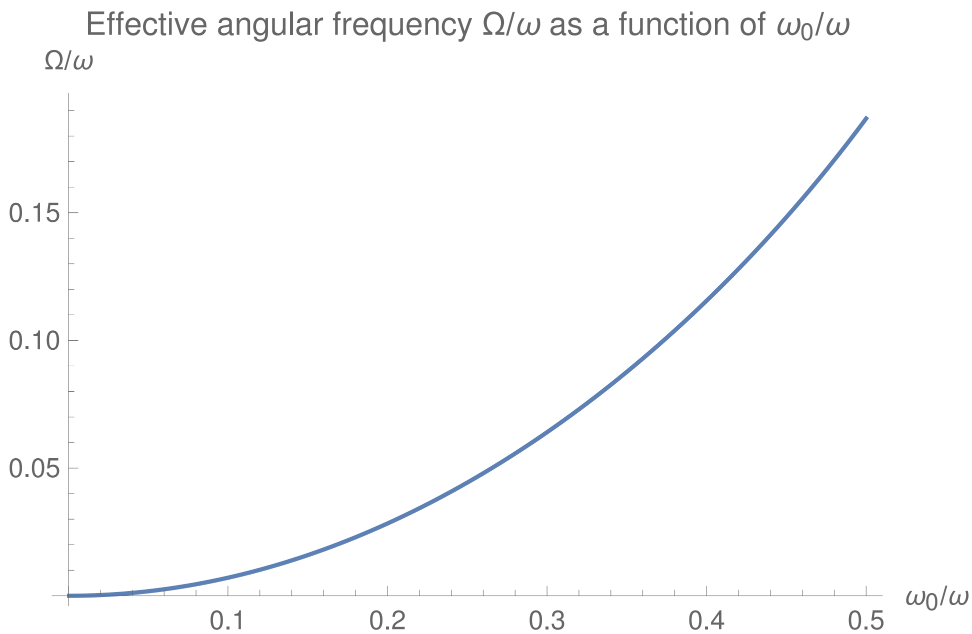

β_1
0.06
0.05
0.04
0.03
0.02
0.01
0.0
0.1
0.2
0.3
0.4
0.5
ω_0/ω


β_2
0.010
0.005
−0.005
−0.010
0.1
0.2
0.3
0.4
0.5
ω_0/ω



Effective Mass M/m as a function of ω_0/ω





Numerical Solution

First we do some preliminary calculations.

```
vα = Table[Subscript[α, k1], {k1, 1, n}];
Ma = LieGetMa[c, J, vα];
ν = LieGetNu[Ma, J];
νi = Inverse[ν];
```

To illustrate the numerical method we start by obtaining just one point of the effective Hamiltonian H_e for the quantum harmonic oscillator with time-dependent frequency (Paul trap). This means that the β parameters are calculated for one value of the mass (m=1.0) and ω_0=2.0. As an example we have chosen $\omega = 4.0$.

```
m = 100;
𝑚 = 1.0;
ω0 = 2.0;
ω = 10.0;
T = 2 π / ω
```

```
0.628319
```

We numerically solve (**NDSolve**) the time differential euqtions **difeqst** for the α parameters obtained above for m=1.0, ω_0=10.0 and $\omega = 4.0$.

```
vαt = Table[Subscript[α, k1][t], {k1, 1, n}];
difeqstn =
  difeqst /. {a₁ → 𝑚/2 ω0^2 Cos[ω t], a₂ → 0, a₃ → 1/(2 𝑚)};
solt = NDSolve[difeqstn, vαt, {t, 0, T}][[1]];
```

The solution is then used to obtain α(T), the α parameters evaluated in *T*.

```
vαT = vαt /. solt /. {t → T}
```

```
{0.0234735, -0.00795757, 0.344456}
```

To simplify the calculation we find the eigenvalue one eigenvectors of M_a^T evaluated in $\alpha(T)$. We use the eigenvalues and eigenvectors calculated in the previous section. To avoid divergencies we multiply the eigenvectors by α_3

```
evaln =
 eval[[1]] /. Table[vα[[k1]] → vαT[[k1]], {k1, 1, n}]
evecn = α3 evec[[1]] /.
   Table[vα[[k1]] → vαT[[k1]], {k1, 1, n}];
ρ = evecn
```

1

$\{0.0234735, 1.05328 \times 10^{-8}, 0.344456\}$

We calculate α as a function of β using Eq. (103) and verify that the condition (104) is met.
The inverse function in Eq. (18) is calculated by finding the minimum of the χ function for α(T)

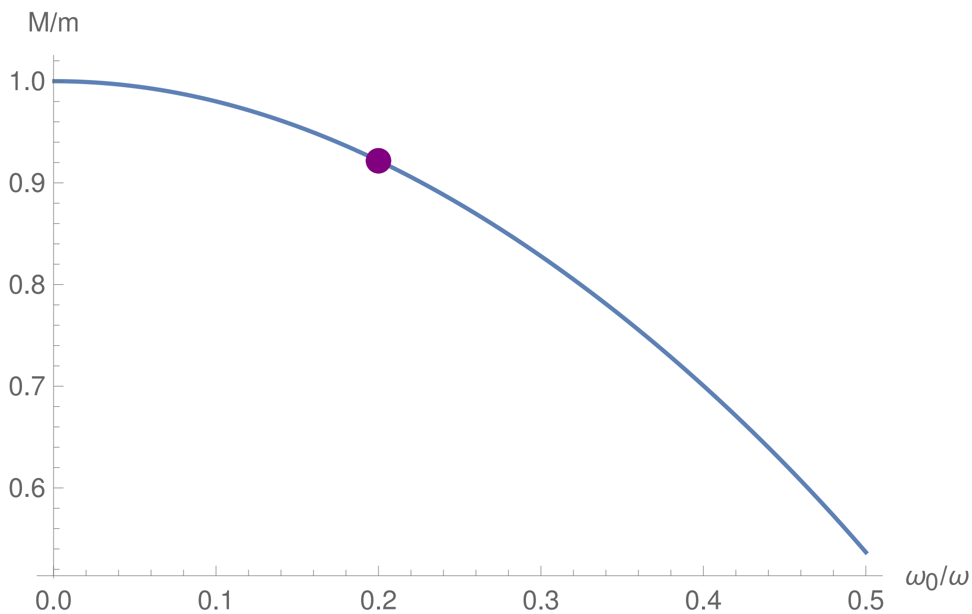
```
χ[γ_] := Norm[vαT - LieGetNAlpha[J, νi, vα, γ ρ, m]]^2;
γ = Sort[Table[{γ, χ[γ]}, {γ, 0.45, 1.01, 0.0005}],
   #1[[2]] < #2[[2]] &][[1]]
vβN = γ[[1]] ρ
```

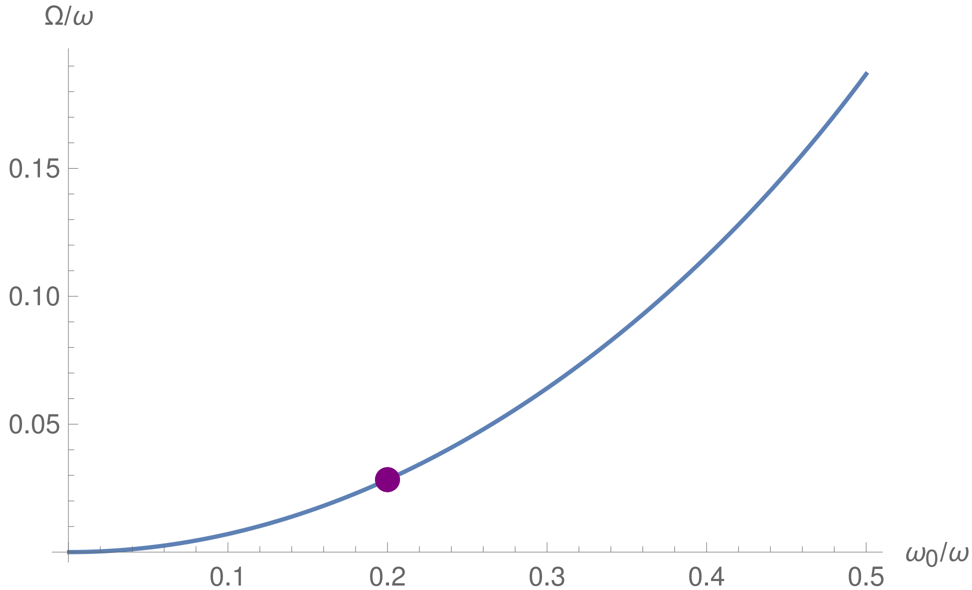
$\{0.9895, 7.19231 \times 10^{-9}\}$

$\{0.023227, 1.04222 \times 10^{-8}, 0.340839\}$

Therefore, for ω=10.0 we have
$\beta_1 = 0.023227$, $\beta_2 = 0.0$ and $\beta_3 = 0.340839$. To test the numerical solution we can compare it with the previously obtained exact solutions for Ω/ω and M/m as functions of ω_0 / ω .

```
ListPlot[{dataM, {{ω0 / ω, π / (m ω vβN[[3]])}}},
  PlotStyle →
   {Automatic, Directive[PointSize[0.03], Purple]},
  Joined → {True, False}, AxesLabel → {"ω₀/ω", "M/m"}]
ListPlot[{dataΩ, {{ω0 / ω, (1/π) Sqrt[vβN[[1]] vβN[[3]]]}}},
  PlotStyle →
   {Automatic, Directive[PointSize[0.03], Purple]},
  Joined → {True, False}, AxesLabel → {"ω₀/ω", "Ω/ω"}]
```





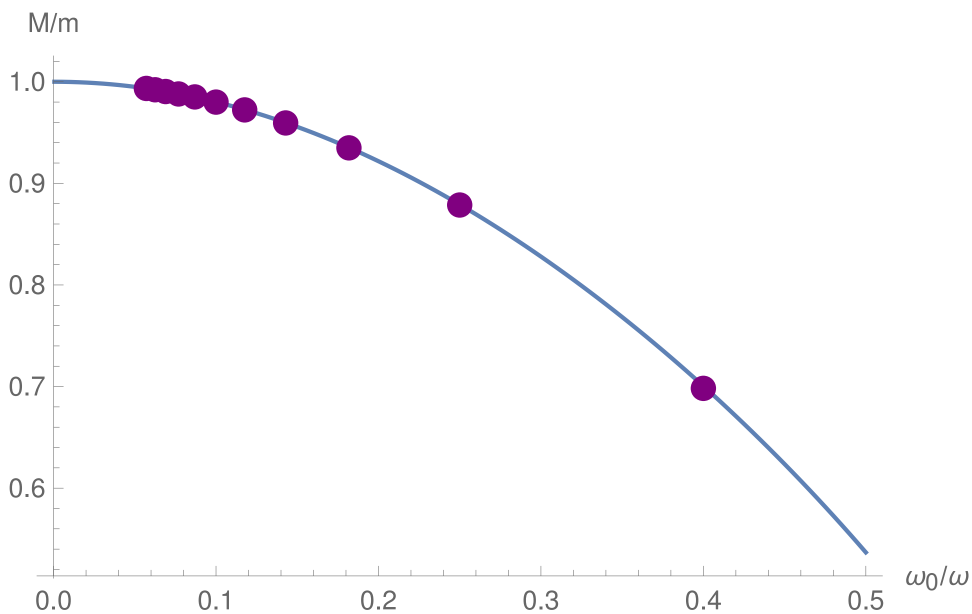
The reader may change the value of the parameter ω and repeat the calculation to verify that the numerical solution is consisten with the exact one.
Now we add a loop to the procedure to calculate the effective Hamiltonian for various values of the parameter ω.

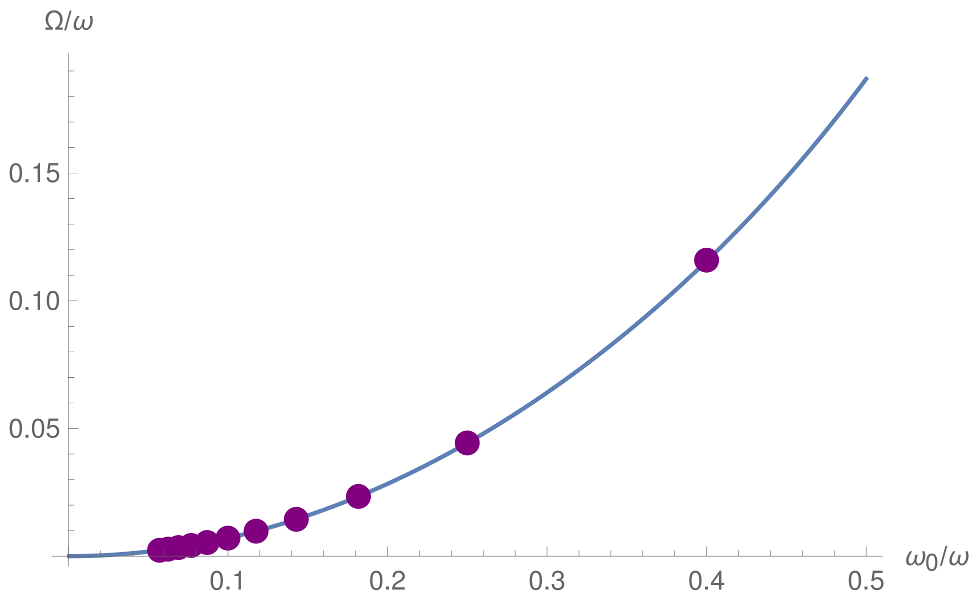
```
m = 100;
m = 1.0;
ω0 = 2.0;
ωmin = 5.0;
ωmax = 35.0;
Δω = 3.0;
dataNM = {};
dataNΩ = {};
prog = 0.0;
ProgressIndicator[Dynamic[prog]]
Do[
 prog = (ω - ωmin) / (ωmax - ωmin);
 T = 2 π / ω;
```

```
vαt = Table[Subscript[α, k1][t], {k1, 1, n}];
difeqstn =
 difeqst /. {a₁ → m/2 ω0^2 Cos[ω t], a₂ → 0, a₃ → 1/(2 m)};
solt = NDSolve[difeqstn, vαt, {t, 0, T}][[1]];
vαT = vαt /. solt /. {t → T};
evaln =
 eval[[1]] /. Table[vα[[k1]] → vαT[[k1]], {k1, 1, n}];
evecn = α₃ evec[[1]] /.
  Table[vα[[k1]] → vαT[[k1]], {k1, 1, n}];
ρ = evecn;
χ[γ_] := Norm[vαT - LieGetNAlpha[J, νi, vα, γ ρ, m]]^2;
γ = Sort[Table[{γ, χ[γ]}, {γ, 0.45, 1.01, 0.0005}],
   #1[[2]] < #2[[2]] &][[1, 1]];
vβN = γ ρ;
AppendTo[dataNM, {ω0 / ω, π/(m ω vβN[[3]])}];
AppendTo[dataNΩ, {ω0 / ω, 1/π Sqrt[vβN[[1]] vβN[[3]]]}];
, {ω, ωmin, ωmax, Δω}]
```

Superimposing the exact solution for Ω/ω and M/m as functions of ω_0/ω, to the numerical one, we obtain the following plot

```
ListPlot[{dataM, dataNM},
 PlotStyle →
  {Automatic, Directive[PointSize[0.03], Purple]},
 Joined → {True, False}, AxesLabel → {"ω₀/ω", "M/m"}]
ListPlot[{dataΩ, dataNΩ},
 PlotStyle →
  {Automatic, Directive[PointSize[0.03], Purple]},
 Joined → {True, False}, AxesLabel → {"ω₀/ω", "Ω/ω"}]
```





We observe that the numerical solution for the effective Hamiltonian is identical to the exact one.