Noncommutative and nonassociative three velocity in special relativity: introducing the lorentz package

Robin K. S. Hankin
Auckland University of Technology

Abstract

Here I present the **lorentz** package for working with the Lorentz group for relativistic physics. The package includes functionality for relativistic velocity addition, which is noncommutative and nonassociative.

Keywords: Lorentz transform, Lorentz group, Lorentz law, Lorentz velocity addition, special relativity, relativistic physics, Einstein velocity addition, Wigner rotation, gyrogroup, gyromorphism, gyrocommutative, gyroassociative, four velocity, three-velocity, nonassociative, noncommutative.

1. Introduction

In special relativity, the Lorentz transformations supercede their classical equivalent, the Galilean transforms. Lorentz transforms are typically expressed in terms of 4×4 matrices that take one set of coordinates to another set which is moving at constant velocity with respect to the first.

The **lorentz** package provides functionality for dealing with Lorentz transformations. It deals with Lorentz boosts, converts between three-velocities and four-velocities, and provides computational support for the gyrogroup structure of three-velocities under successive addition.

2. Lorentz transforms

Consider the following three-velocity:

```
> u <- as.3vel(c(0.3,-0.4,0.8))
> u
```

```
       x    y   z
[1,] 0.3 -0.4 0.8
```

(note that the package uses units in which $c = 1$). Velocity u may be expressed as a four-velocity:

```
> as.4vel(u)
```

```
           t        x         y        z
[1,] 3.015113 0.904534 -1.206045 2.412091
```

The corresponding coordinate transformation is given by `boost()`:

```
> boost(u)
```

```
          t          x          y          z
t  3.015113 -0.9045340  1.2060454 -2.4120908
x -0.904534  1.2037755 -0.2717007  0.5434014
y  1.206045 -0.2717007  1.3622676 -0.7245352
z -2.412091  0.5434014 -0.7245352  2.4490703
```

And then an arbitrary four-vector may be expressed in the boosted coordinates:

```
> boost(u) %*% c(4,5,-2,3)
```

```
       [,1]
t -2.110579
x  4.574347
y -1.432463
z  1.864925
```

Note that coordinate transformation is effected by standard matrix multiplication. Composition of two Lorentz transforms is also ordinary matrix multiplication:

```
> v <- as.3vel(c(0.4,0.2,-0.1))
> L <- boost(u) %*% boost(v)
> L
```

```
          t          x          y          z
t  3.256577 -2.2327055  0.5419596 -2.0800479
x -1.437147  1.6996791 -0.0237489  0.4194255
y  1.091131 -0.7581795  1.1190282 -0.6029155
z -2.519789  1.5878378 -0.2023170  2.1879612
```

But observe that the resulting transform is not a pure boost, as the matrix is not symmetrical. We may decompose the matrix into a pure translation composed with an orthogonal matrix, which represents a coordinate rotation:

```
> U <- orthog(L)
> P <- pureboost(L)
```

In the above, U should be orthogonal and L symmetric:

```
> crossprod(U) - diag(4)
```

```
              [,1]          [,2]          [,3]          [,4]
[1,] -3.907985e-14 -3.010826e-14  7.711957e-15 -3.141176e-14
[2,] -3.010826e-14 -2.575717e-14  6.314393e-15 -2.325917e-14
[3,]  7.711957e-15  6.314393e-15 -1.554312e-15  5.384582e-15
[4,] -3.141176e-14 -2.325917e-14  5.384582e-15 -2.187139e-14
```

```
> P - t(P)
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0
```

(that is, zero to numerical precision).

3. Three velocities

The **lorentz** package includes functionality to compose three-velocities. Three velocities do not form a group under composition as the velocity addition law is not associative. As Ungar (2006) points out: "The nonassociativity of Einstein's velocity addition is not widely known". Ungar shows that the velocity addition law is

$$\mathbf{u}\oplus\mathbf{v}=\frac{1}{1+\mathbf{u}\cdot\mathbf{v}}\left\{\mathbf{u}+\frac{\mathbf{v}}{\gamma_{\mathbf{u}}}+\frac{\gamma_{\mathbf{u}}\left(\mathbf{u}\cdot\mathbf{v}\right)\mathbf{u}}{1+\gamma_{\mathbf{u}}}\right\} \tag{1}$$

where $\gamma_{\mathbf{u}}=\left(1-\mathbf{u}\cdot\mathbf{u}\right)^{-1/2}$ and we are assuming $c=1$. Ungar shows that, in general, $\mathbf{u}\oplus\mathbf{v}\neq\mathbf{v}\oplus\mathbf{u}$ and $(\mathbf{u}\oplus\mathbf{v})\oplus\mathbf{w}\neq\mathbf{u}\oplus(\mathbf{v}\oplus\mathbf{w})$. He also defines the binary operator $\ominus$ as $\mathbf{u}\ominus\mathbf{v}=\mathbf{u}\oplus(-\mathbf{v})$, and implicitly defines $\ominus\mathbf{u}\oplus\mathbf{v}$ to be $(-\mathbf{u})\oplus\mathbf{v}$). If we have

$$\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{x}=-\left(\mathbf{u}\oplus\mathbf{v}\right)\oplus\left(\mathbf{u}\oplus\left(\mathbf{v}\oplus\mathbf{x}\right)\right) \tag{2}$$

Then Ungar shows that

$$\begin{aligned}
\mathbf{u}\oplus\mathbf{v} &= \mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right](\mathbf{v}\oplus\mathbf{u}) && (3)\\
\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{x}\cdot\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{y} &= \mathbf{x}\cdot\mathbf{y} && (4)\\
\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right](\mathbf{x}\oplus\mathbf{y}) &= \mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{x}\oplus\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{y} && (5)\\
\left(\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\right)^{-1} &= \left(\mathrm{gyr}\left[\mathbf{v},\mathbf{u}\right]\right) && (6)\\
\mathbf{u}\oplus(\mathbf{v}\oplus\mathbf{w}) &= (\mathbf{u}\oplus\mathbf{v})\oplus\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{w} && (7)\\
(\mathbf{u}\oplus\mathbf{v})\oplus\mathbf{w} &= \mathbf{u}\oplus(\mathbf{v}\oplus\mathrm{gyr}\left[\mathbf{v},\mathbf{u}\right]\mathbf{w}) && (8)
\end{aligned}$$

Consider the following R session:

```
> library(lorentz)
> u <- as.3vel(c(-0.7,+0.2,-0.3))
> v <- as.3vel(c(+0.3,+0.3,+0.4))
> w <- as.3vel(c(+0.1,+0.3,+0.8))
> x <- as.3vel(c(-0.2,-0.1,-0.9))
> u

        x   y    z
[1,] -0.7 0.2 -0.3
```

Here we have three-vectors `u` etc. We can see that `u` and `v` do not commute:

```
> u+v

              x         y            z
[1,] -0.5454028 0.4815421 -0.004538856

> v+u

              x         y         z
[1,] -0.4292804 0.5723133 0.1324513
```

(the results differ). We can use equation 3

```
> (u+v)-gyr(u,v,v+u)

                x             y            z
[1,] 1.769252e-16 -7.077008e-16 1.230183e-16
```

showing agreement to within numerical error. It is also possible to use the functional idiom:

```
> f <- gyrfun(u,v)
> (u+v)-f(v+u)    # should be zero

                x             y            z
[1,] 1.769252e-16 -7.077008e-16 1.230183e-16
```

Function `gyrfun()` is vectorized, which means that it plays nicely with (R) vectors. Consider

```
> u9 <- r3vel(9)
> u9

               x           y          z
 [1,]  0.4394266 -0.43180089 -0.3808371
 [2,]  0.2045785 -0.44122304  0.7810400
 [3,]  0.2044740  0.66984515 -0.5631968
```

```
 [4,]  0.4870942 -0.42749684  0.6121989
 [5,] -0.0711684  0.05808238 -0.4440446
 [6,] -0.5772239  0.51646405  0.3209645
 [7,] -0.6791358  0.62645363  0.2250279
 [8,]  0.7348901 -0.42616435  0.1924509
 [9,] -0.3746674 -0.41068270 -0.5442920
```

Then we can create a vectorized gyrofunction:

```
> f <- gyrfun(u9,v)
> f(x)
```

```
               x           y          z
 [1,] -0.38380965 -0.01258070 -0.8441160
 [2,] -0.09431017  0.18863324 -0.9030632
 [3,] -0.33144718 -0.40339988 -0.7664276
 [4,] -0.20147532  0.15740913 -0.8914202
 [5,] -0.25368274 -0.19078619 -0.8713471
 [6,]  0.03556765 -0.18704146 -0.9076070
 [7,]  0.10356937 -0.24913239 -0.8872465
 [8,] -0.34367400  0.10535420 -0.8548618
 [9,] -0.21051456 -0.09751487 -0.8978722
```

Note that the package vectorization is transparent when using syntatic sugar:

```
> u9+x
```

```
               x           y           z
 [1,]  0.29201211 -0.44445652 -0.82321642
 [2,]  0.07785034 -0.86108127  0.11922266
 [3,]  0.12400469  0.58026114 -0.79634410
 [4,]  0.49058077 -0.74705469 -0.12849652
 [5,] -0.18804552 -0.01314855 -0.95308570
 [6,] -0.77703005  0.49904608 -0.27914722
 [7,] -0.77403178  0.61291115 -0.08684227
 [8,]  0.69191125 -0.54809308 -0.39513229
 [9,] -0.39837145 -0.39002104 -0.81716304
```

(here, the addition operates using R's standard recycling rules).

3.1. Associativity

Three velocity addition is not associative:

```
> (u+v)+w
```

```
             x        y         z
[1,] -0.4646213 0.655437 0.5008326
```

```
> u+(v+w)
```

```
              x         y         z
[1,] -0.5489863 0.6672455 0.4160947
```

But we can use equations 7 and 8:

```
> (u+(v+w)) - ((u+v)+gyr(u,v,w))
```

```
                 x             y             z
[1,] 6.916191e-16 -1.383238e-15 -6.916191e-16
```

```
> ((u+v)+w) - (u+(v+gyr(v,u,w)))
```

```
     x y            z
[1,] 0 0 5.353251e-16
```

4. Visualization

Consider the following three-velocities:

```
> u <- as.3vel(c(0.4,0,0))
> v <- seq(as.3vel(c(0.4,-0.2,0)), as.3vel(c(-0.3,0.9,0)),len=20)
> w <- as.3vel(c(0.8,-0.4,0))
```

Objects **v** and **w** are single three-velocities, and object **v** is a vector of three velocities. We can see the noncommutativity of three velocity addition in figures 1 and 2, and the nonassociativity in figures 3.

```
> comm_fail1(u=u, v=v)
```

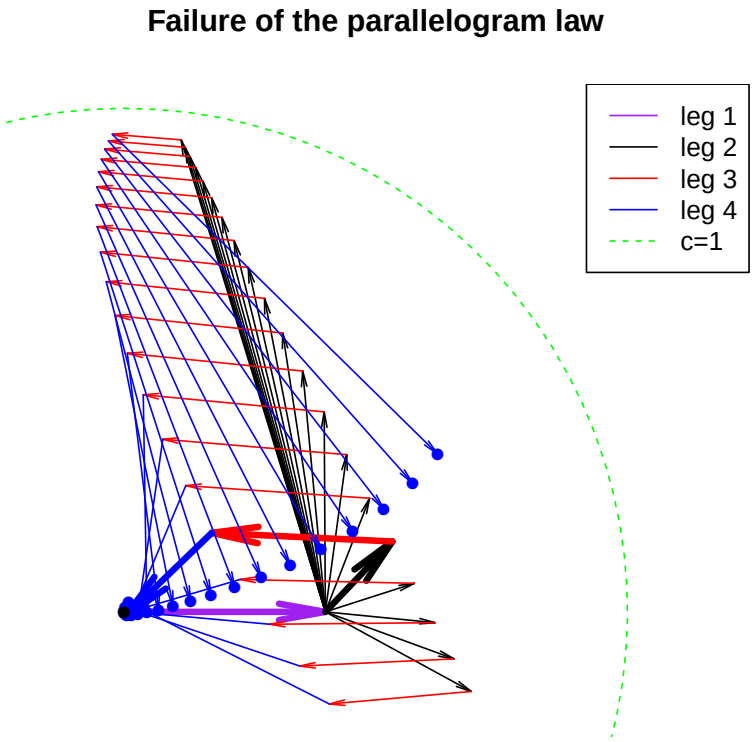

Figure 1: Failure of the commutative law for velocity composition in special relativity. The arrows show successive velocity boosts of $+\mathbf{u}$ (purple), $+\mathbf{v}$ (black), $-\mathbf{u}$ (red), and $-\mathbf{v}$ (blue) for $\mathbf{u}, \mathbf{v}$ as defined above. Velocity $\mathbf{u}$ is constant, while $\mathbf{v}$ takes a sequence of values. If velocity addition is commutative, the four boosts form a closed quadrilateral; the thick arrows show a case where the boosts almost close and the boosts nearly form a parallelogram. The blue dots show the final velocity after four successive boosts; the distance of the blue dot from the origin measures the combined velocity, equal to zero in the classical limit of low speeds. The discrepancy becomes larger and larger for the faster elements of the sequence $\mathbf{v}$

```
> comm_fail2(u=u, v=v)
```

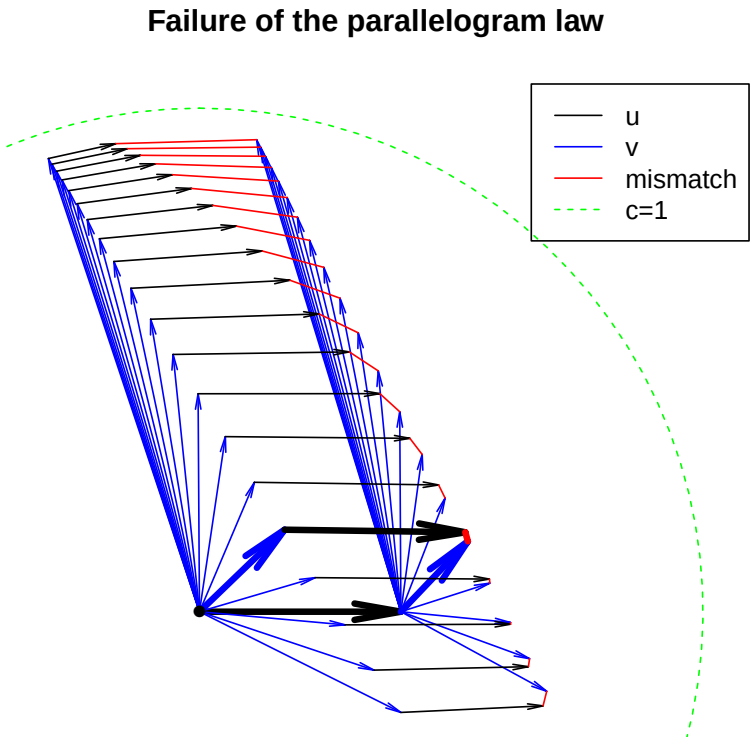

Figure 2: Another view of the failure of the commutative law in special relativity. The black arrows show velocity boosts of $\mathbf{u}$ and the blue arrows show velocity boosts of $\mathbf{v}$, with $\mathbf{u},\mathbf{v}$ as defined above; $\mathbf{u}$ is constant while $\mathbf{v}$ takes a sequence of values. If velocity addition is commutative, then $\mathbf{u}+\mathbf{v}=\mathbf{v}+\mathbf{u}$ and the two paths end at the same point: the parallelogram is closed. The red arrows show the difference between $\mathbf{u}+\mathbf{v}$ and $\mathbf{v}+\mathbf{u}$

```
> ass_fail(u=u, v=v, w=w, bold=10)
```

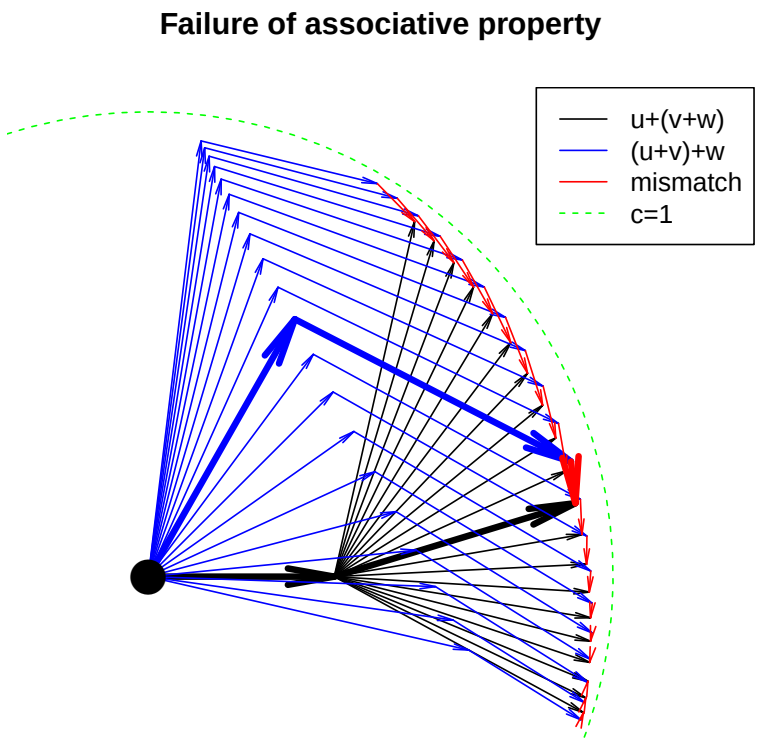

Figure 3: Failure of the associative law for velocity composition in special relativity. The arrows show successive boosts of $\mathbf{u}$ followed by $\mathbf{v}+\mathbf{w}$ (black lines), and $\mathbf{u}+\mathbf{v}$ followed by $\mathbf{w}$ (blue lines), for $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ as defined above; $\mathbf{u}$ and $\mathbf{w}$ are constant while $\mathbf{v}$ takes a sequence of values. The mismatch between $\mathbf{u}+(\mathbf{v}+\mathbf{w})$ and $(\mathbf{u}+\mathbf{v})+\mathbf{w}$ is shown in red

5. The magrittr package: pipes

The lorentz package works nicely with magrittr. If we define

```
>  u <- as.3vel(c(+0.5,0.1,-0.2))
>  v <- as.3vel(c(+0.4,0.3,-0.2))
>  w <- as.3vel(c(-0.3,0.2,+0.2))
```

Then pipe notation operates as expected:

```
> jj1 <- u %>% add(v)
> jj2 <- u+v
> speed(jj1-jj2)

[1] 2.206363e-16
```

The pipe operator is left associative:

```
> jj1 <- u %>% add(v) %>% add(w)
> jj2 <- (u+v)+w
> speed(jj1-jj2)

[1] 7.392965e-17
```

If we want right associative addition, the pipe operator needs brackets:

```
> jj1 <- u %>% add(v %>% add(w))
> jj2 <- u+(v+w)
> speed(jj1-jj2)

[1] 3.446154e-17
```

6. Functional notation

It is possible to replace calls like `gyr(u,v,x)` with functional notation which can make for arguably more natural R idiom. If we have

```
> u <- as.3vel(c(0, 0.8, 0))
> v <- r3vel(5,0.9)
> x <- as.3vel(c(0.7, 0, -0.7))
> y <- as.3vel(c(0.1, 0.3, -0.6))
```

Then we can define `f()` which is the map $\mathbf{x} \longrightarrow \mathrm{gyr}\left[\mathbf{u}, \mathbf{v}\right]\mathbf{x}$:

```
> f <- gyrfun(u,v)
> f(w)
```

```
              x           y         z
[1,] -0.3153498  0.13462804 0.2289755
[2,] -0.1492566  0.36419920 0.1228063
[3,] -0.3309149 -0.04045552 0.2426080
[4,] -0.1575003  0.35944952 0.1264504
[5,] -0.3623831  0.14184215 0.1362326
```

Then numerical verification of equation 4 and 6 is straightforward:

```
> prod3(f(x),f(y)) - prod3(x,y)
```

```
[1] -6.106227e-16  1.110223e-16 -1.554312e-15 -7.771561e-16  2.775558e-16
```

and

```
> f <- gyrfun(u,v)
> g <- gyrfun(v,u)
> f(g(x)) - g(f(x))
```

```
                 x             y             z
[1,]  3.885781e-13  0.000000e+00 -3.885781e-13
[2,]  0.000000e+00 -2.579413e-15 -5.551115e-15
[3,] -9.992007e-14 -2.052550e-14  1.054712e-13
[4,]  3.330669e-14  1.448600e-14 -2.775558e-14
[5,] -1.110223e-14  2.765243e-15  1.110223e-14
```

(zero to numerical precision). It is possible to use pipes together with the functional notation:

```
> x %<>% f %>% add(y)      # x <- f(x)+y
> x
```

```
             x           y          z
[1,] 0.6709349  0.20339750 -0.7098525
[2,] 0.5573147 -0.47167034 -0.6784643
[3,] 0.5702050  0.58313783 -0.5747859
[4,] 0.5678568 -0.45282492 -0.6826258
[5,] 0.6819940  0.02342166 -0.7276382
```

7. SI units

The preceding material used units in which $c = 1$. Here I show how the package deals with units such as SI in which $c = 299792458 \neq 1$. For obvious reasons we cannot have a function called `c()` so the package gets and sets the package with `sol()`:

```
> sol(299792458)

[1] 299792458

> sol()

[1] 299792458
```

(an empty argument queries the speed of light). We can now consider speeds which are fast by terrestrial standards but involve only a small relativistic correction to the Galilean result:

```
> u <- as.3vel(c(100,200,300))
> u

       x   y   z
[1,] 100 200 300
```

The gamma correction term γ is only very slightly larger than 1 and indeed R's default print method suppresses the difference:

```
> gam(u)

[1] 1
```

However, we can display more significant figures by subtracting one:

```
> gam(u)-1

[1] 7.789325e-13
```

or alternatively we can use the `gamm1()` function which calculates $\gamma - 1$ more accurately for speeds $\ll c$:

```
> gamm1(u)

[1] 7.78855e-13
```

The Lorentz boost is again calculated by the `boost()` function

```
> boost(u)

     t             x            y             z
t    1 -1.112650e-15 -2.22530e-15 -3.337950e-15
x -100  1.000000e+00  1.11265e-13  1.668975e-13
y -200  1.112650e-13  1.00000e+00  3.337950e-13
z -300  1.668975e-13  3.33795e-13  1.000000e+00
```

note how the transformation is essentially the Galilean result.

```
> v <- as.3vel(c(400,-200,300))
> boost(u) %*% boost(v)

              t             x             y             z
t  1.000000e+00 -5.563250e-15  4.704692e-27 -6.675900e-15
x -5.000000e+02  1.000000e+00 -5.563250e-13  1.168283e-12
y -2.559179e-10  5.563250e-13  1.000000e+00  6.675900e-13
z -6.000000e+02  2.169668e-12 -6.675900e-13  1.000000e+00

> boost(v) %*% boost(u)

              t             x             y             z
t  1.000000e+00 -5.563250e-15 -2.847319e-27 -6.675900e-15
x -5.000000e+02  1.000000e+00  5.563250e-13  2.169668e-12
y  4.228102e-10 -5.563250e-13  1.000000e+00 -6.675900e-13
z -6.000000e+02  1.168283e-12  6.675900e-13  1.000000e+00
```

References

Ungar AA (2006). "Thomas precession: a kinematic effect of the algebra of Einstein's velocity addition law. Comments on 'Deriving relativistic momentum and energy: II. Three-dimensional case'." *European Journal of Physics*, **27**, L17–L20.

Affiliation:

Robin K. S. Hankin
Auckland University of Technology
E-mail: hankin.robin@gmail.com