Linear Potential

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.

```
In[352]:= 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 $\mathcal{L}_n$.

```
In[353]:= 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 l 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.

In[354]:=
```
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.

In[355]:=
```
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,vα,vβ,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],
- **vα** : 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,
- **vβ** : 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.

In[356]:=
```
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}]
  ]
 ]
```

Main Program

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

The elements of the algebra for a linear potential are given by the operators $h_1 = 1$, $h_2 = x$, $h_3 = p$, $h_4 = p^2$. The following lines define the algebra dimension and the structure constants.

In[357]:=
```
n = 4;
d = Table[0, {k1, 1, n}, {k2, 1, n}, {k3, 1, n}];
d[[2, 3, 1]] = 1;
d[[3, 2, 1]] = -1;
d[[2, 4, 3]] = 2;
d[[4, 2, 3]] = -2;
R = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0},
   {0, 0, 0, 1}};
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.

In[366]:=
```
J = IdentityMatrix[n];
```

Derivation of the time differential equations for α_i(t)

We calculate u using Eq. (12). Notice that in this case vα = {$\alpha_1(t)$, ..., α_n(t)} is a function of time.

In[367]:=
```
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]]
```

Out[371]//MatrixForm=

$$\begin{pmatrix} a_1 - a_3\,\alpha_2[t] + a_4\,\alpha_2[t]^2 - \alpha_1'[t] + \alpha_3[t]\,(a_2 - \alpha_2'[t]) \\ a_2 - \alpha_2'[t] \\ a_3 - 2\,a_4\,\alpha_2[t] + 2\,\alpha_4[t]\,(a_2 - \alpha_2'[t]) - \alpha_3'[t] \\ a_4 - \alpha_4'[t] \end{pmatrix}$$

The simplified differential equations for $\alpha_i(t)$ are obtained from Eq. (14)

In[372]:=
```
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]
```

Out[376]//MatrixForm=

$$\begin{pmatrix} a_1 - a_3\,\alpha_2[t] + a_4\,\alpha_2[t]^2 - \alpha_1'[t] == 0 \\ a_2 - \alpha_2'[t] == 0 \\ a_3 - 2\,a_4\,\alpha_2[t] - \alpha_3'[t] == 0 \\ a_4 - \alpha_4'[t] == 0 \\ \alpha_1[0] == 0 \\ \alpha_2[0] == 0 \\ \alpha_3[0] == 0 \\ \alpha_4[0] == 0 \end{pmatrix}$$

Compare these results with the ones in Eqs. (133)-(136).

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.

In[377]:=
```
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]
```

Out[383]//MatrixForm=

$$\begin{pmatrix} \beta_1 == \beta_2\,\alpha_3[\lambda] + \alpha_1'[\lambda] \\ \beta_2 == \alpha_2'[\lambda] \\ \beta_3 == 2\,\beta_2\,\alpha_4[\lambda] + \alpha_3'[\lambda] \\ \beta_4 == \alpha_4'[\lambda] \\ \alpha_1[0] == 0 \\ \alpha_2[0] == 0 \\ \alpha_3[0] == 0 \\ \alpha_4[0] == 0 \end{pmatrix}$$

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

In[384]:=
```
sol = Simplify[DSolve[difeqsλ, vα, λ][[1]]]
```

Out[384]=
$$\left\{\alpha_1[\lambda] \to \frac{1}{6}\lambda\,(6\,\beta_1 + \lambda\,\beta_2\,(-3\,\beta_3 + 2\,\lambda\,\beta_2\,\beta_4)),\ \alpha_3[\lambda] \to \lambda\,(\beta_3 - \lambda\,\beta_2\,\beta_4),\ \alpha_4[\lambda] \to \lambda\,\beta_4,\ \alpha_2[\lambda] \to \lambda\,\beta_2\right\}$$

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).

In[385]:=
```
vα1 = Table[Subscript[α, k1][t], {k1, 1, n}];
eqs = Table[vα1[[k1]] == ((vα[[k1]] /. sol) /. {λ → 1}),
   {k1, 1, n}];
MatrixForm[eqs]
```

Out[387]//MatrixForm=
$$\begin{pmatrix} \alpha_1[t] == \frac{1}{6}\,(6\,\beta_1 + \beta_2\,(-3\,\beta_3 + 2\,\beta_2\,\beta_4)) \\ \alpha_2[t] == \beta_2 \\ \alpha_3[t] == \beta_3 - \beta_2\,\beta_4 \\ \alpha_4[t] == \beta_4 \end{pmatrix}$$

In[388]:=
```
Solve[eqs, vβ]
```

Out[388]=
$$\left\{\left\{\beta_1 \to \frac{1}{6}\left(6\,\alpha_1[t] + 3\,\alpha_2[t]\,\alpha_3[t] + \alpha_2[t]^2\,\alpha_4[t]\right),\ \beta_2 \to \alpha_2[t],\ \beta_3 \to \alpha_3[t] + \alpha_2[t]\,\alpha_4[t],\ \beta_4 \to \alpha_4[t]\right\}\right\}$$

This final result allows to write the evolution operator in the form (5).