

Esercitazione 1

Elettrodinamica relativistica

A.A. 2023/2024

Simone Iovine

Indice

1	Esercizio 1	1
2	Esercizio 2	4
3	Esercizio 3	5
4	Esercizio 4	11
5	Codice completo	16

1 Esercizio 1

Riscriviamo la definizione della rapidità

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

nel seguente modo

$$e^{2y} = \frac{E + cp_z}{E - cp_z} = \frac{1 + \frac{v_z}{c}}{1 - \frac{v_z}{c}} \quad (2)$$

definendo

$$\beta \doteq \frac{u}{c} \quad (3)$$

dove u è la velocità del boost, abbiamo anche

$$e^{2\eta} = \frac{1 + \beta}{1 - \beta} \quad (4)$$

Considerando il 4-momento

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

un boost lungo z può essere calcolato tramite la matrice

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

dunque

$$\vec{\mathbf{p}}' = \Lambda \vec{\mathbf{p}} \quad (7)$$

Le componenti che hanno subito un cambiamento saranno

$$\begin{cases} E' = \gamma(E - \beta cp_z) \\ cp'_z = \gamma(cp_z - \beta E) \end{cases} \quad (8)$$

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

$$\begin{cases} E' = \gamma(E + \beta cp_z) \\ cp'_z = \gamma(cp_z + \beta E) \end{cases} \quad (9)$$

A questo punto consideriamo la rapidità nel sistema di riferimento primato

$$e^{2y'} = \frac{E' + cp'_z}{E' - cp'_z} \quad (10a)$$

$$(10b)$$

$$= \frac{\gamma(E + \beta cp_z) + \gamma(cp_z + \beta E)}{\gamma(E + \beta cp_z) - \gamma(cp_z + \beta E)} \quad (10c)$$

$$(10d)$$

$$= \frac{E + \beta cp_z + cp_z + \beta E}{E + \beta cp_z - cp_z - \beta E} \quad (10e)$$

$$(10f)$$

$$= \frac{E + \beta cp_z + \beta(E + cp_z)}{E - \beta cp_z - \beta(E - cp_z)} \quad (10g)$$

$$(10h)$$

$$= \frac{E + cp_z}{E - cp_z} \frac{1 + \beta}{1 - \beta} \quad (10i)$$

$$= \underbrace{e^{2y}}_{e^{2y}} \underbrace{e^{2\eta}}_{e^{2\eta}} = e^{2(y+\eta)} \quad (10j)$$

$$e^{2y'} = e^{2y'} e^{2\eta} \implies y' = y + \eta \quad (11)$$

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

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

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

da questi possiamo derivare la velocità nel sistema di riferimento primato

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

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

$$\frac{v'_z}{c} = \frac{\frac{v_z}{c} + \beta}{1 + \beta \frac{v_z}{c}} \quad (15)$$

A questo punto sostituiamo nella definizione della rapidità

$$e^{2y'} = \frac{1 + \frac{v'_z}{c}}{1 - \frac{v'_z}{c}} \quad (16a)$$

$$= \frac{1 + \frac{\frac{v_z}{c} + \beta}{1 + \beta \frac{v_z}{c}}}{1 - \frac{\frac{v_z}{c} + \beta}{1 + \beta \frac{v_z}{c}}} \quad (16b)$$

$$= \frac{1 + \beta \frac{v_z}{c} + \frac{v_z}{c} + \beta}{1 + \beta \frac{v_z}{c} - \frac{v_z}{c} - \beta} \quad (16c)$$

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

$$= \frac{1 + \frac{v_z}{c}}{1 - \frac{v_z}{c}} \frac{1 + \beta}{1 - \beta} \quad (16e)$$

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

$$e^{2y'} = e^{2y'} e^{2\eta} \implies y' = y + \eta \quad (17)$$

2 Esercizio 2

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

$$\sigma = \frac{d\nu}{dV dt} \frac{1}{v_{rel} n_1 n_2} \quad (18)$$

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 :

$$\left\{ \begin{array}{l} d\nu' = d\nu \\ dV' = \frac{1}{\gamma} dV \\ dt' = \gamma dt \\ n'_1 = \frac{1}{\gamma} n_1 \\ n'_2 = \gamma n_2 \end{array} \right. \quad (19)$$

questo perché:

- il numero di urti $d\nu$ 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

$$\sigma' = \frac{d\nu'}{dV' dt'} \frac{1}{v_{rel} n'_1 n'_2} = \frac{d\nu}{\frac{1}{\gamma} dV \gamma dt'} \frac{1}{v_{rel} \frac{1}{\gamma} n_1 \gamma n'_2} = \sigma \quad (20)$$

3 Esercizio 3

Equazione del moto

Consideriamo la seconda legge di Newton

$$\vec{F} = \frac{d\vec{p}}{dt} \quad (21)$$

dove il momento relativistico è pari a

$$\vec{p} = m\gamma\vec{v} \quad (22)$$

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

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

$$\frac{mv}{\sqrt{1 - \frac{v^2}{c^2}}} = Ft \quad (23b)$$

$$\frac{m^2 v^2}{1 - \frac{v^2}{c^2}} = F^2 t^2 \quad (23c)$$

$$\frac{m^2 c^2}{c^2 - v^2} v^2 = F^2 t^2 \quad (23d)$$

$$\frac{m^2 c^2}{F^2 t^2} v^2 = c^2 - v^2 \quad (23e)$$

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

$$v = \frac{c}{\sqrt{1 + \frac{m^2 c^2}{F^2 t^2}}} \quad (23g)$$

$$= \frac{c}{\sqrt{\frac{F^2 t^2 + m^2 c^2}{F^2 t^2}}} \quad (23h)$$

$$= \frac{c F t}{\sqrt{F^2 t^2 + m^2 c^2}} \quad (23i)$$

$$= \frac{c}{m c} \frac{F t}{\sqrt{1 + \frac{F^2 t^2}{m^2 c^2}}} \quad (23j)$$

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

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

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

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

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

da cui otteniamo l'equazione del moto

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

Formula per l'accelerazione

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

$$\begin{cases} c \, dt' = \gamma(c \, dt - \beta \, dz) \\ dz' = \gamma(dz - \beta c \, dt) \end{cases} \quad (26)$$

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

$$v' = c \frac{dz'}{c \, dt'} = c \frac{dz - \beta c \, dt}{c \, dt - \beta \, dz} = c \frac{c \, dt \frac{dz}{c \, dt} - \beta}{c \, dt \left(1 - \beta \frac{dz}{c \, dt}\right)} = c \frac{\frac{v}{c} - \beta}{1 - \beta \frac{v}{c}} = \frac{v - \beta c}{1 - \beta \frac{v}{c}} \quad (27)$$

Usando

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

calcoliamo il differenziale della velocità

$$dv' = d(v - \beta c) \frac{1}{1 - \beta \frac{v}{c}} + (v - \beta c) d \left(\frac{1}{1 - \beta \frac{v}{c}} \right) \quad (29a)$$

$$= \frac{dv}{1 - \beta \frac{v}{c}} + (v - \beta c) \left(-\frac{1}{\left(1 - \beta \frac{v}{c}\right)^2} \right) \left(-\frac{\beta}{c} dv \right) \quad (29b)$$

$$= \left(\frac{1}{1 - \beta \frac{v}{c}} + \frac{\left(\frac{v}{c} - \beta\right) \beta}{\left(1 - \beta \frac{v}{c}\right)^2} \right) dv \quad (29c)$$

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

$$= \frac{1}{\gamma^2 \left(1 - \beta \frac{v}{c}\right)^2} dv \quad (29e)$$

A questo punto possiamo calcolare l'accelerazione

$$a' \doteq \frac{dv'}{dt'} \quad (30a)$$

$$= \frac{dv}{\gamma^2 \left(1 - \beta \frac{v}{c}\right)^2} \frac{c}{\gamma(c dt - \beta dz)} \quad (30b)$$

$$= \frac{c}{\gamma^3 \left(1 - \beta \frac{v}{c}\right)^2 (c dt - \beta dz)} dv \quad (30c)$$

$$= \frac{c}{\gamma^3 \left(1 - \beta \frac{v}{c}\right)^2 c dt \left(1 - \beta \frac{dz}{\underbrace{c dt}_{v/c}}\right)} dv \quad (30d)$$

$$= \frac{1}{\gamma^3 \left(1 - \beta \frac{v}{c}\right)^3} \frac{dv}{dt} \quad (30e)$$

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

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

$$a' = \gamma^3 a \quad (31)$$

Accelerazione di un protone

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

$$\int_0^t a' d\xi = \int_0^t \gamma^3 a d\xi \quad (32a)$$

$$a't = \int_0^t \gamma^3 \frac{dv}{d\xi} d\xi \quad (32b)$$

$$= \int_0^v \left(1 - \frac{w^2}{c^2}\right)^{-3/2} dw \quad (32c)$$

$$= \int_{\tau/2}^{\text{asin}(v/c)} (1 - \sin^2(\theta))^{-3/2} c \cos(\theta) d\theta \quad \left. \begin{array}{l} w \doteq \\ c \sin(\phi) \\ dw = \\ c \cos(\theta) d\theta \end{array} \right\} \quad (32d)$$

$$= \int_{\tau/2}^{\text{asin}(v/c)} \frac{\cos(\theta)}{\cos^3(\theta)} d\theta \quad (32e)$$

$$= c \int_{\tau/2}^{\text{asin}(v/c)} \frac{1}{\cos^2(\theta)} d\theta \quad (32f)$$

$$= c [\tan(\theta)]_{\tau/2}^{\text{asin}(v/c)} \quad (32g)$$

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

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

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

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

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

Sostituiamo ora l'Equazione [32l](#) nella formula per l'energia relativistica, esplicitando il tempo

$$\mathcal{E} = m\gamma c^2 \quad (33a)$$

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

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

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

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

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

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

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

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

Ricordando che

$$F = m\gamma^3 a = ma' = eE \quad (34)$$

$$V = 6.2 \cdot 10^{18} \frac{\text{eV}}{\text{C}} = 6.2 \cdot 10^9 \frac{\text{GeV}}{\text{C}} \quad (35)$$

Sostituendo i dati

$$\left\{ \begin{array}{l} \mathcal{E} = 1 \text{ TeV} = 10^3 \text{ GeV} \\ m = 1 \frac{\text{GeV}}{c^2} \\ E = 1 \frac{\text{V}}{\text{m}} = 6.2 \cdot 10^9 \frac{\text{GeV}}{\text{m C}} \\ e = 1.6 \cdot 10^{-19} \text{ C} \\ c = 3 \cdot 10^8 \frac{\text{m}}{\text{s}} \end{array} \right. \quad (36)$$

otteniamo

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

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

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

il quale espresso in minuti e secondi risulta

$$t \simeq 55 \text{ m } 35 \text{ s} \quad (38)$$

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`.

```

1  # packages import
2
3  import numpy as np
4
5  # true random generator
6
7  rand = np.random.default_rng()
8
9  # metric tensor
10
11 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

```
1  def dot(m1,m2):
2      # dot product
3      return np.dot(m1,m2)
4
5  # ---
6
7  def mod(vec):
8      # module of a 3-vector
9      return np.sqrt(sum(x ** 2 for x in vec))
10
11 # ---
12
13 def dotLor(vec1, vec2):
14     # squared module of a 4-vector
15     return dot(dot(vec1, eta), vec2)
16
17 # ---
18
19 def g(v):
20     # gamma factor for the velocity v
21     return 1 / np.sqrt(1 - mod(v) ** 2)
22
23 # ---
24
25 def rnd3Vel():
26     # generate a random 3-velocity (c = 1)
27     vel = 20*rand.random(3) - 10
28     vel /= mod(vel) / rand.random()
29     return vel
30
31 # ---
32
33 def rnd4Vec(minVal, maxVal):
34     # generate a random 4-vector with entries in (minVal, maxVal)
35     vec = (maxVal - minVal)*rand.random(4) + minVal
36     return vec
37
38 # ---
39
40 def lorTransf(v):
41     # generate a Lorentz boost matrix with velocity given by v
42     B = [ [ g(v), - g(v) * v[0], - g(v) * v[1], - g(v) * v[2] ],
43           [ - g(v) * v[0], 1 + (g(v) - 1) * (v[0] / mod(v)) ** 2, \
44             (g(v) - 1) * (v[0] * v[1] / mod(v) ** 2), \
45             (g(v) - 1) * (v[0] * v[2] / mod(v) ** 2) ],
46           [ - g(v) * v[1], (g(v) - 1) * (v[1] * v[0] / mod(v) ** 2), \
47             1 + (g(v) - 1) * (v[1] / mod(v)) ** 2, \
48             (g(v) - 1) * (v[1] * v[2] / mod(v) ** 2) ],
49           [ - g(v) * v[2], (g(v) - 1) * (v[2] * v[0] / mod(v) ** 2), \
50             (g(v) - 1) * (v[2] * v[1] / mod(v) ** 2), \
51             1 + (g(v) - 1) * (v[2] / mod(v)) ** 2 ] ]
52     return B
53
54 # ---
55
56 def printMatrix(matrix):
57     for lst in matrix:
```

```

58     for element in lst:
59         print("{:10.6f}".format(element), end="\t")
60     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

```

1  class lorVec:
2
3      '''Lorentz vector with entries, module, and squared module'''
4
5      def __init__(self, *vec, **check):
6          self.vec = rnd4Vec(-10, 10) if check.get('rnd', True) else list(*vec)
7          self.mod = dotLor(self.vec, self.vec)
8
9          if self.mod == 0:
10             self.vecType = 'light'
11         elif self.mod > 0:
12             self.vecType = 'time'
13         else:
14             self.vecType = 'space'
15
16     # ---
17
18     class boost:
19
20         '''Lorentz boost, with matrix and versor'''
21
22         def __init__(self, *vel, **check):
23             self.vel = rnd3Vel() if check.get('rnd', True) else list(*vel)
24             self.boostMatrix = lorTransf(self.vel)

```

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

```

1  u = lorVec()
2  v = lorVec()
3
4  vec = [1,1,1,0]
5  w = lorVec(vec, rnd=False)

```

Ne calcoliamo il prodotto scalare

```

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

```

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

```

1  B = boost()
2
3  vel = [0, 0, 0.9]
4  C = boost(vel, rnd=False)

```

Facciamo agire i boost sui vettori definiti in precedenza

```
1 Bu = dot(B.boostMatrix, u.vec)
2 Bv = dot(B.boostMatrix, v.vec)
3 Bw = dot(B.boostMatrix, w.vec)
4
5 Cu = dot(C.boostMatrix, u.vec)
6 Cv = dot(C.boostMatrix, v.vec)
7 Cw = dot(C.boostMatrix, w.vec)
```

e calcoliamo i prodotti dei vettori in seguito ai boost

```
1 BuBv = dotLor(Bu, Bv)
2 BuBw = dotLor(Bu, Bw)
3 BvBw = dotLor(Bv, Bw)
4
5 CuCv = dotLor(Cu, Cv)
6 CuCw = dotLor(Cu, Cw)
7 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 = {:.3f}'.format(BuBv - uv))
print('B u B w - u w = {:.3f}'.format(BuBw - uw))
print('B v B w - v w = {:.3f}'.format(BvBw - vw))

print()

print('C u C v - u v = {:.3f}'.format(CuCv - uv))
print('C u C w - u w = {:.3f}'.format(CuCw - uw))
print('C v C w - v w = {:.3f}'.format(CvCw - vw))

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

```