Modulated Optical Lattice

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 $\mathcal{L}_\mathrm{n}$,
- **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 $\mathcal{L}_\mathrm{n}$,
- **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 × 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 vα = $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ containing the transformation parameters for the U_A. The α parameters must be functions of the auxiliary parameter λ,
- **v*β*** : dimension n list vβ = $\{\beta_1, \beta_2, \ldots, \beta_n\}$ 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,vβ,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_, vα_, vβ_, m_] :=
 Module[{dim, vα0, vα1, cond, νi0},
  dim = Dimensions[vβ][[1]];
  vα0 = Table[0, {k1, 1, dim}];
  Do[
   cond = Table[vα[[k1]] → vα0[[k1]], {k1, 1, dim}];
   νi0 = νi /. cond;
   vα1 = (1/m) νi0.vβ + vα0;
   vα0 = vα1;
   , {m1, 0, m - 1}];
  vα1
 ]
```

Main Program

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

The elements of this algebra are given by the operators $h_1 = H_0$, $h_2 = H$, $h_3 = V$. 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, 3, 2]] = -1;
d[[3, 1, 2]] = 1;
d[[2, 3, 1]] = 1;
d[[3, 2, 1]] = -1;
R = {{0, 0, 1}, {0, 1, 0}, {1, 0, 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];
MatrixForm[Simplify[vu]]
```

$$\begin{pmatrix} a_1 - \alpha_1'[t] \\ \mathrm{Cos}[\alpha_1[t]]\, a_2 - \mathrm{Sin}[\alpha_1[t]]\, a_3 + \alpha_3[t]\, (a_1 - \alpha_1'[t]) - \alpha_2'[t] \\ \mathrm{Sin}[\alpha_1[t]]\, a_2 + \mathrm{Cos}[\alpha_1[t]]\, a_3 - \alpha_2[t]\, (a_1 - \alpha_1'[t]) - \alpha_3'[t] \end{pmatrix}$$

Compare these results with the ones in Eqs. (57)-(59).

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

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

$$\begin{pmatrix} a_1 - \alpha_1'[t] == 0 \\ \text{Cos}[\alpha_1[t]]\, a_2 - \text{Sin}[\alpha_1[t]]\, a_3 - \alpha_2'[t] == 0 \\ \text{Sin}[\alpha_1[t]]\, a_2 + \text{Cos}[\alpha_1[t]]\, a_3 - \alpha_3'[t] == 0 \\ \alpha_1[0] == 0 \\ \alpha_2[0] == 0 \\ \alpha_3[0] == 0 \end{pmatrix}$$

Compare these results with the ones in Eqs. (61)-(63).

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

Using Eq. (16) 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 v have to be recalculated. In **LieGetDifEqLamda** the condition **ci** is set to **True** in order to include 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[LieGetDifEqLambda[J, νi, vα, vβ, λ, True]];
MatrixForm[difeqsλ]
```

$$\begin{pmatrix} \beta_1 == \alpha_1'[\lambda] \\ \beta_2 == \beta_1\,\alpha_3[\lambda] + \alpha_2'[\lambda] \\ \beta_3 + \beta_1\,\alpha_2[\lambda] == \alpha_3'[\lambda] \\ \alpha_1[0] == 0 \\ \alpha_2[0] == 0 \\ \alpha_3[0] == 0 \end{pmatrix}$$

Compare these results with Eqs. (66)-(68).

These equations are simple enough that we can attempt to solve them with **DSolve**.

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

$$\left\{\alpha_1[\lambda] \to \lambda\,\beta_1,\ \alpha_2[\lambda] \to \frac{\mathrm{Sin}[\lambda\,\beta_1]\,\beta_2 + (-1 + \mathrm{Cos}[\lambda\,\beta_1])\,\beta_3}{\beta_1},\ \alpha_3[\lambda] \to \frac{\beta_2 - \mathrm{Cos}[\lambda\,\beta_1]\,\beta_2 + \mathrm{Sin}[\lambda\,\beta_1]\,\beta_3}{\beta_1}\right\}$$

Compare these results with Eqs. (69)-(71).

Setting λ=1 we can obtain a relation between α(t) and β(t) of the form (17).

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

$$\begin{pmatrix} \alpha_1 == \beta_1 \\ \alpha_2 == \frac{\text{Sin}[\beta_1]\,\beta_2 + (-1+\text{Cos}[\beta_1])\,\beta_3}{\beta_1} \\ \alpha_3 == \frac{\beta_2 - \text{Cos}[\beta_1]\,\beta_2 + \text{Sin}[\beta_1]\,\beta_3}{\beta_1} \end{pmatrix}$$

Compare this results with Eqs. (69)-(71).

Relation between α(t) and β(t) via the eigenvalue one eigenvectors of M_a^T

Another way to obtain a relation between α(t) and β(t) for the modulated optical lattice is by obtaining the eigenvalue one eigenvectors of M_a^T.

```
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 = evec[[1]]
```

1

$$\left\{\frac{2-2\,\text{Cos}[\alpha_1]}{(-1+\text{Cos}[\alpha_1])\,\alpha_2+\text{Sin}[\alpha_1]\,\alpha_3},\ \frac{\text{Sin}[\alpha_1]\,\alpha_2-(-1+\text{Cos}[\alpha_1])\,\alpha_3}{(-1+\text{Cos}[\alpha_1])\,\alpha_2+\text{Sin}[\alpha_1]\,\alpha_3},\ 1\right\}$$

Compare these results with Eq. (76).

Substituting the explicit form of γ_1 we obtain the $\beta(t)$ parameters

```
Simplify[γ1 ρ1]
Simplify[γ1 ρ1 /. {γ1 → α1 / ρ1[[1]]}]
```

$$\left\{\frac{(2-2\,\mathrm{Cos}[\alpha_1])\,\gamma_1}{(-1+\mathrm{Cos}[\alpha_1])\,\alpha_2+\mathrm{Sin}[\alpha_1]\,\alpha_3},\ \frac{(\mathrm{Sin}[\alpha_1]\,\alpha_2-(-1+\mathrm{Cos}[\alpha_1])\,\alpha_3)\,\gamma_1}{(-1+\mathrm{Cos}[\alpha_1])\,\alpha_2+\mathrm{Sin}[\alpha_1]\,\alpha_3},\ \gamma_1\right\}$$

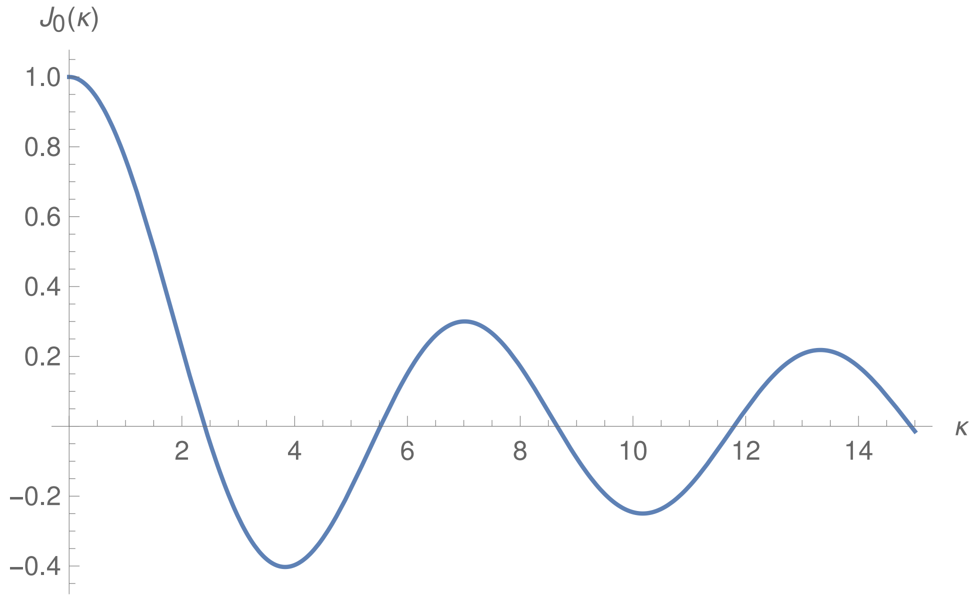
$$\left\{\alpha_1,\ \frac{\alpha_1\,(\mathrm{Sin}[\alpha_1]\,\alpha_2-(-1+\mathrm{Cos}[\alpha_1])\,\alpha_3)}{2-2\,\mathrm{Cos}[\alpha_1]},\ \frac{\alpha_1\,((-1+\mathrm{Cos}[\alpha_1])\,\alpha_2+\mathrm{Sin}[\alpha_1]\,\alpha_3)}{2-2\,\mathrm{Cos}[\alpha_1]}\right\}$$

Compare these results with Eqs. (77)-(80).

Exact Solution

In the calculations above we found that $\beta_1 = \beta_2 = 0$ and $\beta_3 = T J_0(\kappa)$. In the next plot we observe this parameter as a function of κ.

```
Plot[BesselJ[0, κ], {κ, 0, 15},
 AxesLabel → {"κ", "J₀(κ)"}]
```



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 optical lattice. This means that the β parameters are calculated for one value of κ (κ=7.0). As an example we have chosen $\omega = 3.0$.

```
ω = 3.0;
κ = 7.0;
T = 2 π / ω;
```

We numerically solve (**NDSolve**) the time differential euqtions **difeqst** for the α parameters obtained above for $\omega = 3.0$ and κ=7.0.

```
vαt = Table[Subscript[α, k1][t], {k1, 1, n}];
difeqstn = difeqst /. {a₁ → ω κ Cos[ω t], a₂ → 0, a₃ → 1};
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}
```

$\{2.4049 \times 10^{-9}, 1.32604 \times 10^{-8}, 0.628485\}$

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

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

1

$\{0., 2.1099 \times 10^{-8}, 1\}$

$\{0., 2.1099 \times 10^{-8}, 1\}$

Using Eq. (21), β is given by

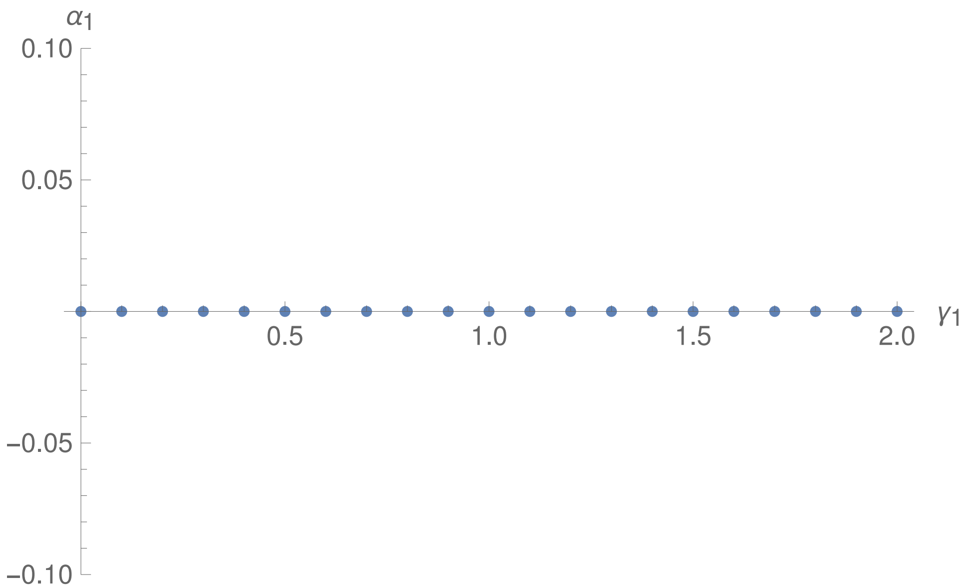
```
vβ = γ ρ
```

$\{0., 2.1099 \times 10^{-8}\,\gamma, \gamma\}$

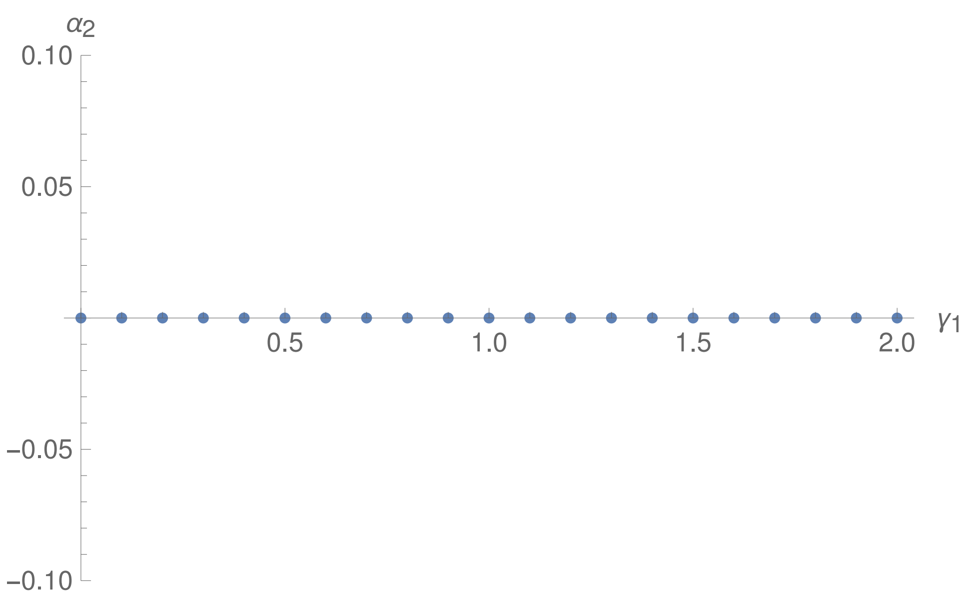
where γ is a parameter yet to be determined.

We calculate α as a function of β using Eq. (103) and verify that condition (104) is met.

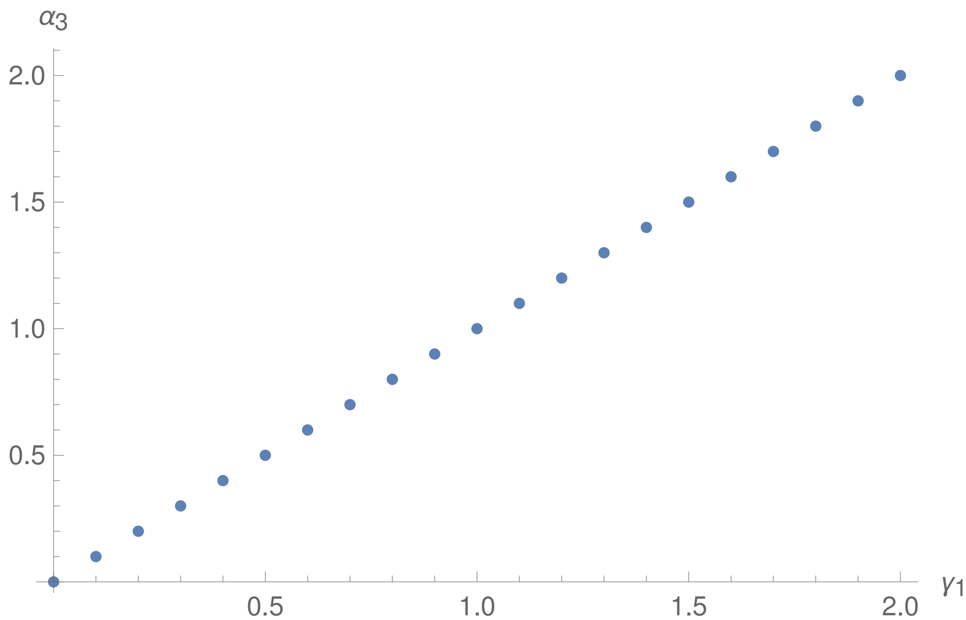
```
m = 100;
plotα1 = {};
plotα2 = {};
plotα3 = {};
Do[
 vαN = LieGetNAlpha[J, vi, vα, vβ, m];
 AppendTo[plotα1, {γ, Re[vαN[[1]]]}];
 AppendTo[plotα2, {γ, Re[vαN[[2]]]}];
 AppendTo[plotα3, {γ, Re[vαN[[3]]]}];
 , {γ, 0, 2.0, 0.1}]
ListPlot[plotα1, AxesLabel → {γ1, α1},
 PlotRange → {-0.1, 0.1}]
ListPlot[plotα2, AxesLabel → {γ1, α2},
 PlotRange → {-0.1, 0.1}]
ListPlot[plotα3, AxesLabel → {γ1, α3}, PlotRange → All]
```



α_1
0.10
0.05
−0.05
−0.10
0.5
1.0
1.5
2.0
γ_1



α_2
0.10
0.05
−0.05
−0.10
0.5
1.0
1.5
2.0
γ_1



We notice that
$\alpha_1 = \alpha_2 = \beta_1 =$

$\beta_2 = 0$ and $\beta_3 = \alpha_3$ are consistent with the analytical results.

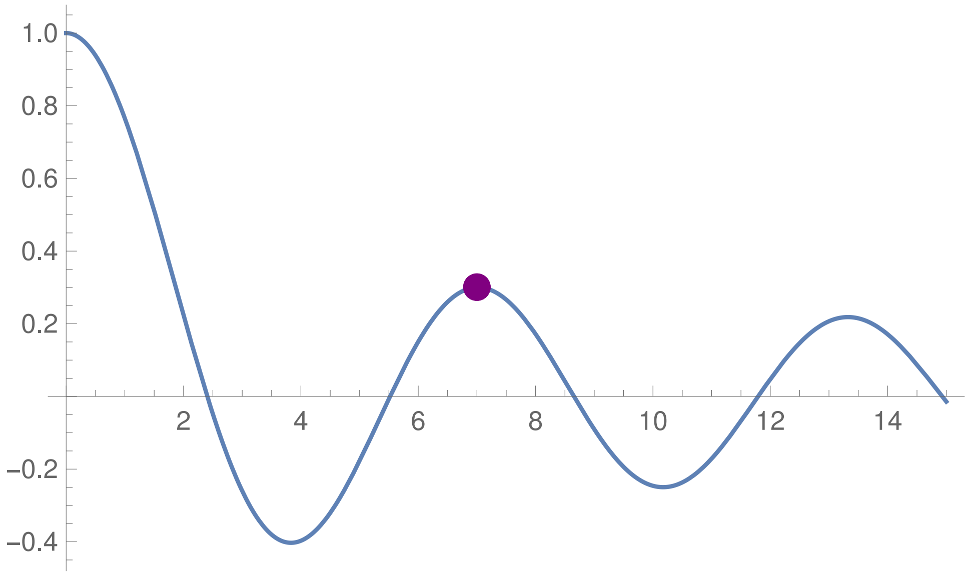
From these caclulations it is clear that β_1=β_2=0, and thus it still remains to calculate β_3. To this end, we use Eq. (18). The inverse function in Eq. (18) is calculated via the χ function of α(T).

```
χ[γ_] := Norm[vαT - LieGetNAlpha[J, vi, vα, γ ρ, m]]^2
γ = Sort[Table[{γ, χ[γ]}, {γ, -1.3, 2.3, 0.01}],
    #1[[2]] < #2[[2]] &][[1, 1]];
vβN = γ ρ
```

```
{0., 1.32923 × 10^-8, 0.63}
```

Therefore, for κ=7.0, $\beta_3 / T = 0.08735$. Plotting this point along with the analytical result $\beta_3 / T = J_0(\kappa)$ we observe

```
Show[{Plot[BesselJ[0, κ], {κ, 0, 15}],
  ListPlot[{{κ, 1/T vβ[[3]]}}, AxesLabel → {"κ", "β3"},
   PlotStyle → Directive[Purple, PointSize[0.03]]]}]
```



The reader may change the value of the parameter κ 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 κ.

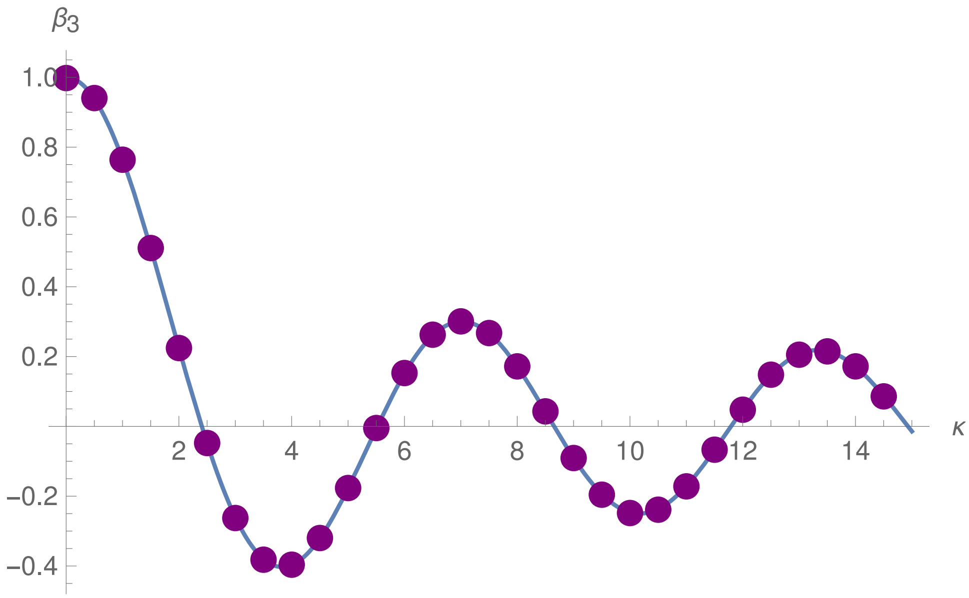
```
ω = 3.0;
T = 2 π / ω
```

```
2.0944
```

```
eval = Eigenvalues[Mat];
evec = Eigenvectors[Mat];
κmin = 0.001;
κmax = 15.0;
Δκ = 0.5;
dataβ3 = {};
proc = 0.0;
ProgressIndicator[Dynamic[proc]]
Do[
 proc = (κ - κmin) / (κmax - κmin);
 vαt = Table[Subscript[α, k1][t], {k1, 1, n}];
 difeqstn = difeqst /. {a₁ → ω κ Cos[ω t], a₂ → 0, a₃ → 1};
 solt = NDSolve[difeqstn, vαt, {t, 0, T}][[1]];
 vαT = vαt /. solt /. {t → T};
 ρ₁ = evec[[1]] /. Table[vα[[k1]] → vαT[[k1]],
     {k1, 1, n}];
 χ[γ_] := Norm[vαT - LieGetNAlpha[J, vi, vα, γ ρ, m]]^2;
 γ = Sort[Table[{γ, χ[γ]}, {γ, -1.3, 2.3, 0.01}],
     #1[[2]] < #2[[2]] &][[1, 1]];
 vβN = γ ρ;
 AppendTo[dataβ3, {κ, 1/T vβ[[3]]}];
 , {κ, κmin, κmax, Δκ}]
```

Superimposing the exact solution for the effective Hamiltonian, $\beta_3 / T = J_0(\kappa)$, to the numerical one, we obtain the following plot

```
Show[
 {Plot[BesselJ[0, κ], {κ, 0, 15},
   AxesLabel → {"κ", "β₃"}],
  ListPlot[dataβ3,
   PlotStyle → Directive[Purple, PointSize[0.03]]]}]
```



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