PERMANÊNCIA RADIOATIVA EM CHERNOBYL

KALIMERA

Cálculo Numérico

PARTICIPANTES DO GRUPO:

Felipe Duarte

João Fahning

Tiago Trindade

Nicholas Rodrigues

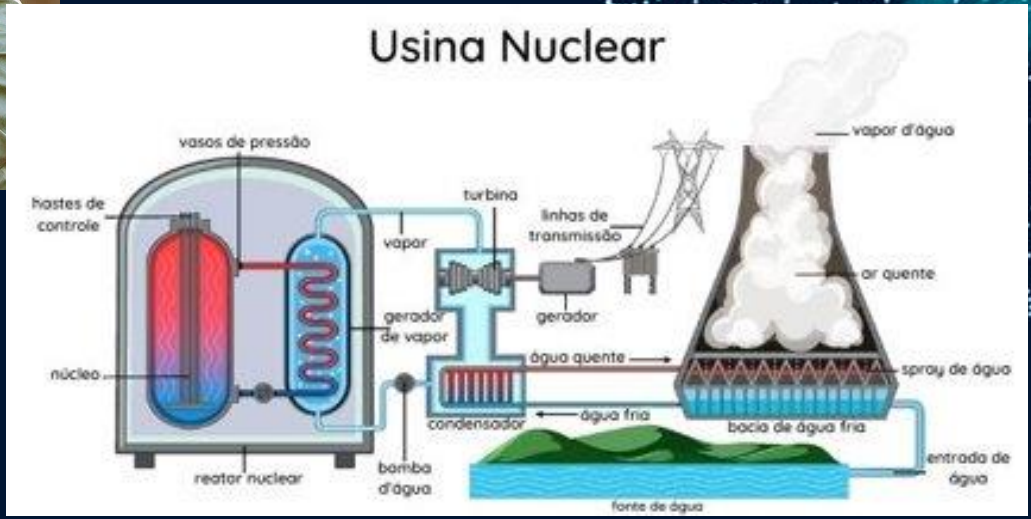
Thiago Leal

Fernando d’Ávila

Euro Da Cunha

Renan Gondim

INTRODUÇÃO

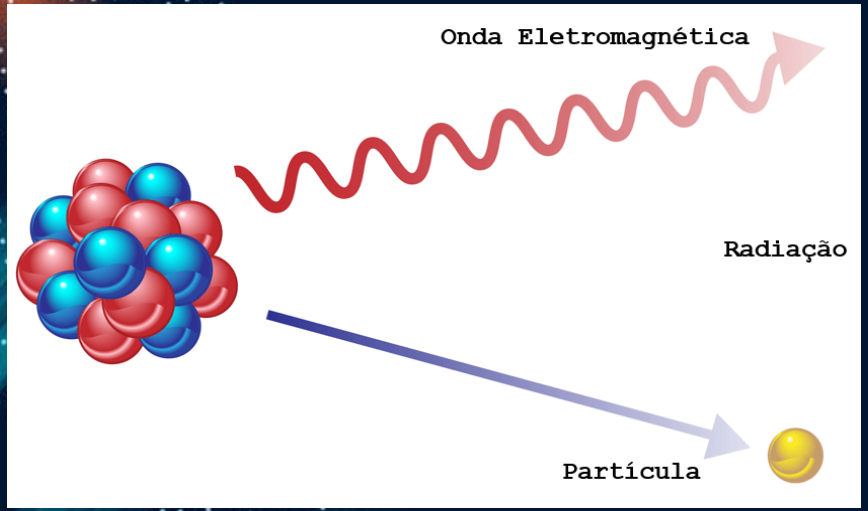

Usina Nuclear
vasos de pressão
hastes de controle
núcleo
reator nuclear
vapor
turbina
gerador de vapor
linhas de transmissão
gerador
vapor d'água
ar quente
água quente
spray de água
condensador
água fria
bacia de água fria
bomba d'água
entrada de água
fonte de água

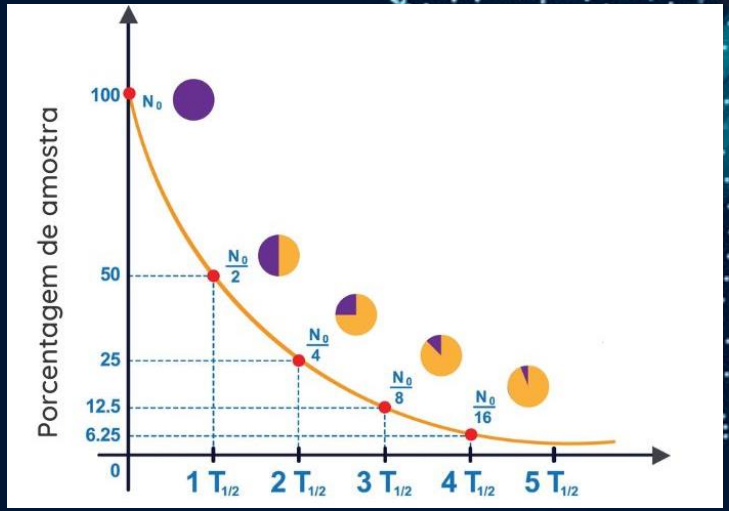
O CRESCENTE DESAFIO DOS RESÍDUOS RADIOATIVOS

Aumento dos resíduos radioativos devido a aplicações médicas, industriais e nucleares

Necessidade crucial de abordar o decaimento radioativo para garantir a segurança em seu manuseio

COMPREENDENDO O DECAIMENTO RADIOATIVO





Núcleos instáveis que emitem partículas ou radiação para atingir estabilidade.

A MATEMÁTICA DO DECAIMENTO

A equação utilizada para modelar o decaimento radioativo é a equação da lei do decaimento exponencial, que descreve a taxa de decaimento de uma substância radioativa ao longo do tempo. A equação é dada por:

$$N(t) = N_0 \cdot e^{-\lambda t}$$

onde:

$N(t)$ é a quantidade da substância radioativa no tempo t.

N_0 é a quantidade inicial da substância no tempo $t = 0$.

λ é a constante de decaimento.

t é o tempo decorrido.

```python
import numpy as np
import matplotlib.pyplot as plt

lambda_cesium_137 = 0.0230433552
lambda_strontium_90 = 0.0140597
initial_quantity = 100
time_step = 1
total_time = 500

quantities_cesium = [initial_quantity]
quantities_strontium = [initial_quantity]

for i in range(total_time):
    new_quantity_cesium = quantities_cesium[-1] * np.exp(-lambda_cesium_137 * time_step)
    new_quantity_strontium = quantities_strontium[-1] * np.exp(-lambda_strontium_90 * time_step)
    quantities_cesium.append(new_quantity_cesium)
    quantities_strontium.append(new_quantity_strontium)
```

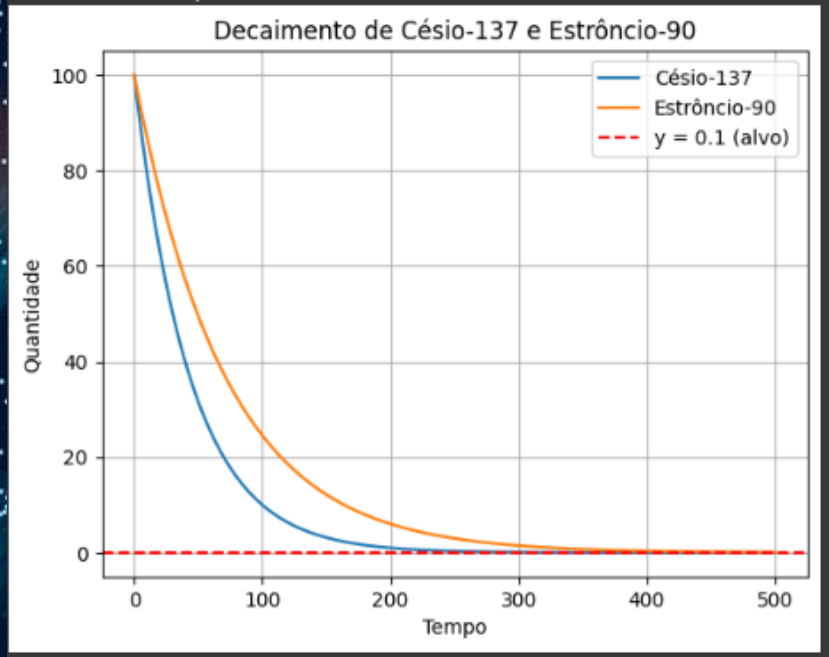
MERGULHANDO EM NÚMEROS

```python
# definição do método da bissecante
def bisection_method(func, a, b, target_value, tolerance=1e-6, max_iterations=100):
    if np.sign(func(a) - target_value) == np.sign(func(b) - target_value):
        return None

    for _ in range(max_iterations):
        c = (a + b) / 2
        f_c = func(c) - target_value

        if np.abs(f_c) < tolerance:
            return c

        if np.sign(func(a) - target_value) != np.sign(f_c):
            b = c
        else:
            a = c

    return None
```

```python
# método da bissecante para encontrar o ponto em que o gráfico toca y = 0.1
target_value = 0.1
result = bisection_method(lambda x: np.interp(x, np.arange(total_time + 1), quantities_cesium), 0, total_time, target_value)

if result is not None:
    print(f"Valor encontrado para Césio-137: {result}")
else:
    print(f"Não foi possível encontrar um valor que resulte em {target_value} para Césio-137.")

result_strontium = bisection_method(lambda x: np.interp(x, np.arange(total_time + 1), quantities_strontium), 0, total_time, target_value)

if result_strontium is not None:
    print(f"Valor encontrado para Estrôncio-90: {result_strontium}")
else:
    print(f"Não foi possível encontrar um valor que resulte em {target_value} para Estrôncio-90.")


x_values = np.arange(total_time + 1)
plt.plot(x_values, quantities_cesium, label="Césio-137")
plt.plot(x_values, quantities_strontium, label="Estrôncio-90")
plt.axhline(y=target_value, color='r', linestyle='--', label='y = 0.1 (alvo)')
plt.title("Decaimento de Césio-137 e Estrôncio-90")
plt.xlabel("Tempo")
plt.ylabel("Quantidade")
plt.legend()
plt.grid(True)
plt.show()
```

MERGULHANDO EM NÚMEROS

```
Valor encontrado para Césio-137: 299.774169921875
Valor encontrado para Estrôncio-90: 491.3177490234375
```



Portanto, por valores arbitrário que definimos, que seria 0.1 (ou 0,1%), em 300 anos que os níveis de Césio estarão aceitáveis e e, 491 que os níveis de estrôncio estarão aceitáveis.

LIMITES NA MODELAGEM

- **Limites da Equação**
- **Considera apenas decaimento alfa, beta e gama**
- **Fatores Externos não contemplados**

COMPUTAÇÃO PARA UM MELHOR ENTENDIMENTO

- **Criação de um modelo simplificado para representar como os isótopos césio-137 e estrôncio-90 se dispersam no ambiente**

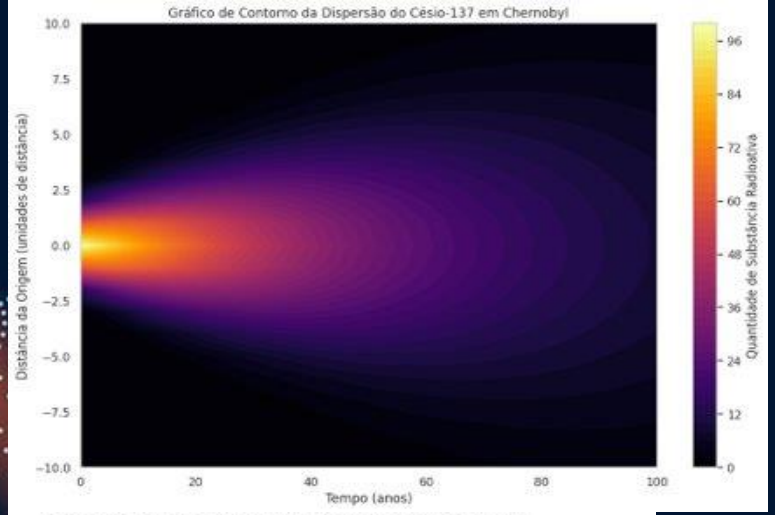
Parametros da Dispersao(em ano/unidade de espasco)

```
[ ]  dispersion_mean = 0
     dispersion_std_dev_initial = 1
     dispersion_spread_rate = 0.05
```

Simulação da dispersão dos isótopos no ambiente

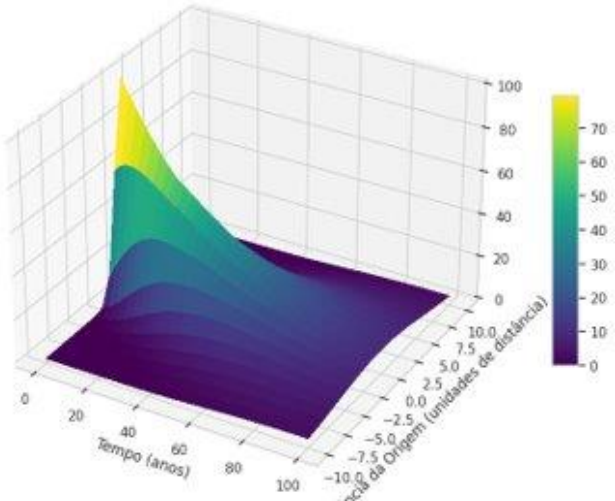
```
[ ]  def simulate_dispersion(quantities, dispersion_std_dev_initial, dispersion_spread_rate, total_time):
         dispersion_data = []
         for i in range(total_time + 1):
             std_dev = dispersion_std_dev_initial + i * dispersion_spread_rate
             dispersion = quantities[i] * np.exp(-(np.arange(-10, 11)**2) / (2 * std_dev**2))
             dispersion_data.append(dispersion)
         return dispersion_data

     dispersion_cesium = simulate_dispersion(quantities_cesium, dispersion_std_dev_initial, dispersion_spread_rate, total_time)
     dispersion_strontium = simulate_dispersion(quantities_strontium, dispersion_std_dev_initial, dispersion_spread_rate, total_time)
```

$$f(x) = a \cdot e^{-\frac{(x-b)^2}{2c^2}}$$




```python
from matplotlib import cm

def create_dispersion_matrix(dispersion_data):
    # dispersion_data esta na forma[tempo][distancia]
    # deve-se transpor para[distancia][tempo] para usar a função contourf
    return np.array(dispersion_data).T

dispersion_cesium_matrix = create_dispersion_matrix(dispersion_cesium)
dispersion_strontium_matrix = create_dispersion_matrix(dispersion_strontium)

def plot_contour(dispersion_matrix, isotope_name):
    plt.figure(figsize=(12, 8))
    X, Y = np.meshgrid(np.arange(total_time + 1), np.arange(-10, 11))
    cp = plt.contourf(X, Y, dispersion_matrix, cmap=cm.inferno, levels=50)
    plt.colorbar(cp, label="Quantidade de Substância Radioativa")
    plt.title(f"Gráfico de Contorno da Dispersão do {isotope_name} em Chernobyl")
    plt.xlabel("Tempo (anos)")
    plt.ylabel("Distância da Origem (unidades de distância)")
    plt.show()

plot_contour(dispersion_cesium_matrix, "Césio-137")
plot_contour(dispersion_strontium_matrix, "Estrôncio-90")
```

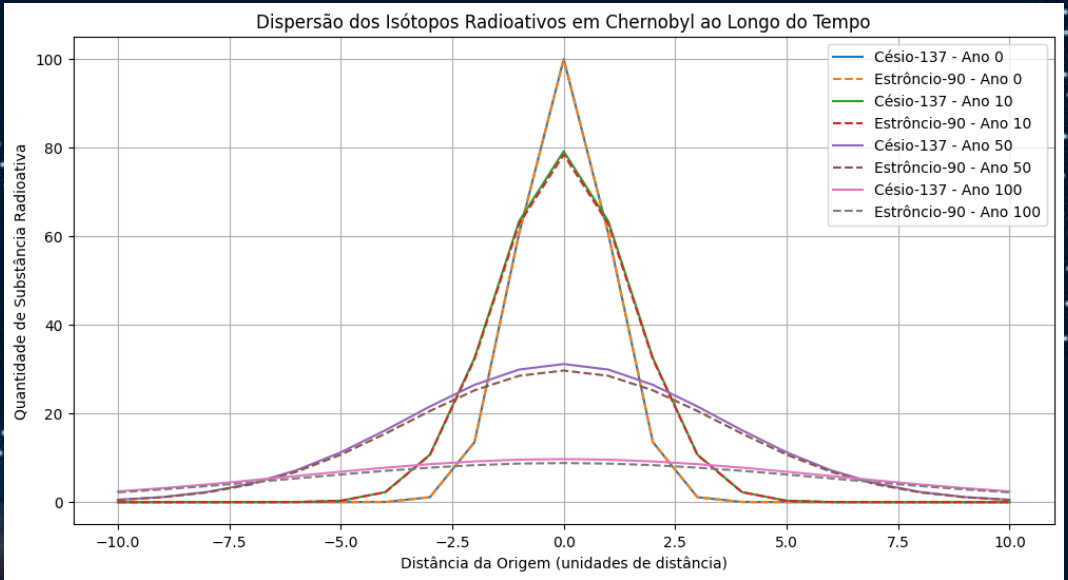
Gráfico de Superfície 3D da Dispersão do Césio-137 em Chernobyl




```python
from mpl_toolkits.mplot3d import Axes3D

def plot_3d_surface(dispersion_matrix, isotope_name):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_subplot(111, projection='3d')
    X, Y = np.meshgrid(np.arange(total_time + 1), np.arange(-10, 11))
    surf = ax.plot_surface(X, Y, dispersion_matrix, cmap=cm.viridis, linewidth=0, antialiased=False)
    plt.title(f"Gráfico de Superfície 3D da Dispersão do {isotope_name} em Chernobyl")
    plt.xlabel("Tempo (anos)")
    plt.ylabel("Distância da Origem (unidades de distância)")
    fig.colorbar(surf, ax=ax, shrink=0.5, aspect=10)
    plt.show()

plot_3d_surface(dispersion_cesium_matrix, "Césio-137")
plot_3d_surface(dispersion_strontium_matrix, "Estrôncio-90")
```

SIMULAÇÕES COMPUTACIONAIS PARA MODELAR DISPERSÃO

PLOTAGEM DE DISPERSÃO PARA DIFERENTES MOMENTOS DO TEMPO



MEDIDAS DE MANIPULAÇÃO

Importância de compreender o decaimento para o manuseio seguro de resíduos

Estabelecimento de critérios de segurança para manipulação sem riscos à saúde

Calculando o Tempo Adequado

Cálculo do tempo necessário para um manuseio seguro

Considerações sobre as partículas emitidas durante o decaimento

Aplicação no Cotidiano

RECAPITULANDO A JORNADA

Reforçando a relevância da compreensão do decaimento radioativo

O poder da modelagem matemática e computacional em decisões seguras

Abordagem às regulamentações governamentais e implicações práticas