

# Guión das prácticas no IGFAE

Prácticas optativas de Grao · Curso 23-24  
Universidade de Santiago de Compostela

## Índice

<b>1. Presentación</b>	<b>1</b>
1.1. Obxectivos . . . . .	1
1.2. Motivación . . . . .	1
<b>2. Principios físicos</b>	<b>2</b>
2.1. Un detector gasoso . . . . .	2
2.2. Perdas de enerxía na materia . . . . .	2
2.3. Resolución en enerxía . . . . .	3
2.4. Cinemática . . . . .	3
<b>3. Para comezar</b>	<b>4</b>
3.1. Introducción a ROOT . . . . .	4
3.2. Github . . . . .	4
<b>4. Programa</b>	<b>5</b>
4.1. Primeira semana . . . . .	5
4.2. Segunda semana . . . . .	6
4.3. Terceira semana . . . . .	7

## 1. Presentación

### 1.1. Obxectivos

O obxectivo destas prácticas vai ser reproducir a resposta real do detector *Active TARget and Time Projection Chamber* (ACTAR TPC) nun computador de cara a preparar a campaña experimental de 2025. Simularase unha reacción nuclear de cuxos resultados avaliaremos a resolución na reconstrución de ángulos.

Abarcaremos varias áreas:

- **Programación:** Aprenderemos a programar en C++ con ROOT. Utilizaremos programas sinxelos para que a adaptación sexa o máis liviana posíbel e fomentaremos o recurso á documentación na web.
- **Detectores:** ACTAR TPC é un detector gasoso moi recente que abre todo unha nova ventá de posibilidades no eido das reaccións nucleares. Introduciremos os conceptos básicos dun detector gasoso e complementaremos cos de detectores de Silicio.
- **Teoría:** Expoñeremos as leis básicas da cinemática que rixen calquera reacción entre partículas e introduciremos os fundamentos da interacción radiación-materia, básicos para a detección experimental.

Iremos adaptando o guión das prácticas en función das necesidades e do tempo dispoñíbel. Gran parte do código computacional está xa listo, e o que se requirirá será facer uso del en programas máis simples.

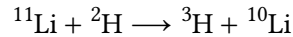
### 1.2. Motivación

En 2025 levarase a cabo un experimento coa reacción  $^{11}\text{Li}(d,t)^{10}\text{Li}$  para estudar a estrutura nuclear do  $^{10}\text{Li}$ . En particular, este tipo de reaccións de transferencia dun nucleón permiten acceder de forma pormenorizada á información espectroscópica do núcleo resultante; neste caso, co obxectivo principal de estudar o estado fundamental do  $^{10}\text{Li}$ .

Para lograr este obxectivo, necesitamos varias compoñentes:

- **Detector:** O gas que encherá ACTAR TPC será unha mestura de  $\text{D}_2$  e  $\text{iC}_4\text{H}_{10}$  (95 % e 5 %, respectivamente) a unha presión de 900 mbar. Será o  $\text{D}_2$  o que provea os  $d$  da reacción.

- **Reacción:** A xa mencionada:



- **Feixe:** O  $^{11}\text{Li}$  terá unha enerxía de 7,5 AMeV (é dicir, 82,5 MeV).

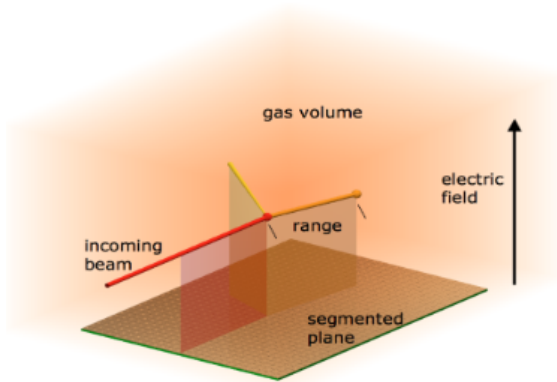
## 2. Principios físicos

### 2.1. Un detector gasoso

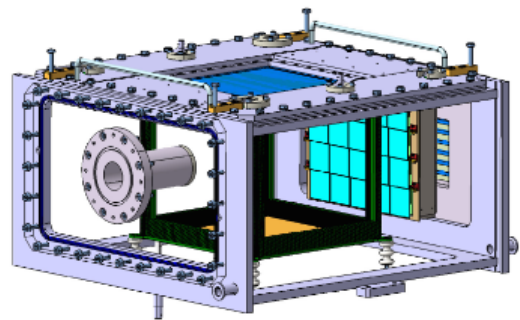
ACTAR TPC segue os principios de funcionamento dunha *time projection chamber*: as partículas ao pasaren polo gas ionizan os átomos, liberando electróns que **derivan** baixo aplicación dun campo eléctrico. Esta carga é recollida nun sensor amplamente fragmentado (), permitindo *seguir* as partículas no seu traspaso dentro do medio en 2D.

Pero é máis: pódese engadir a terceira coordenada coñecendo o tempo que lle leva aos electróns chegar ao *pad plane*, pois a **velocidade de deriva** é constante. Deste xeito, o seguimento das partículas faise en 3D.

Finalmente, o principio de *active target* engade a vantaxe de ser o gas o propio albo da reacción: este actúa como medio de reacción e de detección. O seguinte esquema ilustra este modo de funcionamento.



**Figura 1:** Esquema de funcionamento dunha TPC.



**Figura 2:** Deseño do detector ACTAR TPC. A rexión de detección é a parte negra interna.

Para a simulación vamos necesitar poñerlle números ao detector:

- O tamaño da zona de detección (Figura 2) é de  $256 \times 256 \times 255 \text{ mm}^3$ .
- O criterio de dimensións é que o beam vai no eixo X, e a dimensión vertical é Z.
- Os detectores auxiliares poden poñerse arredor desta zona nos diferentes lados. Nós usaremos algo parecido ao da figura, cun detector de Si (como os azuis) cara adiante.
- Situremos un só detector de Si, cun tamaño de  $80 \times 50 \times 1,5 \text{ mm}^3$  a unha distancia de 10 cm do *pad plane*.

### 2.2. Perdas de enerxía na materia

As partículas ao propagarse a través do gas interaccionarán co campo de Coulomb dos átomos deste, perdendo enerxía e liberando electróns. Esta perda de enerxía en cada interacción é moi pequena, pero como é moi elevado o número de colisións no gas (é, polo tanto, un proceso estocástico), chega a ser moi importante. Nunha aproximación continua, está dada pola **ecuación de Bethe-Bloch**:

$$-\frac{dE}{dx} \cong \frac{4\pi e^4}{m_e} \left( \rho \frac{N_A}{M} \right) \frac{z^2 Z}{v^2} \ln \frac{2m_e v^2}{I}$$

Depende esencialmente da partícula a considerar ( $Z$  e  $A$ ) e do gas (a través de  $\rho$ ). É interesante definir:

- **Rango ( $R$ ):** É o camiño percorrido por unha partícula ata que *case* se detén. É función da enerxía inicial e do material a atravesar. Existe unha **relación unívoca**  $E \iff R$  que imos utilizar para estimar a perda da enerxía en función da distancia percorrida pola partícula.
- **Straggling:** Como a perda de enerxía é un **proceso estatístico**, dúas partículas coa mesma enerxía inicial non van depositar a mesma  $\Delta E$ . O *straggling* mide, dalgunha forma, a  $\sigma$  desta distribución.

Esta información está tabulada nun programa que se coñece como **SRIM** e que utilizaremos sistematicamente nas prácticas. Basicamente, contén táboas cos parámetros que vamos necesitar para os cálculos: enerxías, rangos e stragglings.

Unha pequena apreciación: SRIM vainos dar o *straggling* en posición, mais nós deberémolo converter a en enerxía. Aquí o algoritmo que imos usar:

1. Calculamos  $R_{ini}$  coa enerxía inicial, xunto co seu *straggling*.
2. Sabendo que a partícula percorre unha distancia  $d$  o rango nese punto será  $R_L = R_{ini} - d$ . Avaliamos o *stragg*. para este valor.
3. O *straggling* na distancia estará **correlacionado con ambos**, e sabendo que  $u^2(R_{ini}) = u^2(R_{Left}) + u^2(d)$ , despexamos  $u(d)$ .
4. Aleatorizamos o valor de  $d$  cunha gaussiana centrada no seu valor e de  $\sigma = u(d)$ . Recalculamos o valor de  $R_{Left} \rightarrow R'_{Left}$  coa nova  $d \rightarrow d'$ .
5. Con este novo rango calculamos a enerxía final, efectivamente implementando o *straggling*!

### 2.3. Resolución en enerxía

Complementariamente acostuman situarse detectores de Si que nos van permitir medir a enerxía das partículas á súa saída do *pad plane* (segundo os mesmos principios que na anterior Sección). Un parámetro importante destes é a **resolución en enerxía**, definida como a capacidade para *resolver* enerxías depositadas distintas:  $R = \text{FWHM}/E$ .

Para o noso experimento, esta resolución está tabulada en 50 keV a 5,5 MeV e podemos extrapolala ao resto de enerxías coa función:

$$R = \frac{2,35\sigma}{E} = \frac{K}{\sqrt{E}} \implies \sigma = \frac{K}{2,35} \sqrt{E} = \frac{0,0213 \text{ MeV}}{2,35} \sqrt{E}$$

Isto vai significar que a perda de enerxía  $\Delta E$  que calcules nos Si deberala aleatorizar cunha gaussiana de  $\sigma$  obtida coa anterior fórmula.

### 2.4. Cinemática

Todo proceso de interacción entre partículas podes escribirse en termos de variables cinemáticas (enerxías e ángulos) usando as **leis de conservación** de enerxía e de momento. Esta pode ser descrita no sistema LAB ou no CM (mediante unha transformación Lorentz a un sistema con 3-momento nulo en ambas canles de entrada e saída) do seguinte xeito:

$$\begin{aligned} \text{No LAB: } p_1 &= (E_1, \mathbf{p}_1); p_2 = (m_2, \mathbf{0}); p_3 = (E_3, \mathbf{p}_3); p_4 = (E_4, \mathbf{p}_4) \\ \text{No CM: } p'_1 &= (E'_1, \mathbf{p}'_1); p'_2 = (E'_2, -\mathbf{p}'_1); p'_3 = (E'_3, \mathbf{p}'_3); p'_4 = (E'_4, -\mathbf{p}'_3) \end{aligned}$$

É estándar asumir unha partícula *target* (a 2) en repouso. O que vamos necesitar para a simulación é:

- Partiremos da enerxía cinética  $T_1$  do feixe (1) no LAB.

- Calcularas a transformación Lorentz da canle de entrada ao CM.
- Repartiremos a  $E_{CM}$  (enerxía total no centro de masas) entre as partículas de saída cun  $\theta_{CM}$  e  $\phi_{CM}$  aleatorios.
- Finalmente, recuperaremos as enerxías de saída de ambas partículas no LAB coa transformación inversa.
- Tamén necesitaremos os ángulos no laboratorio,  $\theta_{Lab}$  e  $\phi_{Lab} = \phi_{CM}$ .

Algunhas fórmulas que che poden ser interesantes para unha transformación Lorentz en X (o feixe móvese nesa dirección; é un convenio):

$$\begin{aligned} E' &= \gamma(E - \beta p_x), & p'_x &= \gamma(p_x - \beta E) \\ E &= \gamma(E' + \beta p'_x), & p_x &= \gamma(p'_x + \beta E') \end{aligned}$$

Onde as variables primadas indican que son no CM. Cando transformes ao final do CM ao LAB terás en conta  $\theta_{CM}$ :  $p'_{4,x} = |\mathbf{p}'_4| \cdot \cos(\theta_{CM})$ . O ángulo  $\phi_{CM}$  non aparece de momento posto que é transversal á transformación.

### 3. Para comezar

#### 3.1. Introducción a ROOT

O primeiro de todo é aprender a programar en C++ con [ROOT](#). No [repositorio de Github](#), dentro da carpeta *Macros*, tes algúns exemplos de como traballar con el. Comeza polo básico e despois vas ver como podes crear histogramas, gráficos e números aleatorios: o básico que necesitamos para estar prácticas.

Velaquí algunha información básica que necesitas para comezar:

- ROOT é un código desenvolto polo CERN que permite executar código de C++ sen compilar, mediante o que se coñece como **macros**:

```
$ root -l NomeDoMacro.cxx
```

Vai executar a función chamada *NomeDoMacro* dentro dese ficheiro. Polo tanto, debes nomear o ficheiro igual que a función principal que conteña dentro. Isto non impide que definas outras funcións dentro do mesmo.

- Debes saír sempre da sesión unha vez o programa termina, escribindo `.q`
- ROOT provee de numerosas **clases** que realizan múltiples funcións: **TH1D** (histogramas), **TGraph** (gráficos), ... Sempre podes consular a documentación na web cando non saibas como se constrúe ou que funcións ten unha clase. Escribe no teu buscador favorito: *NomeDaClase root cern* e deberías ter a documentación no primeiro resultado.

#### 3.2. Github

Git é unha ferramenta de control de versións que che permitirá manter un historial do teu código, podendo volver atrás cando sexa necesario. Github é unha web para almacenar repositorios Git. Vamos utilizala para poder revisar o código a distancia.

Para iso, o primeiro é ter unha conta en [www.github.com](http://www.github.com). Tras unha configuración inicial un pouco tediosa, o plan de traballo é o seguinte, que se debe executar cando fagas cambios importantes ou cando remates a túa sesión de traballo:

1. `git add .` vai engadir todos os cambios dende o anterior *commit*
2. `git commit` abrirá un editor de texto no que crear unha mensaxe para informar dos cambios. Péchase con `Ctrl+S`, `Ctrl+X`
3. `git push` envía os cambios á nube

## 4. Programa

Desenvolveremos a simulación de forma modular, separando as distintas partes en diferentes macros e xuntando todo ao final.

A proposta é a seguinte:

- **Primeira semana:** Configuración do entorno e familiarización con ROOT. Macros para constuír gráficos e samplear en histogramas.
- **Segunda semana:** Resolución da cinemática e implementación da xeometría.
- **Terceira semana:** Introducción das perdas de enerxía e das distintas resolucións.
- **Cuarta semana:** Estudo sistemático da resolución en  $\theta_{Lab}$  para distintas configuracións da xeometría, gases, etc.
- **Quinta semana:** Posibilidade de engardir máis cousas á simulación (en función da evolución) ou comezo da memoria.

### 4.1. Primeira semana

Os obxectivos son os seguintes:

#### Cinemática

Resolverás as ecuacións de cinemática para obter as enerxías cinéticas e ángulos no LAB das dúas partículas finais, aínda que presentando máis atención á pesada. Faino de forma xeral, tal e como está escrito na Sección correspondente. O criterio de números é o seguinte:  $1 + 2 \rightarrow 3 + 4$ .

Na simulación, as variables que teremos que introducir nas fórmulas serán as masas, a enerxía do feixe e o ángulo  $\theta_{CM}$  (que será aleatorio, como veremos).

#### Macros iniciais

Quero que fagas dous macros onde me representes as seguintes cousas, usando a axuda do repositorio de Github:

1. Representacións **cinemáticas**: Creas un `TCanvas` con 3 *pads* e representas a cinemática para estas tres reaccións distintas:  $^{11}\text{Li}(d, p)^{12}\text{Li}$ ,  $^{11}\text{Li}(d, d)^{11}\text{Li}$  e  $^{11}\text{Li}(d, t)^{10}\text{Li}$ . Usa a mesma enerxía do feixe para todas de 7,5 AMeV.

A clase que vai facer iso é `ActPhysics::Kinematics`. O criterio que debes usar para chamar ao constructor é o seguinte: ("feixe", "target", "lixreira", "pesada", `Tbeam`), que configura a equivalencia coas fórmulas matemáticas do seguinte xeito:

$$1(\text{feixe}) + 2(\text{target}) \rightarrow 3(\text{lixreira}) + 4(\text{pesada})$$

Ademais, `GetKinematicLine3()` darache a representación  $T_3$  vs  $\theta_{Lab,3}$  (polo tanto, da partícula marcada como *lixreira*) e `GetKinematicLine4()` o mesmo pero da 4, a pesada. Representa ambas no mesmo *pad*.

Vas ver como as formas son moi diferentes, o que forza a deseñar experimentos moi distintos en función da reacción a medir (ou un que sexa capaz de medir todas as canles de reacción).

2. **Sampleado** de variables: Para a simulación vamos necesitar samplear variables gaussianas e uniformes. Crea un macro con 3 histogramas (que vas representar en 3 *pads*) distintos nos que samplees as seguintes distribucións (podes xogar cos parámetros como queiras);

- Gaussianas: `gRandom->Gaus(mean, sigma)`

- Uniforme: `gRandom->Uniform(x0, x1)`
- O ángulo polar  $\theta$  é especial se queres obter unha **distribución uniforme**:

$$\theta = \arccos(\text{gRandom->Uniform}(-1, 1))$$

Nota que este ángulo vai estar en radiáns. Para converter a graos, debes multiplicar polo factor de conversión correspondente. Se non o queres escribir, podes usar `TMath::RadToDeg()` (a inversa é `TMath::DegToRad()`) de ROOT.

## 4.2. Segunda semana

### Representación gráfica da perda de enerxía

Na carpeta **SRIM** do repositorio tes táboas de perda de enerxía calculadas con SRIM. Se abres unha vas ver toda a información que conteñen:

- Información do material no que se calculou a perda.
- O mesmo sobre a partícula que o vai atravesar.
- Múltiples columnas coa  $E$ , o rango, o *straggling* en rango e o  $dE/dx$ .

Para ler e traballar con esta clase vas utilizar a clase `ActPhysics::SRIM`, tal e como tes no macro de exemplo (`Macros/srim.cxx`). As funcións básicas que necesitas son estas:

- `EvalDirect()`: obtén o rango para unha enerxía dada.
- `EvalInverse()`: enerxía para un rango dado.
- `Slow()`: para unha enerxía inicial, calcula a enerxía final tras atravesar un espesor de material a especificar. Tamén lle podes dar un ángulo, de xeito que a lonxitude efectiva é:  $l_{\text{eff}} = l / \cos \theta$ . O ángulo debe especificarse en radiáns.
- `EvalInitialEnergy()`: cos mesmos argumentos que a anterior función, fai o proceso inverso: calcula a enerxía inicial dada unha enerxía final, lonxitude e opcionalmente ángulo.

En todas as funcións o primeiro argumento é unha *string* que especifica a táboa a usar, anteriormente lida con `ReadTable()`.

O que quero que fagas é que plotees a perda de enerxía no gas en función da distancia. Para iso, escolle unha táboa (por exemplo,  $^{11}\text{Li}$  no gas) e asume unha enerxía inicial. Calcula o rango da partícula a esa enerxía.

Despois, discretiza esa distancia nun intervalo suficientemente pequeno e nun bucle *for*, utilizando a función `Slow`, calculas a perda de enerxía (como a diferenza entre as iteracións). Representa iso en dous **TGraphs**: un de perda de enerxía vs distancia e outro de perda de enerxía vs enerxía.

Vas ver unha característica moi importante no gráfico: a medida que a partícula se para, a perda de enerxía aumenta: é o que se coñece como **pico de Bragg**. Estas dúas características (o perfil de perda de carga e o pico de Bragg) permiten identificar as partículas nun detector, xa que son unha pegada única de cada partícula!

### Valicación da cinemática

Crea un macro sinxelo no que definas as funcións necesarias para calcular a cinemática. Mira o exemplo `Macros/funcs.cxx` para ver como podes devolver varios parámetros nunha función e escolle a forma que máis che guste.

Unha vez fagas isto, no propio macro vas validar que todo funcione ben. Escolle a reacción  $^{11}\text{Li}(d, t)^{10}\text{Li}$  a 7,5 AMeV e segue os seguintes pasos:

- Crea un **TGraph**.
- Nun bucle for, vas percorrer unha variable  $\theta_{CM} \in [0, 180]^\circ$ .
- Utiliza as túas funcións para calcular  $T_4, \theta_{Lab,4}$ .
- Enche o gráfico.

Representa isto nun **TCanvas** e **superimpón** a cinemática de `ActPhysis::Kinematics::GetKinematicLine4()`, tal e como aprendiches a facer na anterior semana. Debería quedar exactamente igual!

### 4.3. Terceira semana

#### Comezo da simulación

Unha vez aprendido o básico, vamos comezar co macro de simulación. O primeiro é adaptar o que acabas de aprender de sampleado:

1. Define o tamaño de ACTAR ao comezo do macro en mm. Tes os tamaños ao principio deste guión. En cada iteración, nunha variable tipo **XYZPoint** garda:
  - X: uniforme na dirección do feixe.
  - Y e Z: gaussianas de media o centro do detector e de  $\sigma$  uns 5 mm. Os feixes de partículas creados nos aceleradores non son perfectos e teñen unha determinada anchura espacial. De momento, asumimos un caso sinxelo de dispersión gaussiana dun feixe ben centrado no detector.

Podes crear puntos e vectores no espazo coas clases definidas en `#include "Math/Point3D.h"` e `#include "Math/Vector3D.h"`. Podes acceder a elas como `ROOT::Math::XYZPoint` ou `ROOT::Math::XYZVector`.

Valida que todo vai ben: crea dous histogramas 2D para representar isto:

- Un onde representes Y vs X.
  - Outro no que representes Z vs Y.
2. Fai o mesmo para os ángulos no CM: samplea e crea histogramas para representalos.
  3. O seguinte será usar a clase de **cinemática** que creaches para obter as variables da partícula pesada. Cos ángulos sampleados e coa enerxía do feixe, calcula  $T_4$  e  $\theta_4$ .

Representa nun histograma 2D que todo vai ben: nun plot de enerxía fronte a ángulo deberías reproducir a cinemática teórica.

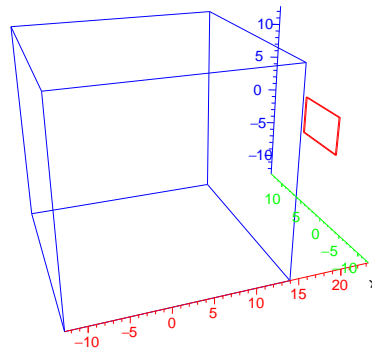
4. A continuación, vamos implementar a xeometría do silicio que vai medir esta partícula pesada. Na variable de tipo `ActSim::Geometry` está implementada a xeometría do detector, tal e como se ve na seguinte [Figura 3](#).

Esta clase contén todos os tamaños necesarios para simular a xeometría, cos seguintes valores

- Os tamaños da TPC normal
- Os do Si, de  $80 \times 50 \times 1,5 \text{ mm}^3$
- Unha distancia de 10,4 cm

A función que imos utilizar é `PropagateTrackToSiliconArray()`. Os argumentos que necesitas son uns cantos:

- O **vértice**, tal e como sampleaches.
- A **dirección** da partícula. Debes definila do seguinte xeito:



**Figura 3:** Esquema da xeometría: en azul, a TPC; en vermello, o silicio.

```
ROOT::Math::XYZVector direction {TMath::Cos(theta4), TMath::Sin(theta4) * TMath::Sin(phi4),
                                  TMath::Sin(theta4) * TMath::Cos(phi4)};
```

- O índice do muro de silicions, xa que pode haber varios. Como só hai un, pasa un 0.
- Os seguintes argumentos van **devolver** o resultado da operación, e deberás pasar as variables obrigatoriamente:
  - Un **bool** co lado. Ignoraremos isto de momento.
  - A distancia dende o vertice ao punto de impacto, en **cm**. Debes convertela a mm. É a variable máis importante.
  - O tipo de silicio. Só hai un, así que ignorarémo.
  - O índice do silicio. Moi importante tamén: se a partícula non chega ao silicon, vale -1.
  - Un **XYZPoint** co punto de impacto. Ignorarémolo.

Deste xeito, só os eventos que cheguen xeometricamente ao silicio terán un valor de `silIndex != -1`, e o máis importante, terás a distancia que teñen que viaxar para chegar alí.

A primeira vez que necesites usar a xeometría debes creala. Hai un macro na carpeta *Geo/Build.cxx* que tes que executar só unha vez. Isto vai crear un ficheiro *.root* cos parámetros necesarios que despois lerás como: `geo.ReadGeometry("./Geo/", "simple");`

5. Finalmente, para os eventos que chegan, representa a enerxía  $T_4$  na entrada do silicio: é dicir, usa a variable de tipo **SRIM** para reducir a perda de enerxía no gas. Representaa respecto do ángulo nun histograma 2D.

Para iso, terás que usar a táboa de enerxía correspondente. O que che recomendo é ler a táboa do  $^{12}\text{Li}$  e que a chames *heavy*, para non evitar malentendidos despois, posto que usaremos as táboas de varias partículas.