Discrepancies between a linear transformation and its matrix -- proposed fixes

Author: Greg Grunberg (GG)
Last updated: 2020-10-25

Note: Output in this notebook makes use of **gprinter.py**, an unofficial GAlgebra module written by Alan Bromborsky.

Discussion

A linear transformation/outermorphism L should have **action** $\mathsf{L}(\mathbf{e}_j) = \sum_{i=1}^{n} \mathbf{e}_i L_{ij}$ of L on the basis vector $\mathbf{e}_j$ if and only if `L` has an $n \times n$ matrix $[L_{ij}]$. Traditional mathematical notation numbers the indexes from 1 to n. Indexes in GAlgebra will range from 0 to $n-1$ or will range over the coordinate names. Given the action of L, the *Fourier formula* $L_{ij} = \mathbf{e}^i \cdot \mathsf{L}(\mathbf{e}_j)$ may be used to find the expansion coefficients. (Although the formula uses the scalar product, its end result L_{ij} does not depend on that product.)

Specific transformations are instantiated by way of a command of the form `L = GA.lt(action_list)` . The jth entry $[L_{1j}, \ldots, L_{nj}]$ in `action_list` specifies both the action $\mathsf{L}(\mathbf{e}_j)$ on the jth basis vector and the entries in the jth column of L's matrix. For a specific transformation the L_{ij}'s are specific real numbers or specific real SymPy symbols. For example, the first test below uses `action_list = [[a,c], [b,d]]` , so in that test `L` should have action $\mathsf{L} : \left\{ \begin{matrix} \mathbf{e}_x \mapsto \mathbf{e}_x a + \mathbf{e}_y c \\ \mathbf{e}_y \mapsto \mathbf{e}_x b + \mathbf{e}_y d \end{matrix} \right\}$ on basis $(\mathbf{e}_x, \mathbf{e}_y)$, and `L.matrix()` should return $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$.

Generic transformations are instantiated by a command of the form `L = GA.lt('L')` . The values of the L_{ij}'s are not specified but are left general; all that's given to the constructor `GA.lt` as an instantiation parameter is a single character, in this case `'L'` . For a generic transformation the return value of `L.matrix()[i,j]` should be a SymPy symbol which prints as L_{ij}.

The problem.

For a specific transformation `L` the action $\mathsf{L}(\mathbf{e}_j)$ is computed correctly to be $\sum_{i=1}^{n} \mathbf{e}_i L_{ij}$, but `L.matrix()` returns $[\sum_{k=1}^{n} L_{ik} g_{kj}]$ rather than the correct $[L_{ij}]$.

For generic transformations `L` the problem is just the opposite. `L.matrix()` correctly returns $[L_{ij}]$ but the action $\mathsf{L}(\mathbf{e}_j)$ incorrectly computes to be $\sum_{i=1}^{n} \mathbf{e}_i (\sum_{k=1}^{n} L_{ik} g^{kj})$.

A proposed solution.

To fix the problems I suggest modifications to two of the `Lt` class functions. First modify the `matrix(self)` method so that it reads

```
def matrix(self) -> Matrix:
    r"""
    Returns the matrix representation :math:`L_{ij}` of
    linear transformation :math:`L`, defined by
    :math:`{{L}\lp {{{\eb}}_{j}} \rp } = {\sum_i} {{\eb}}_{i} L_{ij}`.
    """
    if self.mat is not None:
        return self.mat.doit()
    else:
        if self.spinor:
            self.lt_dict = {}
            for base in self.Ga.basis:
                self.lt_dict[base] = self(base).simplify()
            self.spinor = False
            mat = self.matrix()
            self.spinor = True
            return mat
        else:
            self.mat = Dictionary_to_Matrix(self.lt_dict, self.Ga)
            return self.mat.doit()
```

The above code has four changes relative to the version of `matrix(self)` which appears in GitHub's **lt.py** module as of 2020-10-25:

1. The current docstring says the entries of `L` 's matrix are defined by $\mathsf{L}(\mathbf{e}_i) = L_{ij}\mathbf{e}_j$, with an implicit summation on the *second* index of the matrix entries. I've corrected the docstring so that it says $\mathsf{L}(\mathbf{e}_j) = \sum_i \mathbf{e}_i L_{ij}$, with an explicit summation of the *first* index of the matrix entries, an equation found in virtually all linear algebra textbooks.
2. The current penultimate code line reads `self.mat=Dictionary_to_Matrix(self.lt_dict,self.Ga) * self.Ga.g` I have eliminated the post multiplication by `self.Ga.g`.
3. The current `return self.mat` (appears twice) has been rewritten as `return self.mat.doit()`. This forces completion of any pending operations in `self.mat` before that matrix is returned.
4. The current `matrix(self)` contains code made non-functional by enclosure within triple double-quotation marks. That code has been deleted.

After the suggested changes the `.matrix()` method should work correctly for specific transformations.

I also suggest modifying the `__init__` function of the `Lt` class. Currently that function contains three code lines which read

```
elif isinstance(mat_rep, str):  # String input
    Amat = Symbolic_Matrix(mat_rep, coords=self.Ga.coords,
                mode=mode, f=f)
    self.__init__(Amat, ga=self.Ga)
```

I would change those lines to read

```
elif isinstance(mat_rep, str):  # String input
    Amat = Symbolic_Matrix(mat_rep, coords=self.Ga.coords,
                mode=mode, f=f)
    dim = len(self.Ga.mv())
    action_list = \
        [[Amat[i,j] for i in range(dim)] for j in range(dim)]
    self.__init__(action_list, ga=self.Ga)
```

The generic transformations should then instantiate correctly, with action and matrix consistent with each other.

Test proposed modifications

I have made the modifications suggested above to my **lt.py** module. The rest of this notebook checks that the modifications accomplish their purpose of bringing into accord a transformation's action and matrix.

In [1]:
```
# Initializations
from sys import version
import sympy
from sympy import *
import galgebra
from galgebra.ga import *
from galgebra.mv import *
from galgebra.printer import Fmt, GaPrinter, Format
from galgebra.gprinter import gFormat, gprint
gFormat()
Ga.dual_mode('Iinv+')
from galgebra.lt import *
gprint(r'\text{Initializations executed.}\\',
    r'\text{This notebook is now using}\\',
    r'\qquad\bullet~ \text{Python }', version[:5],
    r'\qquad\bullet~ \text{SymPy }', sympy.__version__[:7],
    r'\qquad\bullet~ \text{GAlgebra }',
       galgebra.__version__[:], r'.')
```

Initializations executed.
This notebook is now using

$\qquad\bullet$ Python 3.8.3 $\qquad\bullet$ SymPy 1.8.*dev* $\qquad\bullet$ GAlgebra 0.5.0.

In [2]:
```
def action(L):
    """Returns as a matrix the coefficients in the basis expansions
    of images of basis vectors by linear transformation `L`. Uses
    the actual action of `L` to compute the coefficients."""
    # Each entry `row` in `rows` will correspond to a row in the
    # matrix.  Reciprocal basis vector `r` determines a row in the
    # matrix, while each basis vector `c` determines a column.
    rows = []
    for r in L.Ga.mvr():
        row = []
        for c in L.Ga.mv():
            row.append((r<L(c)).scalar())
            # Fourier formula is used to calculate appended row entry.
        rows.append(row)
    return simplify(Matrix(rows))
```

The next function takes as its sole argument a linear transformation/outermorphism `L` and returns information about it and the geometric algebra `GA` on which it acts.

In [3]:
```
def lin_tfn_info(L):
    """
    - Argument `L` is an outermorphism on some geometric algebra `GA`.
    - Reports geometric algebra on which `L` acts, the metric tensor
      of that algebra, and the reciprocal metric tensor.
    - Reports `L`'s actual action, the corresponding action matrix
      `action(L)`, and the purported matrix `L.matrix()`.
    """
    gprint(r'\text{L.Ga}= \text{}' + GA_name[L.Ga] + r':\quad',
        r'[g_{ij}] = ', L.Ga.g, r',\quad', r'[g^{ij}] = ', L.Ga.g_inv)
    gprint(r'\text{L}:', L, r',\quad',
        r'\text{action(L)} = ', action(L), r',\quad',
        r'\text{L.matrix()} = ', L.matrix())
    gprint('')
    gprint('')
```

For testing purposes we will employ two representations of $\mathbb{G}(\mathbb{R}^{2,0})$ and three of $\mathbb{G}(\mathbb{R}^{1,1})$. Each representation uses coordinates (x, y) to label points in its underlying manifold. Each representation uses $(\mathbf{e}_x, \mathbf{e}_y)$ to denote a basis for the tangent space at point-of-tangency (x, y).

- `g2a` , a model of $\mathbb{G}(\mathbb{R}^{2,0})$. Metric is Euclidean. $(\mathbf{e}_x, \mathbf{e}_y)$ are *orthonormal*.
- `g2b` , a model of $\mathbb{G}(\mathbb{R}^{2,0})$. Metric is Euclidean. $(\mathbf{e}_x, \mathbf{e}_y)$ are *orthogonal* but not orthonormal.
- `g2c` , a model of $\mathbb{G}(\mathbb{R}^{1,1})$. Metric is Minkowskian. $(\mathbf{e}_x, \mathbf{e}_y)$ are *orthonormal*.
- `g2d` , a model of $\mathbb{G}(\mathbb{R}^{1,1})$. Metric is Minkowskian. $(\mathbf{e}_x, \mathbf{e}_y)$ are *null vectors*.
- `g2e` , a model of $\mathbb{G}(\mathbb{R}^{1,1})$. Metric is Minkowskian. $(\mathbf{e}_x, \mathbf{e}_y)$ are *oblique*.

In [4]:
```
a, b, c, d, x, y = symbols('a b c d x y', real=True)
g2a  = Ga('\mathbf{e}', g = [[1,0], [0,1]],     coords=(x,y))
g2b  = Ga('\mathbf{e}', g = [[1,0], [0,4]],     coords=(x,y))
g2c  = Ga('\mathbf{e}', g = [[1,0], [0,-1]],    coords=(x,y))
g2d  = Ga('\mathbf{e}', g = [[0,1], [1,0]],     coords=(x,y))
g2e  = Ga('\mathbf{e}', g = [[0,-1], [-1,-1]], coords=(x,y))
GAs = [g2a, g2b, g2c, g2d, g2e]
GA_name = {g2a:'g2a', g2b:'g2b', g2c:'g2c', g2d:'g2d', g2e:'g2e'}
```

Test of specific transformations.
In the next cell, for each of the five geometric algebras in `GAs` , a specific linear transformation/outermorphism `L` is instantiated by a command of the form `L = GA.lt([[a,c], [b,d]])` . Therefore `L` should have action $\left\{\begin{matrix} \mathbf{e}_x \mapsto \mathbf{e}_x a + \mathbf{e}_y c \\ \mathbf{e}_y \mapsto \mathbf{e}_x b + \mathbf{e}_y d \end{matrix}\right\}$ on basis $(\mathbf{e}_x, \mathbf{e}_y)$, and it should have matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ with respect to that basis. Equality of the matrices `action(L)` and `L.matrix()` is what's desired.

In [5]:
```
for GA in GAs:
    L = GA.lt([[a, c], [b, d]])
    lin_tfn_info(L)
```

$$\mathrm{L.Ga} = g2a: \quad [g_{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad [g^{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathrm{L}: \left\{ \begin{array}{l} \mathbf{e}_x \mapsto a\mathbf{e}_x + c\mathbf{e}_y \\ \mathbf{e}_y \mapsto b\mathbf{e}_x + d\mathbf{e}_y \end{array} \right\}, \quad \mathrm{action(L)} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \mathrm{L.matrix()} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\mathrm{L.Ga} = g2b: \quad [g_{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}, \quad [g^{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{4} \end{bmatrix}$$

$$\mathrm{L}: \left\{ \begin{array}{l} \mathbf{e}_x \mapsto a\mathbf{e}_x + c\mathbf{e}_y \\ \mathbf{e}_y \mapsto b\mathbf{e}_x + d\mathbf{e}_y \end{array} \right\}, \quad \mathrm{action(L)} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \mathrm{L.matrix()} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\mathrm{L.Ga} = g2c: \quad [g_{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad [g^{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\mathrm{L}: \left\{ \begin{array}{l} \mathbf{e}_x \mapsto a\mathbf{e}_x + c\mathbf{e}_y \\ \mathbf{e}_y \mapsto b\mathbf{e}_x + d\mathbf{e}_y \end{array} \right\}, \quad \mathrm{action(L)} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \mathrm{L.matrix()} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\mathrm{L.Ga} = g2d: \quad [g_{ij}] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad [g^{ij}] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\mathrm{L}: \left\{ \begin{array}{l} \mathbf{e}_x \mapsto a\mathbf{e}_x + c\mathbf{e}_y \\ \mathbf{e}_y \mapsto b\mathbf{e}_x + d\mathbf{e}_y \end{array} \right\}, \quad \mathrm{action(L)} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \mathrm{L.matrix()} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\mathrm{L.Ga} = g2e: \quad [g_{ij}] = \begin{bmatrix} 0 & -1 \\ -1 & -1 \end{bmatrix}, \quad [g^{ij}] = \begin{bmatrix} 1 & -1 \\ -1 & 0 \end{bmatrix}$$

$$\mathrm{L}: \left\{ \begin{array}{l} \mathbf{e}_x \mapsto a\mathbf{e}_x + c\mathbf{e}_y \\ \mathbf{e}_y \mapsto b\mathbf{e}_x + d\mathbf{e}_y \end{array} \right\}, \quad \mathrm{action(L)} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \mathrm{L.matrix()} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Success!

Test of *generic* transformations.

We now test generic transformations instantiated through a command of the form `L = GA.lt('L')` . Such a transformation should have action $\left\{ \begin{array}{l} \mathbf{e}_x \mapsto \mathbf{e}_x L_{xx} + \mathbf{e}_y L_{yx} \\ \mathbf{e}_y \mapsto \mathbf{e}_x L_{xy} + \mathbf{e}_y L_{yy} \end{array} \right\}$ on basis $(\mathbf{e}_x, \mathbf{e}_y)$ and matrix $\begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix}$ with respect to that basis.

In [6]:
```
for GA in GAs:
    L = GA.lt('L')
    lin_tfn_info(L)
```

$$\text{L.Ga} = g2a: \quad [g_{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad [g^{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\text{L}: \left\{ \begin{array}{l} \mathbf{e}_x \mapsto L_{xx}\mathbf{e}_x + L_{yx}\mathbf{e}_y \\ \mathbf{e}_y \mapsto L_{xy}\mathbf{e}_x + L_{yy}\mathbf{e}_y \end{array} \right\}, \quad \text{action(L)} = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix}, \quad \text{L.matrix()} = \begin{bmatrix} L_{xx} \\ L_{yx} \end{bmatrix}$$

$$\text{L.Ga} = g2b: \quad [g_{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}, \quad [g^{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{4} \end{bmatrix}$$

$$\text{L}: \left\{ \begin{array}{l} \mathbf{e}_x \mapsto L_{xx}\mathbf{e}_x + L_{yx}\mathbf{e}_y \\ \mathbf{e}_y \mapsto L_{xy}\mathbf{e}_x + L_{yy}\mathbf{e}_y \end{array} \right\}, \quad \text{action(L)} = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix}, \quad \text{L.matrix()} = \begin{bmatrix} L_{xx} \\ L_{yx} \end{bmatrix}$$

$$\text{L.Ga} = g2c: \quad [g_{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad [g^{ij}] = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\text{L}: \left\{ \begin{array}{l} \mathbf{e}_x \mapsto L_{xx}\mathbf{e}_x + L_{yx}\mathbf{e}_y \\ \mathbf{e}_y \mapsto L_{xy}\mathbf{e}_x + L_{yy}\mathbf{e}_y \end{array} \right\}, \quad \text{action(L)} = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix}, \quad \text{L.matrix()} = \begin{bmatrix} L_{xx} \\ L_{yx} \end{bmatrix}$$

$$\text{L.Ga} = g2d: \quad [g_{ij}] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad [g^{ij}] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\text{L}: \left\{ \begin{array}{l} \mathbf{e}_x \mapsto L_{xx}\mathbf{e}_x + L_{yx}\mathbf{e}_y \\ \mathbf{e}_y \mapsto L_{xy}\mathbf{e}_x + L_{yy}\mathbf{e}_y \end{array} \right\}, \quad \text{action(L)} = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix}, \quad \text{L.matrix()} = \begin{bmatrix} L_{xx} \\ L_{yx} \end{bmatrix}$$

$$\text{L.Ga} = g2e: \quad [g_{ij}] = \begin{bmatrix} 0 & -1 \\ -1 & -1 \end{bmatrix}, \quad [g^{ij}] = \begin{bmatrix} 1 & -1 \\ -1 & 0 \end{bmatrix}$$

$$\text{L}: \left\{ \begin{array}{l} \mathbf{e}_x \mapsto L_{xx}\mathbf{e}_x + L_{yx}\mathbf{e}_y \\ \mathbf{e}_y \mapsto L_{xy}\mathbf{e}_x + L_{yy}\mathbf{e}_y \end{array} \right\}, \quad \text{action(L)} = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix}, \quad \text{L.matrix()} = \begin{bmatrix} L_{xx} \\ L_{yx} \end{bmatrix}$$

Success! again.

I have not examined other instantiation circumstances. According to the documentation, `GA.lt(mat_rep)` should result in a linear transformation when `mat_rep` is a dictionary, a list of lists, a matrix, a spinor, or a character. (It's the circumstances "list of lists" and "character" which I called a "specific" transformation and "generic" transformation, respectively.)