UNIVERSITÀ DEGLI STUDI DI CATANIA

Dipartimento di Fisica e Astronomia "Ettore Majorana"

Corso di Laurea in Fisica

Relazioni di Laboratorio di Fisica 3

Le quattro cose

Anno Accademico 2024 – 2025

Indice

Sommario

In questo documento sono raccolte le quattro relazioni brevi da svolgere durante il corso annuale di *Laboratorio di Fisica 3* del Corso di Laurea in *Fisica* presso l'Università degli Studi di Catania.

Le esperienze sono esposte nei quattro capitoli seguenti:

1. *Implementazione numerica della formula di Bethe–Bloch.* Attraverso un codice in C che implementa numericamente la formula di Bethe–Bloch ho simulato il passaggio di una particella α a 5 MeV attraverso un sottile foglio di alluminio, realizzando un grafico che rappresenta l'energia della particella e la quantità di energia ceduta in funzione della distanza percorsa dentro il materiale.

2. *Misura di temperature con Arduino.* Attraverso l'uso di un microcontrollore Arduino, un sensore di temperatura e un semplice codice ho misurato la variazione di temperatura di una stanza in seguito all'accensione del riscaldamento. Nella relazione analizzo qualitativamente i dati raccolti ed estrapolo una possibile funzione che ne modelli l'andamento.

3. *Misura di resistenze con un multimetro digitale.* Utilizzando un multimetro digitale ho effettuato la misura dei resistori forniti dal kit del multimetro, verificandone la distribuzione statistica. A partire dai risultati di questo studio ho confrontato le misure di alcune delle resistenze collegate in parallelo con i valori previsti teoricamente. Infine ho trovato la resistività di un anello d'argento sfruttando di nuovo una misura di resistenza e considerazioni geometriche.

4. *Accettanza geometrica di un rivelatore.* Con un altro codice in C ho applicato il metodo Montecarlo per valutare numericamente l'accettanza geometrica di un rivelatore a forma di dischetto in presenza di una sorgente di radiazioni isotropa ed estesa. Dai dati simulati ho realizzato delle immagini rappresentative del sistema e un istogramma che mostri la distribuzione della radiazione sul sensore.

Riporto inoltre i collegamenti alle mie due repository su GitHub dove è possibile consultare e scaricare i codici sorgente in C qualora si desiderasse utilizzarli.

1. https://github.com/ImAure/bethe-bloch-simulation
2. https://github.com/ImAure/geometric-acceptance

Il codice realizzato per l'esperienza con Arduino, essendo molto più breve, è invece esposto in appendice §A.

1 Implementazione Numerica della Formula di Bethe–Bloch

1.1 Il modello

Come descritto dal Particle Data Group [1], la formula di Bethe–Bloch è un modello sperimentale che descrive la perdita di energia di particelle cariche pesanti di media energia—come protoni e particelle α—nella materia:

$$\left\langle -\frac{\mathrm{d}E}{\mathrm{d}x} \right\rangle = Kz^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} \log \frac{2m_e c^2 \beta^2 \gamma^2 W_{\max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right], \tag{1.1}$$

dove β e γ sono le usuali quantità relativistiche mentre il resto dei simboli sono esplicitati in Tab. 1.1. La perdita di energia media data dalla (1.1) è misuarta in $\mathrm{MeV\,g^{-1}\,cm^2}$ ma può essere portata in $\mathrm{MeV\,cm^{-1}}$ moltiplicando entrambi i membri per la densità volumica ρ del bersaglio misurata in $\mathrm{g\,cm^{-3}}$. La quantità $W_{\max}$ è la massima energia che una particella carica può cedere a un elettrone e si

Simbolo	Definizione	Valore o unità di misura
$m_e c^2$	massa a riposo dell'elettrone $\times c^2$	$0.510\,998\,950\,00(15)\,\mathrm{MeV}$
r_e	raggio classico dell'elettrone $e^2/4\pi\epsilon_0 m_e c^2$	$2.817\,940\,322\,7(19)\,\mathrm{fm}$
N_A	numero di Avogadro	$6.022\,140\,857(74) \times 10^{23}\,\mathrm{mol^{-1}}$
ρ	densità	$\mathrm{g\,cm^{-3}}$
x	massa per unità di area	$\mathrm{g\,cm^{-2}}$
M	massa della particella incidente	$\mathrm{MeV}\,c^{-2}$
E	energia della particella incidente $\gamma M c^2$	MeV
$W_{\max}$	massima energia trasferibile per collisioni	MeV
z	numero di carica della particella incidente	
Z	numreo atomico del bersaglio	
A	numero di massa atomica del bersaglio	
K	$4\pi N_\mathrm{A} r_e^2 m_e c^2$	$0.307\,075\,\mathrm{MeV\,mol^{-1}\,cm^2}$
I	energia media di eccitazione	eV
$\delta(\beta\gamma)$	correzione di ionizzazione	

Tabella 1.1: Notazione e unità di misura per la formula di Bethe–Bloch. Si tratta di un riassunto della tabella del PDG [1].

Tabella 1.2: caption!!

esprime come

$$W_{\max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + \gamma\, m_e/M + (m_e/M)^2}\,. \tag{1.2}$$

La (1.1) e la (1.2) sono valide nell'approssimazione $0.1 \lesssim \beta\gamma \lesssim 1000$ poiché al limite inferiore la velocità del proiettile diventa confrontabile con la "velocità" degli elettroni atomici mentre al limite superiore gli effetti radiativi non sono più trascurabili.

1.2 La simulazione

Ho scelto di simulare la perdita di energia di una particella α a 5 MeV attraverso un foglio di alluminio di spessore[1] 0.022 mm. Per realizzare la simulazione ho scritto un codice in C che implementa la (1.1) in modo numerico.

1.2.1 Descrizione del codice e dei calcoli

L'utente che lancia il programma sceglie lo spessore del materiale bersaglio e il numero di step in cui suddividere il calcolo. Il programma quindi chiede all'utente di selezionare un proiettile tra α, protoni o muoni e un materiale tra alluminio e rame. I dati come le masse delle particelle, le densità dei materiali e i valori di I, sono definiti come costanti all'inizio del programma. Non li riporto in questo elaborato per brevità—la totalità del codice, costanti incluse, è consultabile sul mio profilo GitHub.[2]

Assumiamo che in ciascun intervallo di spessore[3] ds tutte le quantità variabili di nostro interesse siano costanti—velocità, energia, perdita di energia *et cetera*. Dalla teoria della relatività ristretta scriviamo per l'n-esimo intervallo

$$E_n = T_n + Mc^2\,,$$
$$E_n = \gamma_n Mc^2\,,$$

dove M è la massa a riposo della particella incidente, T_n la sua energia cinetica ed E_n la sua energia totale. Note queste ultime due quantità, è possibile calcolare il fattore di Lorentz γ_n e β_n^2 come

$$\gamma_n = \frac{T_n + Mc^2}{Mc^2}\,, \quad \beta_n^2 = 1 - \frac{1}{\gamma_n^2}\,.$$

[1] Dopo aver provato con uno spessore di 0.015 mm e aver constatato che la curva risultava tagliata—la particella non cedeva tutta l'energia all'alluminio—ho scelto di aumentare di poco lo spessore per il gusto di un grafico più completo.

[2] Repository: https://github.com/ImAure/bethe-bloch-simulation

[3] Ricordo che in questo caso la x non indica una distanza.

Noti γ_n e β_n, è possibile calcolare la perdita di energia media per unità di lunghezza attraverso la (1.1) moltiplicata per ρ. Possiamo in particolare calcolare l'energia cinetica con cui la particella α entra nello strato successivo, T_{n+1} come

$$T_{n+1} = T_n + \mathrm{d}T_n = T_n + \rho \left\langle -\frac{\mathrm{d}E}{\mathrm{d}x} \right\rangle_n \mathrm{d}s \ ,$$

essendo naturalmente $\mathrm{d}T_n \leq 0$. Il calcolo di $\rho \left| \langle -\mathrm{d}E \,/\, \mathrm{d}x \rangle_n \right|$ è svolto dalla funzione `bethe()`, con l'ausilio della funzione `wmax()` che in particolare calcola il termine $W_{\max}$:

```
1 double wmax(double beta2, double gamma, double gamma2, double m,
  double m2) {
2     return (2.0 * E_M * beta2 * gamma2) / (1.0 + (2.0 * gamma * m) +
  m2);
3 }
4
5 double bethe(double z_inc, double z_tar, double a_tar, double beta2,
  double gamma2, double w_max, double i2) {
6     return (((K * z_inc * z_inc * z_tar) / (a_tar * beta2)) * (0.5 *
  log((2.0 * E_M * beta2 * gamma2 * w_max) / (i2 * 1.0e-12)) - beta2));
7 }
```

Il calcolo viene quindi ripetuto a partire dalla nuova energia T_{n+1} per ottenere la perdita di energia attraverso il successivo strato. Per ogni reiterazione il codice stampa su un file la distanza totale percorsa, l'energia cinetica della particella e la quantità $|\mathrm{d}T_n| = \rho \left| \langle -\mathrm{d}E \,/\, \mathrm{d}x \rangle_n \right|$.

1.2.2 Risultati della simulazione

Osserviamo adesso i dati che si ottengono inserendo nel programma un numero arbitrariamente alto di step pari a 500. In Fig. 1.1 sono riportati in due grafici i valori dell'energia cinetica della particella α e del potere d'arresto del materiale al variare della distanza percorsa.

Dal primo grafico risulta evidente la decrescita dell'energia della particella. Si nota che l'energia cinetica non si annulla del tutto, nonostante si avvicini ragionevolmente a 0 MeV: questo può essere dovuto a imprecisioni nel codice come il modo in cui vengono gestiti valori di T negativi e valori di $\mathrm{d}T$ che farebbero aumentare l'energia.

Nel secondo grafico invece è riportato l'andamento del potere d'arresto, detto comunemente *curva di Bragg*. L'energia depositata dalla particella α è inversamente proporzionale al quadrato della velocità, per questo subito prima del totale arresto si osserva il massimo deposito di energia nel tratto di grafico detto *picco di Bragg*.

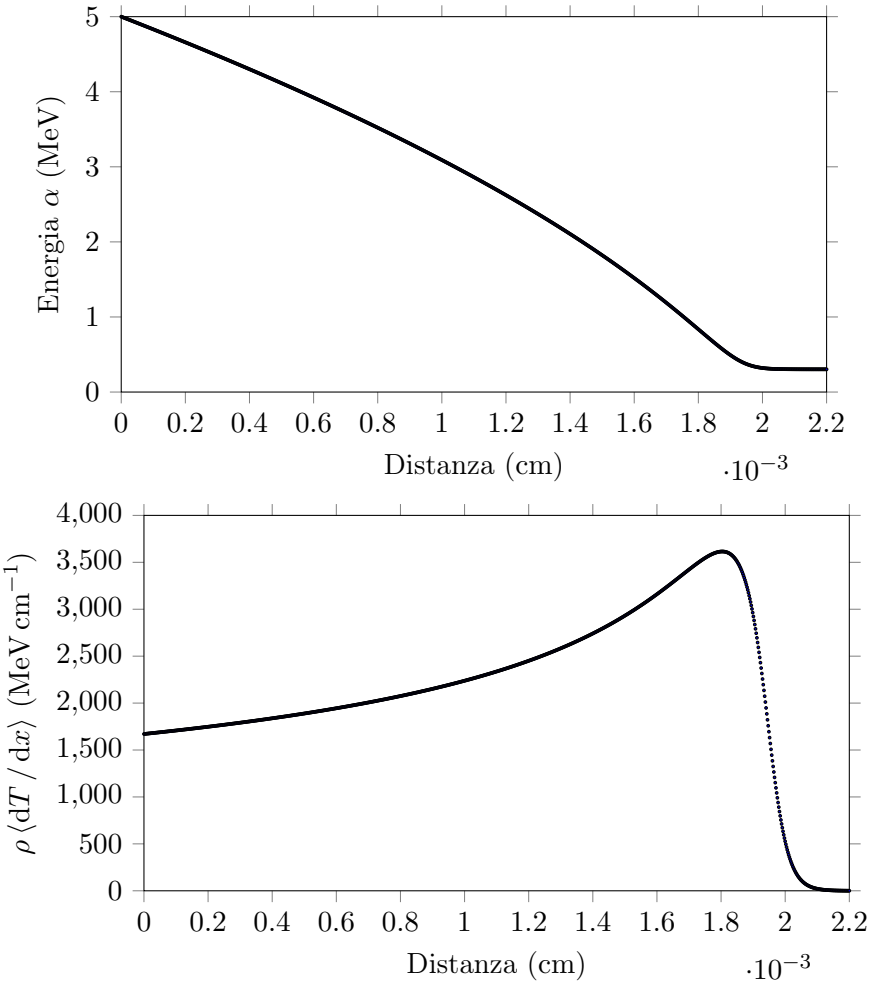

Figura 1.1: In alto l'energia della particella α che diminuisce man mano che la particella penetra l'alluminio. In basso il potere d'arresto con l'evidente picco di Bragg.

2 Accettanza Geometrica di un Rivelatore

L'utlima esperienza consiste nel calcolare l'accettanza geometrica di un rivelatore, definita come il rapporto tra il numero di particelle incidenti e il numero di particelle emesse, sfruttando la generazione di numeri casuali e il metodo montecarlo.

2.1 Il sistema fisico proposto

L'obiettivo che mi sono posto è stato quello di calcolare l'accettanza geometrica di un rivelatore a forma di dischetto con raggio variabile, posto in una posizione arbitraria nel semispazio positivo delle z. Allo stesso modo, la sorgente di particelle è costituita da un secondo dischetto, anch'esso con raggio variabile, centrato nell'origine.

Le ipotesi che ho fatto sono le seguenti:

1. Ciascun punto della sorgente emette in maniera isotropa, ovvero in tutte le direzioni con la stessa probabilità;
2. La probabilità di emissione da parte di un punto della sorgente è uniforme, ovvero tutti i punti hanno la stessa probabilità di emettere una particella;
3. Il rivelatore e la sorgente sono sempre paralleli tra loro e al piano Oxy, e il rivelatore è sempre posto sopra la sorgente.

Per ragioni di simmetria, la posizione del rivelatore sopra la sorgente non causa alcuna perdita di generalità. Se il rivelatore avesse altezza $z = 0$, visto lo spessore nullo sia della sorgente che del rivelatore, nessuna particella verrebbe rivelata. La sorgente inoltre emette in modo simmetrico rispetto al piano Oxy, quindi il caso $z < 0$ è equivalente al caso $z > 0$.

2.1.1 Sorgente puntiforme

Dal momento che la sorgente può emettere da un solo punto alla volta, cominciamo col risolvere il problema nel caso di una sorgente puntiforme nell'origine e un rivelatore che può essere spostato rispettando le ipotesi di cui sopra.

Affinché la distribuzione delle particelle sia uniforme, generare delle coordinate cartesiane casuali non è sufficiente. La scelta che ho fatto è quella di generare delle *direzioni* casuali in coordinate sferiche $(\varrho, \vartheta, \varphi)$, risparmiando la spesa computazionale di generare la distanza ϱ dall'origine. Per le proprietà delle coordinate sferiche possiamo generare uniformemente $\cos\vartheta \in [-1; 1]$, per poi ricavare ϑ prendendone l'arcocoseno, mentre possiamo generare uniformemente $\varphi \in [0; 2\pi]$.

Fissata una direzione (ϑ, φ), la particella emessa dalla sorgente si muoverà lungo la retta che passa per l'origine avente tale direzione. Ci chiediamo quindi se questa retta intersecherà il rivelatore, e in caso affermativo, in quale punto.[1]

Da semplici considerazioni geometrice, se indichiamo con $P \equiv (x, y, z)$ il generico punto dello spazio, con $C \equiv x_c, y_c, z_c$ il centro del rivelatore e con R il suo raggio, il punto appartiene al dischetto se si verificano contemporaneamente le seguenti condizioni:

$$(P - C)^2 \leq R^2 \,, \tag{2.1}$$

$$z = z_c \,. \tag{2.2}$$

La prima condizione corrisponde a $(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 \leq R^2$, che messa a sistema con la seconda dà $(x - x_c)^2 + (y - y_c)^2 \leq R^2$. Passiamo quindi a coordinate polari per trovare le condizioni che devono rispettare i valori di ϑ e φ:

$$\begin{cases} x = \varrho \sin\vartheta \cos\varphi \\ y = \varrho \sin\vartheta \sin\varphi \\ z = \varrho \cos\vartheta \end{cases} .$$

Sostituendo queste relazioni nella condizione di appartenenza al cerchio troviamo:

$$(\varrho \sin\vartheta \cos\varphi - x_c)^2 + (\varrho \sin\vartheta \sin\varphi - y_c)^2 \leq R^2 \,,$$

che con qualche passaggio si riduce a

$$\varrho^2 \sin^2\vartheta + x_c^2 + y_c^2 - 2\varrho \sin\vartheta (x_c \cos\varphi + y_c \sin\varphi) \leq R^2 \,,$$

che a sua volta, a patto di porre $d^2 = x_c^2 + y_c^2$, $x_c = d\cos\varphi_c$ e $y_c = d\sin\varphi_c$, non è altro che il teorema del coseno applicato al triangolo formato dalle proiezioni sul piano Oxy dei punti O, P e C, che ha un angolo in O proprio pari a $\varphi - \varphi_c$:

$$\varrho^2 \sin^2\vartheta + d^2 - 2\varrho \sin\vartheta d \cos(\varphi - \varphi_c) \leq R^2 \,. \tag{2.3}$$

Se assumiamo $\cos\vartheta \neq 0$—che è vero se $P \notin Oxy$, e ciò non è restrittivo in quanto il piano è un sottoinsieme a misura nulla dello spazio $\mathbb{R}^3$—possiamo invertire la (2.2) scritta in coordinate polari per esplicitare ϱ:

$$\varrho = \frac{z_c}{\cos\vartheta} \,. \tag{2.4}$$

[1] Sapere il punto non è necessario per il calcolo dell'accettanza geometrica, ma è utile allo scopo di generare un'immagine del sistema.

Sostituendo quest'ultima relazione (2.4) nella (2.3) si ottiene

$$z_c \tan\vartheta + d^2 - 2z_c \tan\vartheta d \cos(\varphi - \varphi_c) \leq R^2 \,. \tag{2.5}$$

I valori z_c, d e φ_c sono costanti, per cui una volta generati ϑ e φ è sufficiente verificare la disequazione per sapere se il raggio interseca o no il rivelatore. Resta quindi da trovare esplicitamente la posizione dei punti $P^* \equiv (x^*, y^*, z^*)$ che soddisfano la relazione; il problema può essere ridotto a quello dell'intersezione tra la retta individuata dalla direzione (ϑ, φ) e il piano $z = z_c$. La quantità $z_c \tan\vartheta$ è la proiezione del segmento $\overline{OP^*}$ sul piano Oxy: il suo prodotto per $\cos\varphi$ e $\sin\varphi$ ci dà rispettivamente le coordinate x^* e y^*. L'altezza z^* è banalmente quella del rivelatore z_c.

È importante notare che il problema non distingue tra "sopra" e "sotto", per cui un raggio rivolto verso il simmetrico di C rispetto all'origine sarà ugualmente soluzione della disequazione. Per tener conto di ciò basterà dividere per 2 il conteggio totale dei punti che intersecano il rivelatore.

2.1.2 Sorgente estesa

Per estendere quanto sviluppato al punto precedente alla sorgente estesa, è sufficiente fare una semplice traslazione: generato un punto $S \equiv (x_s, y_s, 0)$ arbitrario nella sorgente circolare, è possibile calcolare la posizione relativa del centro C' che avrà quindi coordinate $C' \equiv (x_c - x_s, y_c - y_s, z_c)$. Fatto ciò possiamo generare una direzione casuale e, con gli stessi passaggi del punto precedente, trovare gli eventuali punti di intersezione nel sistema di riferimento del punto sorgente. Trovati tali punti, sarà immediato riportarli nel sistema di riferimento del centro della sorgente con la traslazione inversa. Si trova in particolare

$$\begin{cases} x^* = x_s + z_c \tan\vartheta \cos\varphi \\ y^* = x_s + z_c \tan\vartheta \sin\varphi \\ z^* = z_c \end{cases} ,$$

dove ϑ e φ sono ancora le direzioni generate casualmente nel sistema del punto sorgente.

2.2 Codice e generazione dei punti

Come per la simulazione dell'assorbimento di particelle cariche esposto al capitolo §1, anche per questo progetto l'interezza del codice è reperibile sul mio profilo GitHub.[2]

Il programma può essere lanciato scegliendo fin da subito il numero di punti da generare, le dimensioni della sorgente e del rivelatore e la posizione di quest'ultimo. Dopo aver verificato la validità dell'input e caricati i dati in memoria, il programma

[2]Repository: https://github.com/ImAure/geometric-acceptance

genera un punto casuale come sorgente in coordinate polari piane e una direzione casuale per la radiazione in coordinale polari sferiche.

```
int rand_polar2D(polar2D_t *ptr, double radius) {
    if (ptr == NULL) return -1;
    ptr->rho = (radius < 0) ? 1 : (sqrt((double)rand() / RAND_MAX) *
radius);
    ptr->phi = (radius == 0) ? 0 : ((double)rand() / RAND_MAX) * (2 *
M_PI);
    return 0;
}

int rand_polar3D(polar3D_t *ptr, double radius) {
    if (ptr == NULL) return -1;
    ptr->rho = (radius < 0) ? 1 : (cbrt((double)rand() / RAND_MAX) *
radius);
    if (radius == 0) {
        ptr->theta = 0;
        ptr->phi   = 0;
    } else {
        ptr->theta = acos(1 - 2 * ((double)rand() / RAND_MAX));
        ptr->phi   = ((double)rand() / RAND_MAX) * (2 * M_PI);
    }
    return 0;
}
```

Affinché la distribuzione sia uniforme, nel caso bidimensionale viene generato uniformemente $\varrho^2 \in [0;1]$, ne viene presa la radice quadrata e il risultato è moltiplicato per il raggio massimo. Nel caso tridimensionale la funzione fa la stessa cosa con ϱ^3 e la radice terza, mentre come detto prima viene generato[3] $\cos\vartheta \in [-1;1]$ e ne viene preso l'arcocoseno per avere ϑ.

Entrambe le funzioni ammettono come input anche un raggio massimo negativo: in quel caso il raggio verrà posto uguale a 1 e verrà generata solo una direzione. Questo comportamento è sfruttato nella generazione casuale della radiazione, in cui la chiamata a funzione avviene come `rand_polar3D(&ray, -1)`.

La funzione `intercept()` è pensata appositamente per verificare l'intersezione del raggio nel caso di una sorgente puntiforme posta nell'origine. Di conseguenza viene calcolata la posizione relativa del rivelatore (x'_c, y'_c, z_c) e data in input alla funzione insieme alla direzione. La funzione verifica quindi la disequazione (2.5) e in caso di successo aggiorna il conteggio e salva i dati in un file che raccoglie solo i punti del rivelatore dove si è verificata un'intersezione e i punti sorgente da cui si è originato il raggio.

[3] Per essere precisi, il coseno è generato tra 1 e -1 piuttosto che tra -1 e 1. Questo perché per $\vartheta = 0$ si ha $\cos\vartheta = 1$ e per $\vartheta = \pi$ si ha $\cos\vartheta = -1$

A Codice per Arduino

Riporto in questa appendice il codice utilizzato per l'esperienza descritta al Capitolo §??.

```
#define SENSOR_PIN  A0
#define MAX_READ    1023.0
#define MAX_V       5.0
#define N           20
#define A           0.5
#define B           100

void setup() {
    // set SENSOR_PIN (A0) as analog input pin and initialize the
serial monitor
    pinMode(SENSOR_PIN, INPUT);
    Serial.begin(9600);
}

float         tmp;
int           i = 0;
unsigned long dt = 0;

void loop() {
    // print the time since the program started
    Serial.print(dt = millis());

    // measure the temperature 20 times and calculate the average
    for (i = 0, tmp = 0; i < N; i++) {
        tmp += (((float)analogRead(SENSOR_PIN) / MAX_READ) * MAX_V -
A) * B;
    }
    tmp = tmp / N;

    // print the measured temperature next to the corresponding time
    Serial.print(", ");
    Serial.println(tmp);

    // keep waiting until 30 seconds have passed
    dt = millis() - dt;
    delay(30000 - dt);
}
```

Bibliografia

[1] S. Navas et al. «Review of Particle Physics». In: *Phys. Rev. D* 110.3 (2024), p. 030001. DOI: 10.1103/PhysRevD.110.030001. URL: https://link.aps.org/doi/10.1103/PhysRevD.110.030001.