Esercitazione 1

Elettrodinamica relativistica

A.A. 2023/2024

Simone Iovine

Indice

1 Esercizio 1

Riscriviamo la definizione della rapidità

$$y \doteq \frac{1}{2}\ln\left(\frac{E+cp_z}{E-cp_z}\right) = \frac{1}{2}\ln\left(\frac{1+\dfrac{v_z}{c}}{1-\dfrac{v_z}{c}}\right) \tag{1}$$

nel seguente modo

$$e^{2y} = \frac{E+cp_z}{E-cp_z} = \frac{1+\dfrac{v_z}{c}}{1-\dfrac{v_z}{c}} \tag{2}$$

definendo

$$\beta \doteq \frac{u}{c} \tag{3}$$

dove u è la velocità del boost, abbiamo anche

$$e^{2\eta} = \frac{1+\beta}{1-\beta} \tag{4}$$

Considerando il 4-momento

$$\vec{\mathbf{p}} = (E, c\vec{\mathbf{p}}) = (E, cp_x, cp_y, cp_z) \tag{5}$$

un boost lungo z può essere calcolato tramite la matrice

$$\Lambda = \begin{bmatrix} \gamma & 0 & 0 & -\beta\gamma \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\beta\gamma & 0 & 0 & \gamma \end{bmatrix} \tag{6}$$

dunque

$$\vec{\mathbf{p}}' = \Lambda\vec{\mathbf{p}} \tag{7}$$

Le componenti che hanno subito un cambiamento saranno

$$\begin{cases} E' = \gamma(E - \beta c p_z) \\ c p_z' = \gamma(c p_z - \beta E) \end{cases} \tag{8}$$

Siccome i fasci sono incidenti, operiamo il cambiamento di segno $\beta \to -\beta$, da cui

$$\begin{cases} E' = \gamma(E + \beta c p_z) \\ c p_z' = \gamma(c p_z + \beta E) \end{cases} \tag{9}$$

A questo punto consideriamo la rapidità nel sistema di riferimento primato

$$e^{2y'} = \frac{E' + c p_z'}{E' - c p_z'} \tag{10a}$$

$$\tag{10b}$$

$$= \frac{\gamma(E + \beta c p_z) + \gamma(c p_z + \beta E)}{\gamma(E + \beta c p_z) - \gamma(c p_z + \beta E)} \tag{10c}$$

$$\tag{10d}$$

$$= \frac{E + \beta c p_z + c p_z + \beta E}{E + \beta c p_z - c p_z - \beta E} \tag{10e}$$

$$\tag{10f}$$

$$= \frac{E + \beta c p_z + \beta(E + c p_z)}{E - \beta c p_z - \beta(E - c p_z)} \tag{10g}$$

$$\tag{10h}$$

$$= \underbrace{\frac{E + c p_z}{E - c p_z}}_{e^{2y}} \underbrace{\frac{1 + \beta}{1 - \beta}}_{e^{2\eta}} \tag{10i}$$

$$= e^{2(y+\eta)} \tag{10j}$$

$$e^{2y'} = e^{2y'} e^{2\eta} \implies y' = y + \eta \tag{11}$$

Analogamente, si può trovare la stessa soluzione usando la seconda definizione della rapidità. Consideriamo ancora il boost in z della 4-posizione

$$\begin{cases} \vec{\mathbf{x}} = (ct, \vec{\mathbf{x}}) \\ \vec{\mathbf{x}}' = \Lambda \vec{\mathbf{x}} \end{cases} \tag{12}$$

$$\begin{cases} ct' = \gamma(ct - \beta z) \\ z' = \gamma(z - \beta ct) \end{cases} \tag{13}$$

da questi possiamo derivare la velocità nel sistema di riferimento primato

$$\frac{v'_z}{c} = \frac{z'}{ct'} = \frac{z - \beta ct}{ct - \beta z} = \frac{ct}{ct} \frac{\frac{z}{ct} - \beta}{1 - \beta \frac{z}{ct}} = \frac{\frac{v_z}{c} - \beta}{1 - \beta \frac{v_z}{c}} \tag{14}$$

Ancora una volta, siccome i fasci sono incidenti, operiamo il cambiamento di segno $\beta \to -\beta$, da cui

$$\frac{v'_z}{c} = \frac{\frac{v_z}{c} + \beta}{1 + \beta \frac{v_z}{c}} \tag{15}$$

A questo punto sostituiamo nella definizione della rapidità

$$e^{2y'} = \frac{1 + \dfrac{v'_z}{c}}{1 - \dfrac{v'_z}{c}} \tag{16a}$$

$$= \frac{1 + \dfrac{\dfrac{v_z}{c} + \beta}{1 + \beta\dfrac{v_z}{c}}}{1 - \dfrac{\dfrac{v_z}{c} + \beta}{1 + \beta\dfrac{v_z}{c}}} \tag{16b}$$

$$= \frac{1 + \beta\dfrac{v_z}{c} + \dfrac{v_z}{c} + \beta}{1 + \beta\dfrac{v_z}{c} - \dfrac{v_z}{c} - \beta} \tag{16c}$$

$$= \frac{1 + \beta + \dfrac{v_z}{c}(1+\beta)}{1 - \beta - \dfrac{v_z}{c}(1-\beta)} \tag{16d}$$

$$= \underbrace{\frac{1 + \dfrac{v_z}{c}}{1 - \dfrac{v_z}{c}}}_{e^{2y}} \underbrace{\frac{1+\beta}{1-\beta}}_{e^{2\eta}} \tag{16e}$$

$$= e^{2(y+\eta)} \tag{16f}$$

$$e^{2y'} = e^{2y'}e^{2\eta} \implies y' = y + \eta \tag{17}$$

2 Esercizio 2

Riscriviamo la formula per la sezione d'urto nel seguente modo

$$\sigma = \frac{\mathrm{d}\nu}{\mathrm{d}V\,\mathrm{d}t}\,\frac{1}{v_{rel}\,n_1\,n_2} \tag{18}$$

Tutte le quantità riportate in questa formula sono misurate nel sistema di riferimento del laboratorio, il quale è solidale anche al bersaglio di densità n_2.
Sfruttando i fenomeni di dilatazione del tempo e contrazione delle lunghezze, possiamo riscrivere le quantità riportate nella formula nel sistema di riferimento solidale al fascio incidente di densità n_1:

$$\begin{cases} \mathrm{d}\nu' = \mathrm{d}\nu \\ \mathrm{d}V' = \dfrac{1}{\gamma}\,\mathrm{d}V \\ \mathrm{d}t' = \gamma\,\mathrm{d}t \\ n_1' = \dfrac{1}{\gamma}\,n_1 \\ n_2' = \gamma\,n_2 \end{cases} \tag{19}$$

questo perché:

- il numero di urti $\mathrm{d}\nu$ rimane costante indipendentemente dal sistema di riferimento considerato,
- il volume considerato dove si contano gli urti appare contratto per un osservatore solidale con il fascio in movimento a causa della velocità relativa,
- per lo stesso motivo, l'intervallo di tempo di misura appare dilatato,
- la densità di particelle del fascio incidente diminuisce in quanto il volume in cui questa viene calcolata è più grande rispetto al volume misurato nel sistema di riferimento solidale al laboratorio (il quale appare contratto a causa del movimento del fascio incidente rispetto al laboratorio),
- la densità di particelle del bersaglio invece aumenta per lo stesso motivo.

A questo punto, sostituendo le quantità calcolate nel sistema di riferimento solidale con il fascio incidente nella formula per la sezione d'urto calcolata in questo sistema, otteniamo

$$\sigma' = \frac{\mathrm{d}\nu'}{\mathrm{d}V'\,\mathrm{d}t'}\,\frac{1}{v_{rel}\,n_1'\,n_2'} = \frac{\mathrm{d}\nu}{\dfrac{1}{\gamma}\,\mathrm{d}V\,\gamma\,\mathrm{d}t'}\,\frac{1}{v_{rel}\,\dfrac{1}{\gamma}\,n_1\,\gamma\,n_2'} = \sigma \tag{20}$$

3 Esercizio 3

Equazione del moto

Consideriamo la seconda legge di Newton

$$\vec{\mathbf{F}} = \frac{\mathrm{d}\vec{\mathbf{p}}}{\mathrm{d}t} \tag{21}$$

dove il momento relativistico è pari a

$$\vec{\mathbf{p}} = m\gamma\vec{\mathbf{v}} \tag{22}$$

Prendendo solo i moduli dei vettori considerati e ricordando che la forza considerata è costante, integriamo la seconda legge di Newton

$$\int_0^t \frac{\mathrm{d}(m\gamma v)}{\mathrm{d}\xi}\,\mathrm{d}\xi = \int_0^t F\,\mathrm{d}\xi \tag{23a}$$

$$\frac{mv}{\sqrt{1-\dfrac{v^2}{c^2}}} = Ft \tag{23b}$$

$$\frac{m^2v^2}{1-\dfrac{v^2}{c^2}} = F^2t^2 \tag{23c}$$

$$\frac{m^2c^2}{c^2-v^2}\,v^2 = F^2t^2 \tag{23d}$$

$$\frac{m^2c^2}{F^2t^2}\,v^2 = c^2 - v^2 \tag{23e}$$

$$\left(\frac{m^2c^2}{F^2t^2}+1\right)v^2 = c^2 \tag{23f}$$

$$v = \frac{c}{\sqrt{1+\dfrac{m^2c^2}{F^2t^2}}} \tag{23g}$$

$$= \frac{c}{\sqrt{\dfrac{F^2t^2+m^2c^2}{F^2t^2}}} \tag{23h}$$

$$= \frac{cFt}{\sqrt{F^2t^2+m^2c^2}} \tag{23i}$$

$$= \frac{c}{mc}\frac{Ft}{\sqrt{1+\dfrac{F^2t^2}{m^2c^2}}} \tag{23j}$$

$$= \frac{Ft}{m}\left(1+\frac{F^2t^2}{m^2c^2}\right)^{-1/2} \tag{23k}$$

Integrando ancora la velocità e ponendo $x(t=0)=0$,

$$\int_0^t v(\xi)\,\mathrm{d}\xi = \frac{F}{m}\int_0^t \xi\left(1+\frac{F^2\xi^2}{m^2c^2}\right)^{-1/2}\mathrm{d}\xi \tag{24a}$$

$$x = \frac{F}{m}\left[\frac{m^2c^2}{F^2}\left(1+\frac{F^2\xi^2}{m^2c^2}\right)^{1/2}\right]_0^t \tag{24b}$$

$$= \frac{mc^2}{F}\left(1+\frac{F^2t^2}{m^2c^2}\right)^{1/2} - \frac{mc^2}{F} \tag{24c}$$

da cui otteniamo l'equazione del moto

$$x(t) = \frac{mc^2}{F}\left(\sqrt{1+\frac{F^2}{m^2c^2}\,t^2}-1\right) \tag{25}$$

Formula per l'accelerazione

Poniamo che la velocità $\beta \doteq u/c$ sia limitata all'asse z, dunque la trasformazione è data da

$$\begin{cases} c\,\mathrm{d}t' = \gamma(c\,\mathrm{d}t - \beta\,\mathrm{d}z) \\ \mathrm{d}z' = \gamma(\mathrm{d}z - \beta c\,\mathrm{d}t) \end{cases} \tag{26}$$

da questi possiamo derivare la velocità v nel sistema di riferimento primato v' come

$$v' = c\,\frac{\mathrm{d}z'}{c\,\mathrm{d}t'} = c\,\frac{\mathrm{d}z - \beta c\,\mathrm{d}t}{c\,\mathrm{d}t - \beta\,\mathrm{d}z} = c\,\frac{c\,\mathrm{d}t}{c\,\mathrm{d}t}\frac{\dfrac{\mathrm{d}z}{c\,\mathrm{d}t} - \beta}{1 - \beta\dfrac{\mathrm{d}z}{c\,\mathrm{d}t}} = c\,\frac{\dfrac{v}{c} - \beta}{1 - \beta\dfrac{v}{c}} = \frac{v - \beta c}{1 - \beta\dfrac{v}{c}} \tag{27}$$

Usando

$$\gamma \doteq \frac{1}{\sqrt{1-\beta^2}} \implies 1 - \beta^2 = \frac{1}{\gamma^2} \tag{28}$$

calcoliamo il differenziale della velocità

$$\mathrm{d}v' = \mathrm{d}(v - \beta c)\,\frac{1}{1 - \beta\dfrac{v}{c}} + (v - \beta c)\,\mathrm{d}\left(\frac{1}{1 - \beta\dfrac{v}{c}}\right) \tag{29a}$$

$$= \frac{\mathrm{d}v}{1 - \beta\dfrac{v}{c}} + (v - \beta c)\left(-\frac{1}{\left(1 - \beta\dfrac{v}{c}\right)^2}\right)\left(-\frac{\beta}{c}\,\mathrm{d}v\right) \tag{29b}$$

$$= \left(\frac{1}{1 - \beta\dfrac{v}{c}} + \frac{\left(\dfrac{v}{c} - \beta\right)\beta}{\left(1 - \beta\dfrac{v}{c}\right)^2}\right)\mathrm{d}v \tag{29c}$$

$$= \frac{1 - \beta\dfrac{v}{c} + \beta\dfrac{v}{c} - \beta^2}{\left(1 - \beta\dfrac{v}{c}\right)^2}\,\mathrm{d}v \tag{29d}$$

$$= \frac{1}{\gamma^2\left(1 - \beta\dfrac{v}{c}\right)^2}\,\mathrm{d}v \tag{29e}$$

A questo punto possiamo calcolare l'accelerazione

$$a' \doteq \frac{\mathrm{d}v'}{\mathrm{d}t'} \tag{30a}$$

$$= \frac{\mathrm{d}v}{\gamma^2\left(1-\beta\frac{v}{c}\right)^2}\frac{c}{\gamma(c\,\mathrm{d}t-\beta\,\mathrm{d}z)} \tag{30b}$$

$$= \frac{c}{\gamma^3\left(1-\beta\frac{v}{c}\right)^2(c\,\mathrm{d}t-\beta\,\mathrm{d}z)}\,\mathrm{d}v \tag{30c}$$

$$= \frac{c}{\gamma^3\left(1-\beta\frac{v}{c}\right)^2 c\,\mathrm{d}t\left(1-\beta\underbrace{\frac{\mathrm{d}z}{c\,\mathrm{d}t}}_{v/c}\right)}\,\mathrm{d}v \tag{30d}$$

$$= \frac{1}{\gamma^3\left(1-\beta\frac{v}{c}\right)^3}\frac{\mathrm{d}v}{\mathrm{d}t} \tag{30e}$$

$$= \frac{1}{\gamma^3\left(1-\beta\frac{v}{c}\right)^3}a \tag{30f}$$

Essendo la velocità del boost la stessa della velocità di cui abbiamo operato la trasformazione, i.e. $\beta = v/c$, otteniamo

$$a' = \gamma^3 a \tag{31}$$

Accelerazione di un protone

Integriamo ora l'Equazione 31 nel tempo, considerando costante l'accelerazione a'

$$\int_0^t a' \,\mathrm{d}\xi = \int_0^t \gamma^3 a \,\mathrm{d}\xi \tag{32a}$$

$$a't = \int_0^t \gamma^3 \frac{\mathrm{d}v}{\mathrm{d}\xi} \,\mathrm{d}\xi \tag{32b}$$

$$= \int_0^v \left(1 - \frac{w^2}{c^2}\right)^{-3/2} \mathrm{d}w \qquad w \doteq c\sin(\phi) \tag{32c}$$

$$= \int_{\tau/2}^{\operatorname{asin}(v/c)} \left(1 - \sin^2(\theta)\right)^{-3/2} c\cos(\theta)\,\mathrm{d}\theta \qquad \mathrm{d}w = c\cos(\theta)\,\mathrm{d}\theta \tag{32d}$$

$$= \int_{\tau/2}^{\operatorname{asin}(v/c)} \frac{\cos(\theta)}{\cos^3(\theta)} \,\mathrm{d}\theta \tag{32e}$$

$$= c\int_{\tau/2}^{\operatorname{asin}(v/c)} \frac{1}{\cos^2(\theta)} \,\mathrm{d}\theta \tag{32f}$$

$$= c\left[\tan(\theta)\right]_{\tau/2}^{\operatorname{asin}(v/c)} \tag{32g}$$

$$= c\tan\left(\operatorname{asin}\left(\frac{v}{c}\right)\right) \tag{32h}$$

$$= v\left(1 - \frac{v^2}{c^2}\right)^{-1/2} \tag{32i}$$

$$(a't)^2\left(1 - \frac{v^2}{c^2}\right) = v^2 \tag{32j}$$

$$v^2 - (a't)^2\frac{v^2}{c^2} = (a't)^2 \tag{32k}$$

$$v = \frac{a't}{\sqrt{1 + \left(\frac{a't}{c}\right)^2}} \tag{32l}$$

Sostituiamo ora l'Equazione 32l nella formula per l'energia relativistica, esplicitando il tempo

$$\mathcal{E} = m\gamma c^2 \tag{33a}$$

$$= mc^2 \left(1 - \frac{v^2}{c^2}\right)^{-1/2} \tag{33b}$$

$$1 - \frac{v^2}{c^2} = \frac{m^2c^4}{\mathcal{E}^2} \tag{33c}$$

$$v^2 = c^2 \left(1 - \frac{m^2c^4}{\mathcal{E}^2}\right) \tag{33d}$$

$$\frac{a'^2t^2}{1 + \dfrac{a'^2}{c^2}\,t^2} = c^2 \left(1 - \frac{m^2c^4}{\mathcal{E}^2}\right) \tag{33e}$$

$$\frac{a'^2}{c^2}\,t^2 = \left(1 - \frac{m^2c^4}{\mathcal{E}^2}\right)\left(1 + \frac{a'^2}{c^2}\,t^2\right) \tag{33f}$$

$$\cancel{\frac{a'^2}{c^2}}\,t^2 = 1 - \frac{m^2c^4}{\mathcal{E}^2} + \cancel{\frac{a'^2}{c^2}}\,t^2 - \frac{m^2c^4}{\mathcal{E}^2}\frac{a'^2}{c^2}\,t^2 \tag{33g}$$

$$\frac{m^2c^2a'^2}{\mathcal{E}^2}\,t^2 = 1 - \frac{m^2c^4}{\mathcal{E}^2} \tag{33h}$$

$$t = \frac{\mathcal{E}}{mca'}\sqrt{1 - \frac{m^2c^4}{\mathcal{E}^2}} \tag{33i}$$

Ricordando che

$$F = m\gamma^3 a = ma' = eE \tag{34}$$

$$\mathrm{V} = 6.2 \cdot 10^{18}\,\frac{\mathrm{eV}}{\mathrm{C}} = 6.2 \cdot 10^{9}\,\frac{\mathrm{GeV}}{\mathrm{C}} \tag{35}$$

Sostituendo i dati

$$\begin{cases} \mathcal{E} = 1\,\mathrm{TeV} = 10^3\,\mathrm{GeV} \\ m = 1\,\dfrac{\mathrm{GeV}}{\mathrm{c}^2} \\ E = 1\,\dfrac{\mathrm{V}}{\mathrm{m}} = 6.2 \cdot 10^{9}\,\dfrac{\mathrm{GeV}}{\mathrm{m\,C}} \\ e = 1.6 \cdot 10^{-19}\,\mathrm{C} \\ c = 3 \cdot 10^{8}\,\dfrac{\mathrm{m}}{\mathrm{s}} \end{cases} \tag{36}$$

otteniamo

$$t = \frac{\mathcal{E}}{mca'}\sqrt{1-\frac{m^2c^4}{\mathcal{E}^2}} \tag{37a}$$

$$= \frac{\mathcal{E}}{ecE}\sqrt{1-\frac{m^2c^4}{\mathcal{E}^2}} \tag{37b}$$

$$= \underbrace{\frac{10^3\,\mathrm{GeV}}{(1.6\cdot 10^{-19}\,\mathrm{C})\left(3\cdot 10^8\,\frac{\mathrm{m}}{\mathrm{s}}\right)\left(6.2\cdot 10^9\,\frac{\mathrm{GeV}}{\mathrm{m\,C}}\right)}}_{secondi}\underbrace{\sqrt{1-\frac{1\,\frac{\mathrm{GeV}^2}{\mathrm{c}^4}\,c^4}{10^6\,\mathrm{GeV}^2}}}_{adimensionale} \tag{37c}$$

il quale espresso in minuti e secondi risulta

$$t \simeq 55\,\mathrm{m}\,35\,\mathrm{s} \tag{38}$$

4 Esercizio 4

Il linguaggio scelto per questo esercizio è Python.
Importiamo `numpy` per la radice quadrata, il prodotto matriciale, il generatore di numeri casuali e le matrici diagonali. Tramite quest'ultima, definiamo il tensore metrico `eta`.

```
# packages import

import numpy as np

# true random generator

rand = np.random.default_rng()

# metric tensor

eta = np.diag([1, -1, -1, -1])
```

Di seguito la definizione delle funzioni che saranno utili nella definizione delle classi e nel codice stesso. Nello specifico:

dot prodotto matriciale

mod modulo di un 3-vettore

dotLor prodotto di due vettori (entrambi covarianti o controvarianti), intramezzato dal tensore metrico per abbassare/sollevare uno degli indici

g fattore di Lorentz

rnd3Vel generatore di un 3-vettore con entrate casuali, normalizzato in unità di c, velocità della luce

rnd4Vec generatore di un 4-vettore con entrate casuali nell'intervallo $(-10, 10)$

lorTransf generatore di una boost di Lorentz in nella direzione della velocità inserita come argomento

printMatrix funzione per stampare nel terminale le matrici in modo più leggibile

```
def dot(m1,m2):
  # dot product
  return np.dot(m1,m2)

# ---

def mod(vec):
  # module of a 3-vector
  return np.sqrt(sum(x ** 2 for x in vec))

# ---

def dotLor(vec1, vec2):
  # squared module of a 4-vector
  return dot(dot(vec1, eta), vec2)

# ---

def g(v):
  # gamma factor for the velocity v
  return 1 / np.sqrt(1 - mod(v) ** 2)

# ---

def rnd3Vel():
  # generate a random 3-velocity (c = 1)
  vel = 20*rand.random(3) - 10
  vel /= mod(vel) / rand.random()
  return vel

# ---

def rnd4Vec(minVal, maxVal):
  # generate a random 4-vector with entries in (minVal, maxVal)
  vec = (maxVal - minVal)*rand.random(4) + minVal
  return vec

# ---

def lorTransf(v):
# generate a Lorentz boost matrix with velocity given by v
  B = [ [ g(v), - g(v) * v[0], - g(v) * v[1], - g(v) * v[2] ],
  [ - g(v) * v[0], 1 + (g(v) - 1) * (v[0]  / mod(v)) ** 2, \
  (g(v) - 1) * (v[0] * v[1]  / mod(v) ** 2), \
  (g(v) - 1) * (v[0] * v[2]  / mod(v) ** 2) ],
  [ - g(v) * v[1], (g(v) - 1) * (v[1] * v[0]  / mod(v) ** 2), \
  1 + (g(v) - 1) * (v[1]  / mod(v)) ** 2, \
  (g(v) - 1) * (v[1] * v[2]  / mod(v) ** 2) ],
  [ - g(v) * v[2], (g(v) - 1) * (v[2] * v[0]  / mod(v) ** 2), \
  (g(v) - 1) * (v[2] * v[1]  / mod(v) ** 2), \
  1 + (g(v) - 1) * (v[2]  / mod(v)) ** 2 ] ]
  return B

# ---

def printMatrix(matrix):
  for lst in matrix:
```

```
      for element in lst:
        print("{:10.6f}".format(element), end="\t")
      print("")
```

A questo punto, definiamo le due classi che ci serviranno per la soluzione: un vettore di Lorentz e un boost di Lorentz. Entrambi vengono generati casualmente se non viene dato loro come argomento un 4-vettore o una 3-velocità rispettivamente, seguito dal comando `rnd=False`.

lorVec un vettore di Lorentz con entrate modificabili, modulo e tipo (luce, tempo, spazio)

boost un boost di Lorentz con entrate modificabili e velocità su cui è calcolato il boost disponibile

```
class lorVec:

  '''Lorentz vector with entries, module, and squared module'''

  def __init__(self, *vec, **check):
    self.vec = rnd4Vec(-10, 10) if check.get('rnd', True) else list(*vec)
    self.mod = dotLor(self.vec, self.vec)

    if self.mod == 0:
      self.vecType = 'light'
    elif self.mod > 0:
      self.vecType = 'time'
    else:
      self.vecType = 'space'

# ---

class boost:

  '''Lorentz boost, with matrix and versor'''

  def __init__(self, *vel, **check):
    self.vel = rnd3Vel() if check.get('rnd', True) else list(*vel)
    self.boostMatrix = lorTransf(self.vel)
```

Nella parte principale del codice generiamo tre vettori (due casuali e uno fisso)

```
u = lorVec()
v = lorVec()

vec = [1,1,1,0]
w = lorVec(vec, rnd=False)
```

Ne calcoliamo il prodotto scalare

```
uv = dotLor(u.vec, v.vec)
uw = dotLor(u.vec, w.vec)
vw = dotLor(v.vec, w.vec)
```

Generiamo due boost (uno casuale e uno fisso generato da una velocità sull'asse z)

```
B = boost()

vel = [0, 0, 0.9]
C = boost(vel, rnd=False)
```

Facciamo agire i boost sui vettori definiti in precedenza

```
Bu = dot(B.boostMatrix, u.vec)
Bv = dot(B.boostMatrix, v.vec)
Bw = dot(B.boostMatrix, w.vec)

Cu = dot(C.boostMatrix, u.vec)
Cv = dot(C.boostMatrix, v.vec)
Cw = dot(C.boostMatrix, w.vec)
```

e calcoliamo i prodotti dei vettori in seguito ai boost

```
BuBv = dotLor(Bu, Bv)
BuBw = dotLor(Bu, Bw)
BvBw = dotLor(Bv, Bw)

CuCv = dotLor(Cu, Cv)
CuCw = dotLor(Cu, Cw)
CvCw = dotLor(Cv, Cw)
```

A questo punto possiamo calcolare semplicemente le differenze tra i prodotti scalari prima e dopo i boost:

```
VECTORS DECLARATIONS

random vector u:

u =  [-6.11635617 -8.01803261  3.43250309  6.35179459]
module =  -79.00640609375716
type =  space

random vector v:

v =  [ 8.8227775  -4.49636568  6.24238156  8.14558065]
module =  -47.6937131194255
type =  space

fixed vector w:

w =  [1, 1, 1, 0]
module =  -1
type =  space

INNER PRODUCTS BETWEEN VECTORS

u v =  -163.18130540548972
u w =  -1.530826658809973
v w =  7.076761622107156

RANDOM AND FIXED BOOST DECLARATIONS

random boost B:
```

```
boost velocity =  [ 0.13380972 -0.14730109  0.26939327]
boost velocity module =  0.33492594285180055
boost matrix =
1.061296  -0.142012   0.156330  -0.285906
-0.142012   1.009784  -0.010770   0.019697
0.156330  -0.010770   1.011856  -0.021683
-0.285906   0.019697  -0.021683   1.039656

fixed boost C:

boost velocity =  [0, 0, 0.9]
boost velocity module =  0.9
boost matrix =
2.294157  -0.000000  -0.000000  -2.064742
-0.000000   1.000000   0.000000   0.000000
-0.000000   0.000000   1.000000   0.000000
-2.064742   0.000000   0.000000   2.294157

BOOSTING VECTORS

boost via B:

B u =  [-6.63202075 -7.13974113  2.46565788  8.12002052]
B v =  [ 8.64911418 -5.70008076  7.56746121  5.72219358]
B w =  [ 1.07561404  0.85700186  1.15741592 -0.28789191]

boost via C:

C u =  [-27.14669796  -8.01803261   3.43250309  27.20071123]
C v =  [ 3.42232049 -4.49636568  6.24238156  0.47048784]
C w =  [ 2.29415734  1.          1.         -2.0647416 ]

INNER PRODUCTS BETWEEN BOOSTED VECTORS

B u B v =  -163.18130540548972
B u B w =  -1.5308266588099717
B v B w =  7.076761622107155

C u C v =  -163.1813054054897
C u C w =  -1.5308266588099713
C v C w =  7.0767616221071545

COMPARISON BETWEEN INNER PRODUCTS BEFORE AND AFTER BOOST

B u B v - u v = 0.000
B u B w - u w = 0.000
B v B w - v w = -0.000
```

```
C u C v - u v = 0.000
C u C w - u w = 0.000
C v C w - v w = -0.000
```

Queste risultano tutte nulle come ci si aspetta in quanto il prodotto scalare è un invariante di Lorentz.

5 Codice completo

```
# ============================================================================
# EXERCISE 1-4
# ============================================================================

# the following exercise uses the convention:
# c = 1

# ============================================================================
# packages import
# ============================================================================

import numpy as np

# ============================================================================
# constants definition
# ============================================================================

# true random generator

rand = np.random.default_rng()

# metric tensor

eta = np.diag([1, -1, -1, -1])

# ============================================================================
# functions definition
# ============================================================================

def dot(m1,m2):
# dot product
return np.dot(m1,m2)

# ---

def mod(vec):
# module of a 3-vector
return np.sqrt(sum(x ** 2 for x in vec))
```

```
# ---

def dotLor(vec1, vec2):
# squared module of a 4-vector
return dot(dot(vec1, eta), vec2)

# ---

def g(v):
# gamma factor for the velocity v
return 1 / np.sqrt(1 - mod(v) ** 2)

# ---

def rnd3Vel():
# generate a random 3-velocity (c = 1)
vel = 20*rand.random(3) - 10
vel /= mod(vel) / rand.random()
return vel

# ---

def rnd4Vec(minVal, maxVal):
# generate a random 4-vector with entries in (minVal, maxVal)
vec = (maxVal - minVal)*rand.random(4) + minVal
return vec

# ---

def lorTransf(v):
# generate a Lorentz boost matrix with velocity given by v
B = [ [ g(v), - g(v) * v[0], - g(v) * v[1], - g(v) * v[2] ],
[ - g(v) * v[0], 1 + (g(v) - 1) * (v[0]  / mod(v)) ** 2, \
(g(v) - 1) * (v[0] * v[1]  / mod(v) ** 2), \
(g(v) - 1) * (v[0] * v[2]  / mod(v) ** 2) ],
[ - g(v) * v[1], (g(v) - 1) * (v[1] * v[0]  / mod(v) ** 2), \
1 + (g(v) - 1) * (v[1]  / mod(v)) ** 2, \
(g(v) - 1) * (v[1] * v[2]  / mod(v) ** 2) ],
[ - g(v) * v[2], (g(v) - 1) * (v[2] * v[0]  / mod(v) ** 2), \
(g(v) - 1) * (v[2] * v[1]  / mod(v) ** 2), \
1 + (g(v) - 1) * (v[2]  / mod(v)) ** 2 ] ]
return B

# ---

def printMatrix(matrix):
for lst in matrix:
for element in lst:
```

```
print("{:10.6f}".format(element), end="\t")
print("")

# ============================================================================
# classes definition
# ============================================================================

class lorVec:

'''Lorentz vector with entries, module, and squared module'''

def __init__(self, *vec, **check):
self.vec = rnd4Vec(-10, 10) if check.get('rnd', True) else list(*vec)
self.mod = dotLor(self.vec, self.vec)

if self.mod == 0:
self.vecType = 'light'
elif self.mod > 0:
self.vecType = 'time'
else:
self.vecType = 'space'

# ---

class boost:

'''Lorentz boost, with matrix and versor'''

def __init__(self, *vel, **check):
self.vel = rnd3Vel() if check.get('rnd', True) else list(*vel)
self.boostMatrix = lorTransf(self.vel)

# ============================================================================
# actual code
# ============================================================================

print('============================================================================')
print('OUTPUT')
print('============================================================================')

# generate two random vectors and one fixed

print('VECTORS DECLARATIONS\n')

print('random vector u:\n')

u = lorVec()
print('u = ', u.vec, '\nmodule = ', u.mod, '\ntype = ', u.vecType)
```

```
print('\nrandom vector v:\n')

v = lorVec()
print('v = ', v.vec, '\nmodule = ', v.mod, '\ntype = ', v.vecType)

print('\nfixed vector w:\n')

vec = [1,1,1,0]

w = lorVec(vec, rnd=False)
print('w = ', w.vec, '\nmodule = ', w.mod, '\ntype = ', w.vecType)

# inner products between 4-vectors

print('\nINNER PRODUCTS BETWEEN VECTORS\n')

uv = dotLor(u.vec, v.vec)
uw = dotLor(u.vec, w.vec)
vw = dotLor(v.vec, w.vec)

print('u v = ', uv)
print('u w = ', uw)
print('v w = ', vw)

# boost of v in random direction

print('\nRANDOM AND FIXED BOOST DECLARATIONS\n')

print('random boost B:\n')

B = boost()
print('boost velocity = ', B.vel, '\nboost velocity module = ', mod(B.vel))
print('boost matrix =')
printMatrix(B.boostMatrix)

print('\nfixed boost C:\n')

vel = [0, 0, 0.9]

C = boost(vel, rnd=False)
print('boost velocity = ', vel, '\nboost velocity module = ', mod(vel))
print('boost matrix =')
printMatrix(C.boostMatrix)

# boosting vectors

print('\nBOOSTING VECTORS\n')
```

```
print('boost via B:\n')

Bu = dot(B.boostMatrix, u.vec)
Bv = dot(B.boostMatrix, v.vec)
Bw = dot(B.boostMatrix, w.vec)

print('B u = ', Bu)
print('B v = ', Bv)
print('B w = ', Bw)

print('\nboost via C:\n')

Cu = dot(C.boostMatrix, u.vec)
Cv = dot(C.boostMatrix, v.vec)
Cw = dot(C.boostMatrix, w.vec)

print('C u = ', Cu)
print('C v = ', Cv)
print('C w = ', Cw)


# inner products between boosted 4-vectors

print('\nINNER PRODUCTS BETWEEN BOOSTED VECTORS\n')

BuBv = dotLor(Bu, Bv)
BuBw = dotLor(Bu, Bw)
BvBw = dotLor(Bv, Bw)

print('B u B v = ', BuBv)
print('B u B w = ', BuBw)
print('B v B w = ', BvBw)

print()

CuCv = dotLor(Cu, Cv)
CuCw = dotLor(Cu, Cw)
CvCw = dotLor(Cv, Cw)

print('C u C v = ', CuCv)
print('C u C w = ', CuCw)
print('C v C w = ', CvCw)

# comparison between inner products before and after boost

print('\nCOMPARISON BETWEEN INNER PRODUCTS BEFORE AND AFTER BOOST\n')
```

```
print('B u B v - u v = {:2.3f}'.format(BuBv - uv))
print('B u B w - u w = {:2.3f}'.format(BuBw - uw))
print('B v B w - v w = {:2.3f}'.format(BvBw - vw))

print()

print('C u C v - u v = {:2.3f}'.format(CuCv - uv))
print('C u C w - u w = {:2.3f}'.format(CuCw - uw))
print('C v C w - v w = {:2.3f}'.format(CvCw - vw))

print('=============================================================================')
print('END OF OUTPUT')
print('=============================================================================')
```