Ruby Lie algebra パッケージマニュアル

谷村 慈則

2015 年 2 月 1 日

1 はじめに

Ruby Lie algebra パッケージは Lie 代数に関する計算をする開発中のパッケージです。rep.rb を load すれば全てのクラスとメソッドを用いることができます。

2 追加したメソッド

rep.rb を load すると標準添付ライブラリから mathn と matrix が require されます。

mathn を require することで有理数を扱うことができます。このパッケージはコンピュータ上で線形代数を厳密に運用するために有理数体上で計算することを前提に作られています。

matrix パッケージはベクトル演算や行列計算をするためのパッケージですが、機能が足りないのでいくつか追加しました。

`Vector`

- `*(other) => Vector`
  `other` が `Vector` の場合は内積の値を返します。
  `other` が `Rational` の場合はスカラー倍を返します。
- `-@ => Vector`
  `self * (-1)`
- `Vector.zero(n) => Vector`
  `n` 次元 0 ベクトルを返します。
- `normal(b) => Vector`
  `b` は `Subspace` を想定しています。`self` の `b` に関する垂直成分を返します。
- `proj(b) => Vector`
  `b` は `Subspace` を想定しています。`self` の `b` への射影を返します。
- `clean => Vector`
  分母を払ってから最大公約数で各成分を割ります。

- to_mat(n,m) => Matrix
  nm 次元ベクトルを m 個の n 次元ベクトルに分解して横に並べることで n × m 行列にします。
- wrt(b) => Vector
  Subspace クラス b の基底によって self を行列表示します。
- P(i,j) => Vector
  self の第 i 成分と第 j 成分を交換した Vector を返します。

Matrix

- Matrix.elementary(k,l,n) => Matrix
  k,l 成分が 1 の n 次の行列単位を返します。
- anti_sym? => boolean
  self が反対称行列かどうかを判定し、反対称行列なら true を返します。
- to_vec => Vector
  各列を縦に並べて縦ベクトルを返します。
- bracket(other) => Matrix
  self * other - other * self
- mprint => nil
  self を行列形式で表示します。
- to_subsp => Subspace
  self の像を返します。
- P(i,j) => Matrix
  self の i 行と j 行を交換した Matrix を返します。
- Q(i,r) => Matrix
  i 行を r 倍した Matrix を返します。
- R(i,j,r) => Matrix
  i 行の r 倍を j 行に足した Matrix を返します。
- rP(i,j) => Matrix
  i 列と j 列を交換した Matrix を返します。
- gauss => Subspace
  ガウスの消去法を用いて self の解空間を決定します。

3 Subspace クラス

Subspace は affine 空間の部分線型空間のクラスです。データは直交基底を用いて扱います。正規直交基底でないのは、有理数体上で扱うことを前提としているためです。

`Subspace < Array`

- `new(v) => Subspace`
  `Vector` の配列 `v` の要素によって張られる部分線型空間を返します。
  Schmidt の直交化法を用いて直交基底を求めます。
- `Subspace.std(n) => Subspace`
  `n` 次元 affine 空間の標準基底を返します。
- `in(other) => boolean`
  `self` $\subset$ `other` かどうかを判定し、真ならば `true` を返します。
- `==(other) => boolean`
  affine 空間の部分線型空間として `self` と `other` が同じものかどうか判定し、同じものであるならば `true` を返します。
- `cap(other) => Subspace`
  `self` $\cap$ `other` を返します。
- `+(other) => Subspace`
  `self` $+$ `other` を返します。
- `to_mat => Matrix`
  `self` を定める直交基底を並べて行列化します。
- `normal => Subspace`
  `self` の直交補空間を返します。
- `bprint => nil`
  `self` を定める直交基底を並べて表示します。
- `subalgebra(c) => Liealgebra`
  `self` を `Liealgebra c` の部分 Lie 代数とみて、`self` を Lie 代数として返します。
- `subalgebra?(c) => boolean`
  `self` が `Liealgebra c` の部分 Lie 代数かどうかを判定して、部分 Lie 代数ならば `true` を返します。
- `ideal?(c) => boolean`
  `self` が `Liealgebra c` のイデアルかどうかを判定して、部分 Lie 代数ならば `true` を返します。

4 Liealgebra クラス

Lie 代数のクラスです。Lie 代数のデータを構造定数を用いて管理します。

`Liealgebra`

- `new(c) => Liealgebra`
  c を構造定数とする Lie 代数を返します。
  c のデータは c[i][j,k] の形でインプットしてください。c[i][j,k]$= c^i_{j,k}$ です。
- `size => Integer`
  self の次元を返します。
- `str_const => Array`
  self の構造定数を返します。
- `jacobian? => boolean`
  self を定義するときに構造定数として定めたテンソルが Jacobi 恒等式を満たすかどうかを判定し、Jacobi 恒等式を満たすならば true を返します。
- `liealgebra? => boolean`
  self を定義するときに構造定数として定めたテンソルが Lie bracket かどうかを判定し、Lie bracket ならば true を返します。
- `killing_form => Matrix`
  self の Killing 形式を返します。
- `semi_simple? => boolean`
  self が半単純 Lie 代数かどうかを判定し、半単純ならば true を返します。
  判定には Killing 形式の行列式を用います。
- `bracket(v,w) => Vector`
  v,w の self の bracket を用いて計算します。
  v,w は Vector か Subspace で両者は同じ型にしてください。
- `subalgebra?(b) => boolean`
  Subspace b が self の部分 Lie 代数かどうかを判定して、部分 Lie 代数ならば true を返します。
- `ideal?(b) => boolean`
  Subspace b が self のイデアルかどうかを判定して、イデアルならば true を返します。
- `derived => Subspace`
  self の交換子 Lie 代数 [self,self] を返します。
- `radical => Subspace`
  self の根基を返します。
  根基は Killing 形式に関する交換子 Lie 代数の直交補空間として計算します。
- `nilpotent_radical => Subspace`
  self の冪零根基 (Bourbaki の意味での) を返します。
  冪零根基は根基と交換子 Lie 代数の共通部分として計算します。
- `center => Subspace`
  self の中心を返します。
- `subalgebra(u) => Liealgebra`

self の部分 Lie 代数 Subspace u の bracket を計算し、Liealgebra として返します。

- quotient(u) => Liealgebra
  self のイデアル Subspace u による商 Lie 代数を返します。
- derivation => LinearLiealgebra
  self の微分 Lie 代数を返します。
- adrep => Representation
  self の随伴表現を返します。
- Liealgebra.abelian(size) => Liealgebra
  size 次元の可換 Lie 代数を返します。
- Liealgebra.heisenberg(n) => Liealgebra
  2n+1 次元の Heisenberg 群の Lie 代数を返します。
- Liealgebra.ladder(n) => Liealgebra
  $n+1$ 次元の ladder Lie algebra を返します。

5 LinearLiealgebra クラス

線型 Lie 代数のクラスです。行列の基底で線型 Lie 代数のデータを管理します。

LinearLiealgebra < Liealgebra

- new(m) => LinearLiealgebra
  正方行列の配列 m をもとに LinearLiealgebra クラスを作成します。
  m の各要素を to_vec でベクトルにした後で schmidt の直交化を用いて直交基底にしてから to_mat でもとのサイズの正方行列に戻します。
- size => Integer
  self の次元を返します。
- to_a => Array
  self を管理している基底を並べた配列を返します。
- [](n) => Matrix
  self を管理している基底を配列とみなしたときの第 n 成分を返します。
- str_const => Array
  self の構造定数を返します。
- mprint => nil
  self を管理している基底を mprint の形式で順に表示します。
- sublinliealg(b) => Lienearlielagebra
  Subspace b で表された self の線型 Lie 代数として部分空間を返します。
- to_rep => Representation

線形 Lie 代数 self の恒等写像を表現とみなします。

- *(g)
  Liealgebra g に対して self.to_rep * g を返します。
- LinearLiealgebra.gl(n)
  線型 Lie 代数 $\mathfrak{gl}(n, \mathbb{Q})$ を返します。
- LinearLiealgebra.sl(n)
  線型 Lie 代数 $\mathfrak{sl}(n, \mathbb{Q})$ を返します。
- LinearLiealgebra.o(n)
  線型 Lie 代数 $\mathfrak{o}(n, \mathbb{Q})$ を返します。
- LinearLiealgebra.heisenberg(n)
  2n+1 次元 Heisenberg 群の Lie 代数の n+2 次上三角行列による表現を返します。

6 Representation クラス

行列の表現のクラスです。

Representation

- Representation.new(g,m) => Representation
  Matrix の配列 m によって表される Liealgebra g からの表現を返します。
- domain => Liealgebra
  self が表現する Lie 代数を返します。
- size => Liealgebra
  self が表現する Lie 代数の次元を返します。
- [](key) => Matrix
  self を規定する行列の配列の第 key 成分を返します。
- to_a => Array
  self を規定する行列の配列を返します。
- image => LinearLiealgebra
  self の像の線型 Lie 代数を返します。
- to_mat => Matrix
  self を self.domain から self.image への線形写像と見た場合の行列表示を返します。
- Representation.std(g) => Representation
  Liealgebra g の自明表現を返します。
- representation? => boolean
  self が Lie 代数 self.domain の表現になっているかを判定し、なっていれば true を返します。
- mprint => nil

self を規定する表列の配列を mprint 形式で並べて表示します。

- *(h) => Liealgebra
  self を self.domain から Liealgebra h への表現とみなして、self に関する self.domain と h の半直積を返します。
- b1 => Array
  self.domain の表現 self に関する 1 次完全形式全体の基底返します。
- z1 => Array
  self.domain の表現 self に関する 1 次閉形式全体の基底を返します。
- h1 => Integer
  self.domain の表現 self に関する 1 次コホモロジーの次元を返します。