Roteiro para Atividade Laboratorial: Ajuste e Modelagem de Perfis de Temperatura e Velocidade no MATLAB

Objetivo: Este experimento visa introduzir os alunos ao MATLAB, utilizando habilidades de programação para ajustar dados experimentais, calcular gradientes e realizar análises baseadas em modelos teóricos. O foco é desenvolver habilidades em manipulação de dados, ajustes de curvas e construção de gráficos para aplicações em física moderna

Parte 1: Ajuste do Perfil de Velocidade

Passos:

I) **Carregar os dados experimentais de velocidade e temperatura do link abaixo:**

https://fap.if.usp.br/~jhsevero/Fisica_Experimental_C_Semestral_2025/page-6/

Obs_01: Escolha os dados de acordo com seu grupo de trabalho. Para se certificar do número do seu grupo de trabalho consulte o endereço: https://fap.if.usp.br/~jhsevero/Fisica_Experimental_C_Semestral_2025/page-12/

Objetivo: Ajustar os dados experimentais de velocidade de rotação do plasma a polinômios de 4º e 5º graus, comparar a qualidade dos ajustes usando o coeficiente R^2 que mede a qualidade do ajuste. Gerar um gráfico com barras de erro, intervalos de confiança e legenda informativa.

II) **Carregar o matlab e crie um novo script chamado script_01.**

Obs_02: Vocês no final das atividades desse laboratório terão criado 4 scripts e todos eles devem ser anexados ao relatório final.

Obs_03: Comece sempre o script com as funções clc; clear; close all. Em matlab:

- **clear**: apaga todas as variáveis do workspace, liberando a memória que estava sendo utilizada por elas (ou seja, após usar clear, as variáveis não estarão mais disponíveis).
- **clc**: limpa a janela de comandos, removendo todo o texto exibido anteriormente.
- **close all**: fecha todas as janelas de figuras (plots) que estiverem abertas.

Esses comandos são frequentemente utilizados no início de scripts para "resetar" o ambiente, evitando a interferência de variáveis ou figuras anteriores.

III) **Carregar os dados experimentais:**

Matlab

% Vetores de entrada

```
r_Exp = [0; 0.01; 0.02; ... ]; % Raio experimental (m)
Vphi_Exp = [24.4725; 22.2933; ... ]; % Velocidade experimental (m/s)
Vphi_Exp_err = [0.9428; 1.8117; ... ]; % Incertezas experimentais (m/s)
```

IV) **Ajustar os dados a polinômios de 4º e 5º graus:**

Matlab

```
p4 = polyfit(r_Exp, Vphi_Exp, 4); % Ajuste de 4º grau
p5 = polyfit(r_Exp, Vphi_Exp, 5); % Ajuste de 5º grau
poly4_val = polyval(p4, r);       % Avaliação do polinômio de 4º grau
poly5_val = polyval(p5, r);       % Avaliação do polinômio de 5º grau
```

Obs_04: Para que você possa avaliar o ajuste polinomial, crie um vetor r com 1000 pontos. Para isso use a função linspace do Matlab. Para ver como escrever corretamente essa função vá até a Command Window e escreva help linspace. Você verá a seguinte informação: linspace(X1, X2, N)', tome X1=0; X2=0.18 e N=1000.

V) **Calcular o coeficiente R^2 para cada ajuste:**

Obs_05: O intervalo de confiança é calculado apenas para o modelo com maior R^2

Matlab

```
SSres4 = sum((Vphi_Exp - polyval(p4, r_Exp)).^2);
SSres5 = sum((Vphi_Exp - polyval(p5, r_Exp)).^2);
SStot = sum((Vphi_Exp - mean(Vphi_Exp)).^2);
R2_4 = 1 - SSres4/SStot;
R2_5 = 1 - SSres5/SStot;
```

VI) **Selecionar o melhor ajuste (maior R^2):**

Matlab

```
if R2_5 > R2_4
    best_poly = poly5_val; % Seleciona 5º grau se R² maior
else
    best_poly = poly4_val; % Seleciona 4º grau se R² maior
```

```
end
```

VII) Calcular o intervalo de confiança de 95% para o melhor modelo:

Matlab

```
RMSE_best = sqrt(SSres_best / (length(r_Exp) - grau)); % RMSE (Raiz do Erro Quadrático Médio) do melhor modelo
conf_range = 1.96 * RMSE_best; % Intervalo de 95%
upper_conf = best_poly + conf_range; % Limite superior
lower_conf = best_poly - conf_range; % Limite inferior
```

VIII) **Gerar o gráfico comparativo:**

Matlab

```
figure;
hold on;
% Intervalo de confiança (preenchimento)
fill([r; flipud(r)], [upper_conf; flipud(lower_conf)], 'b', ...
    'FaceAlpha', 0.1, 'EdgeColor', 'none');
% Dados experimentais com erro
errorbar(r_Exp, Vphi_Exp, Vphi_Exp_err, 'ko', 'LineWidth', 1.5, ...
    'MarkerFaceColor', 'w', 'CapSize', 0);
% Ajustes polinomiais
p1 = plot(r, poly4_val, 'b-', 'LineWidth', 2);
p2 = plot(r, poly5_val, 'r--', 'LineWidth', 2);

% Configurações do gráfico
xlabel('Raio (r) [m]', 'FontSize', 12);
ylabel('Velocidade de Rotação (V_{\phi}) [m/s]', 'FontSize', 12);
```

```
title('Perfil de Velocidade - TCABR', 'FontSize', 14);
legend([p1 p2], ...
    sprintf('Ajuste 4º Grau (R² = %.4f)', R2_4), ...
    sprintf('Ajuste 5º Grau (R² = %.4f)', R2_5), ...
    'Location', 'northeast');
grid on;
ylim([0 1.2*max(Vphi_Exp)]);
set(gca, 'FontSize', 12);
```

Saída Esperada:

- Gráfico contendo:

- Pontos experimentais com barras de erro.

- Curvas dos ajustes de 4º e 5º graus.

- Intervalo de confiança de 95% para o melhor modelo.

- Legenda com R^2 de cada ajuste.

- Resultados numéricos:

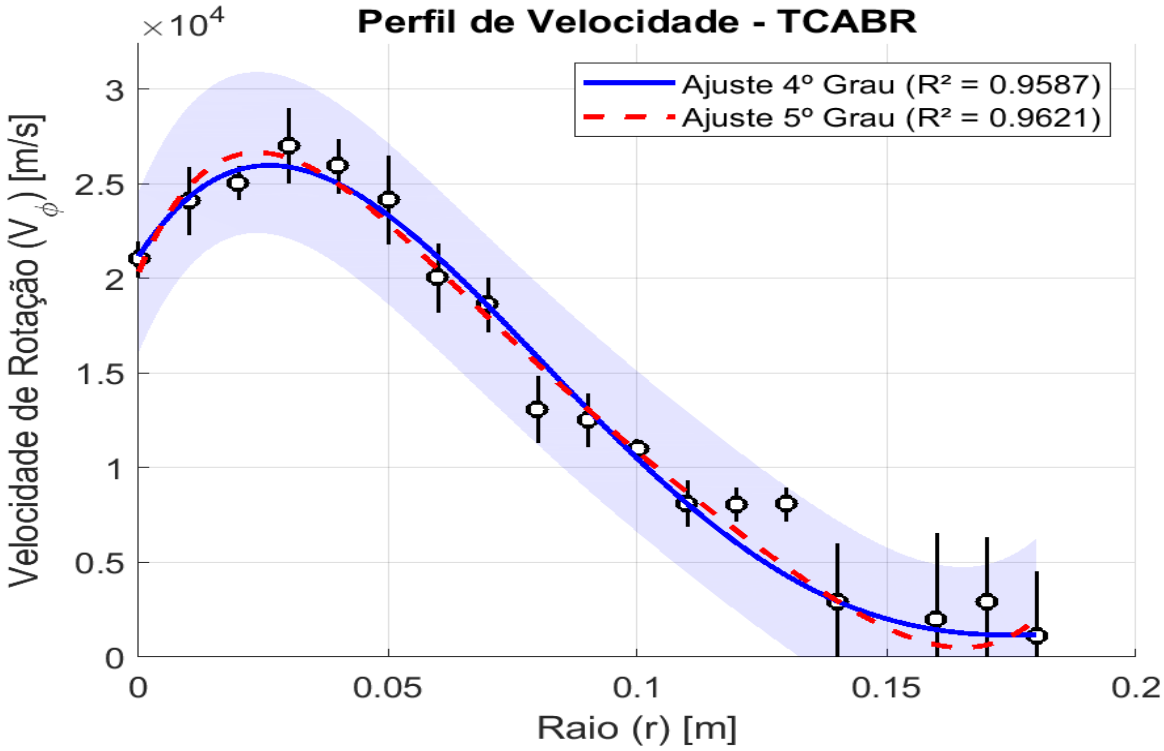
- Coeficientes dos polinômios (impressos via `fprintf`).

- Valores de R^2 e RMSE.

Grau do polinômio	R^2	$RMSE$	$a\,x^5$	$b x^4$	$c x^3$	$d x^2$	$e x^1$	f
4 Grau (a_4, b_4, c_4, d_4, e_4)								
5 Grau (a_5, b_5, c_5, d_5, e_5, f_5)								

Observações:

- O intervalo de confiança é calculado apenas para o modelo com maior R^2.



- O script fornecido já implementa todas as etapas acima. Certifique-se de incluí-lo no relatório.

……..

Parte 2: Ajuste do Perfil de Temperatura Iônica

Objetivo: Ajustar os dados experimentais de temperatura a uma função canônica do tipo $T_i(r) = (T_{\mathrm{i,max}} - T_{\mathrm{i,min}})(1-(r/a)^2)^n + T_{\mathrm{i,min}}$, calcular métricas de qualidade (R^2), $RMSE$) e gerar um gráfico com intervalo de confiança de 95%.

Passos para Implementação no MATLAB

I. Carregar Dados Experimentais e Definir Variáveis. Crie um novo script chamado script_02:

Matlab

```
clc; clear; close all;
```

% Dados experimentais (exemplo)

```
r_data = [0.1021, 4.5631, ..., 17]/100;      % Raio experimental (m)

Ti_reconstructed = [269.93, 212.63, ..., 32]; % Temperatura (eV)

Ti_data_err = 1*[12.54, 8.897, ..., 11];      % Incertezas (eV)


% Parâmetros físicos

a = 0.18;  % Raio menor do plasma (m)
```

II. Definir a Função Canônica

Matlab

```
% Função de ajuste: T_i(r) = (T_max - T_min)*(1 - (r/a)^2)^n + T_min

Ti_function = @(params, r) (params(1) - params(2)) .* (1 - (r/a).^2).^params(3) + params(2);
```

III. Ajustar Parâmetros com lsqcurvefit

Matlab

```
% Chute inicial e configurações

initial_guess = [270, 30, 3.5]; % [T_max, T_min, n]

options = optimset('Display', 'iter', 'MaxIter', 1000);
```

Obs_05: No site da disciplina (https://fap.if.usp.br/~jhsevero/Fisica_Experimental_C_Semestral_2025/resources/Franck-Hertz/PDF/lsqcurvefit.pdf) há um material explicativo sobre a função Matlab lsqcurvefit. Recomendo fortemente a leitura para compreender melhor o funcionamento dessa função.

```
% Otimização

[best_params, ~, residual, ~, ~, ~, J] = lsqcurvefit(...

    Ti_function, initial_guess, r_data, Ti_reconstructed, [], [], options);


% Resultados

T_i_max = best_params(1);
```

```
T_i_min = best_params(2);
n = best_params(3);
```

IV. Encontrar os Parâmetros Ajustados e o Jacobiano (J)

```
[best_params_temp, ~, residual, ~, ~, ~, J] = lsqcurvefit(...
    @(params, r) Ti_function(params, r), ...
    initial_guess_temp, r_data, Ti_reconstructed, [], [], options);
```

IX) Obter as Métricas

```
Ti_fit = Ti_function(best_params_temp, r_data);
SSres = sum((Ti_reconstructed - Ti_fit).^2);
SStot = sum((Ti_reconstructed - mean(Ti_reconstructed)).^2);
R2 = 1 - SSres/SStot;
RMSE = sqrt(SSres / length(Ti_reconstructed));
```

X) Intervalo de Confiança

% 1. Estimar a matriz de covariância dos parâmetros (a partir do jacobiano)

```
cov_matrix = inv(J' * J) * SSres / (length(Ti_reconstructed) - length(initial_guess_temp));
```

% 2. Define a malha fina e calcula a curva ajustada

```
r_fine = linspace(0, a, 1000);
Ti_fine = Ti_function(best_params_temp, r_fine);
```

% 3. Cálculo da variância da predição usando o Jacobiano analítico

```
pred_var = zeros(size(r_fine));
for i = 1:length(r_fine)
    X = 1 - (r_fine(i)/a)^2;  % termo intermediário
```

```
    % Derivada em relação a T_i,max: (1 - (r/a)^2)^n
    dTmax = X^(best_params_temp(3));
    % Derivada em relação a T_min: 1 - (1 - (r/a)^2)^n
    dTmin = 1 - X^(best_params_temp(3));
    % Derivada em relação a n: (T_max - T_min) * (1 - (r/a)^2)^n * log(1 - (r/a)^2)
    if X <= 0
        dn = 0;  % Evita log(0)
    else
        dn = (best_params_temp(1) - best_params_temp(2)) * X^(best_params_temp(3)) *
log(X);
    end
    J_val = [dTmax, dTmin, dn];
    pred_var(i) = J_val * cov_matrix * J_val';
end

% Calcula o intervalo de confiança 95%
conf_range = 1.96 * sqrt(pred_var + RMSE^2 + mean(Ti_data_err)^2);
upper_conf = Ti_fine + conf_range;
lower_conf = Ti_fine - conf_range;
```

VI. Gerar Gráfico Comparativo

Matlab

```
figure;
hold on;

% Plot do intervalo de confiança
```

```
h_patch = fill([r_fine, fliplr(r_fine)], [upper_conf, fliplr(lower_conf)], 'b', ...
    'FaceAlpha', 0.15, 'EdgeColor', 'none');
uistack(h_patch, 'bottom');

% Plot dos dados experimentais com barras de erro
errorbar(r_data, Ti_reconstructed, Ti_data_err, 'ko', ...
    'LineWidth', 1.5, 'MarkerFaceColor', 'w', 'CapSize', 6, 'MarkerSize', 8);

% Plot da curva ajustada
plot(r_fine, Ti_fine, 'b-', 'LineWidth', 2);

% Configurações do gráfico
xlabel('Raio (r) [m]', 'FontSize', 12);
ylabel('Temperatura Iônica (T_i) [eV]', 'FontSize', 12);
title('Perfil de Temperatura com Intervalo de Confiança', 'FontSize', 14);
grid on;
xlim([0 a]);
set(gca, 'FontSize', 12);

% Legenda com R² incluído no item da curva ajustada
legend('Intervalo de 95%', 'Dados Experimentais', sprintf('Ajuste (R^2 = %.4f)', R2), ...
    'Location', 'northeast');

% Resultados Numéricos
fprintf('\n=== Resultados ===\n');
fprintf('T_i,max = %.2f eV\n', best_params_temp(1));
```

```
fprintf('T_i,min = %.2f eV\n', best_params_temp(2));

fprintf('Expoente n = %.2f\n', best_params_temp(3));

fprintf('R² = %.4f\n', R2);

fprintf('RMSE = %.2f eV\n', RMSE);```
```

Saída Esperada:

- Gráfico contendo:

- Dados experimentais com barras de erro.

- Curva ajustada da função canônica.

- Intervalo de confiança de 95%.

- Legenda com R^2.

- Resultados numéricos:

- Parâmetros ajustados ($T_{i,max}$, $T_{i,min}$, n).

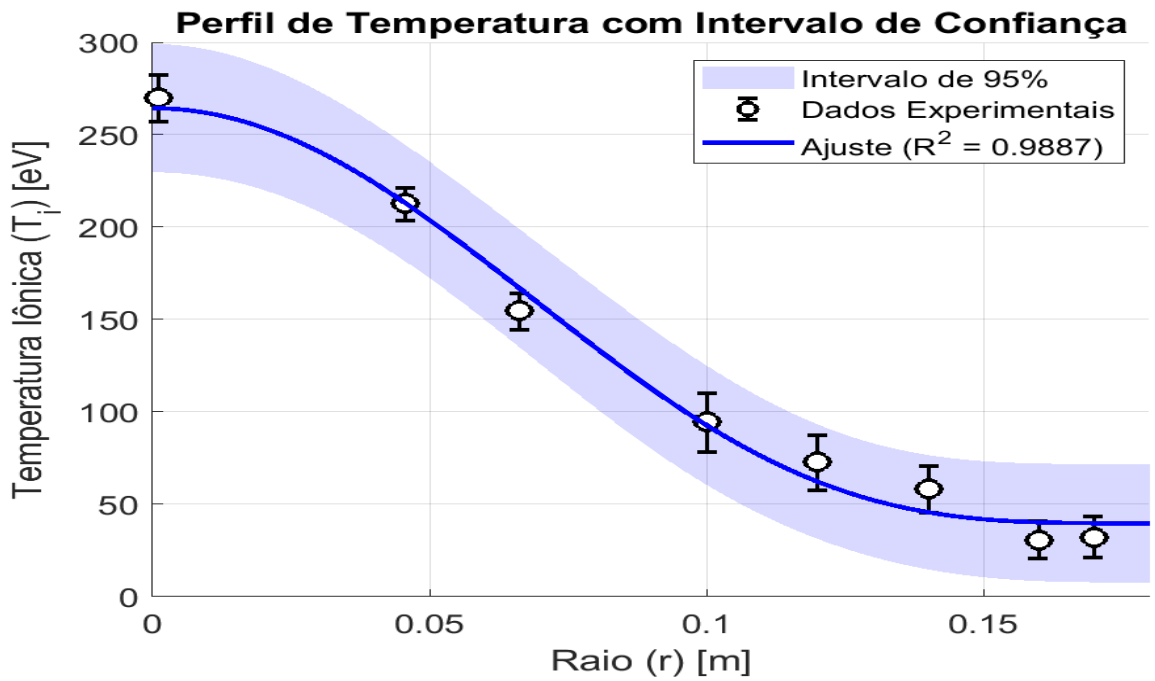
- R^2 e RMSE.

Instruções para o Relatório:

1. Inclua o gráfico do perfil de temperatura.

2. Apresente os coeficientes da função canônica em uma tabela:

Exemplo de Gráfico:



Parâmetro	Valor [eV]
$T_{i,max}$	
$T_{i,min}$	
n	

Observações:

- O intervalo de confiança considera incertezas experimentais e variabilidade do modelo.

- O uso de `lsqcurvefit` garante maior estabilidade numérica em comparação com `fminsearch`.

Parte 3: Gradiente de Temperatura e Velocidade de Helander

Objetivo: Calcular o gradiente de temperatura a partir do ajuste canônico, determinar o perfil de velocidade de Helander e compará-lo com os dados experimentais e o ajuste polinomial.

Passos:

I) Carregar os dados experimentais de velocidade e temperatura e crie um novo script chamado script_03: Calcular o gradiente de temperatura:

Matlab

```
r_Exp = [0; 0.01; 0.02; ... ]; % Raio experimental (m)
Vphi_Exp = [24.4725; 22.2933; ... ]; % Velocidade experimental (m/s)
Vphi_Exp_err = [0.9428; 1.8117; ... ]; % Incertezas experimentais (m/s)

r_data = [0.1021, 4.5631, ..., 17]/100; % Raio experimental (m)
Ti_reconstructed = [269.93, 212.63, ..., 32]; % Temperatura (eV)
Ti_data_err = 1*[12.54, 8.897, ..., 11]; % Incertezas (eV)
```

II) **Calcule a Derivada analítica do perfil de temperatura ajustado**

$$\frac{dT_i}{dr} = n(T_{i,max} - T_{i,min})(1 - (r/a)^2)^{n-1}(-2r/a^2);$$

Obs_05: Utilize os parâmetros de temperatura ajustados no Script_02.

III) Calcular a velocidade de Helander:

Use a fórmula para calcular a velocidade de Helander: $V_\phi(r) = \frac{2\epsilon}{B_{pol}}\frac{dT_i}{dr}$ onde $B_{pol} = -\left\{\mu_0\left[\frac{I_P}{2\pi}(r/a^2)\left[2 - (r/a)^2\right] + 0.001\right]\right\}$ com $\mu_0 = 4.\pi.10^{-7}$ sendo a constante magnética; $\epsilon = a/R_{centr} = 0{,}18/0{,}61 = 0{,}295$, $I_P = 90\ kA$ a corrente de plasma; e $a = 0{,}18\ m$ raio menor da coluna de plasma. Para a variável r **use: r = linspace(1e-3, a, 1000)';** % Malha radial

IV) Ajuste polinomial para comparação:

Obs_06: Utilize o polinômio cujo grau gerou o melhor R^2.

Matlab

```
p5 = polyfit(r_Exp, Vphi_Exp, 5);
poly5_val = polyval(p5, r);
```

V) **Calcular intervalo de confiança do polinômio:**

Matlab

```
SSres_poly = sum((Vphi_Exp - polyval(p5, r_Exp)).^2);
RMSE_poly = sqrt(SSres_poly / (length(r_Exp) - 6));
conf_range = 1.96 * RMSE_poly;
upper_conf = poly5_val + conf_range;
lower_conf = poly5_val - conf_range;
```

VI) **Calcular métricas de qualidade R^2 para a velocidade de Helander:**

Matlab

```
V_helander_Exp = interp1(r, V_helander, r_Exp, 'spline');
SSres_helander = sum((Vphi_Exp - V_helander_Exp).^2);
SStot_helander = sum((Vphi_Exp - mean(Vphi_Exp)).^2);
R2_helander = 1 - SSres_helander / SStot_helander;
```

VII) **Razão de aderência ao intervalo do polinômio (Quantifica o número de pontos dentro do intervalo de confiança):**

Matlab

```
dentro_intervalo = (V_helander >= lower_conf) & (V_helander <= upper_conf);
razao_aderencia = sum(dentro_intervalo) / length(r)
```

VIII) **Gerar um gráfico comparativo:**

Matlab

```
figure;
hold on;

% Intervalo de confiança
fill([r; flipud(r)], [upper_conf; flipud(lower_conf)], 'r', ...
    'FaceAlpha', 0.1, 'EdgeColor', 'none', ...
    'DisplayName', 'Intervalo de Confiança (Polinômio 5º)');

% Dados experimentais
errorbar(r_Exp, Vphi_Exp, Vphi_Exp_err, 'ko', ...
    'LineWidth', 1.5, 'MarkerFaceColor', 'w', 'CapSize', 6, ...
    'DisplayName', 'Dados Experimentais');
```

% Velocidade de Helander

```
plot(r, V_helander, 'b-', 'LineWidth', 2, ...
        'DisplayName', sprintf('Helander (R²=%.3f | %.1f%% no IC)', R2_helander, razao_aderencia*100));
```

% Ajuste polinomial

```
plot(r, poly5_val, 'm--', 'LineWidth', 2, ...
    'DisplayName', 'Ajuste Polinomial (5º Grau)');
```

% Configurações do gráfico

```
xlabel('Raio (r) [m]', 'FontSize', 12);
ylabel('Velocidade [m/s]', 'FontSize', 12);
title('Velocidade de Helander vs Experimental', 'FontSize', 14);
grid on;
legend('show', 'Location', 'best'); % Mostra a legenda automaticamente
set(gca, 'FontSize', 12);
```

Saída Esperada:

- Gráfico contendo:

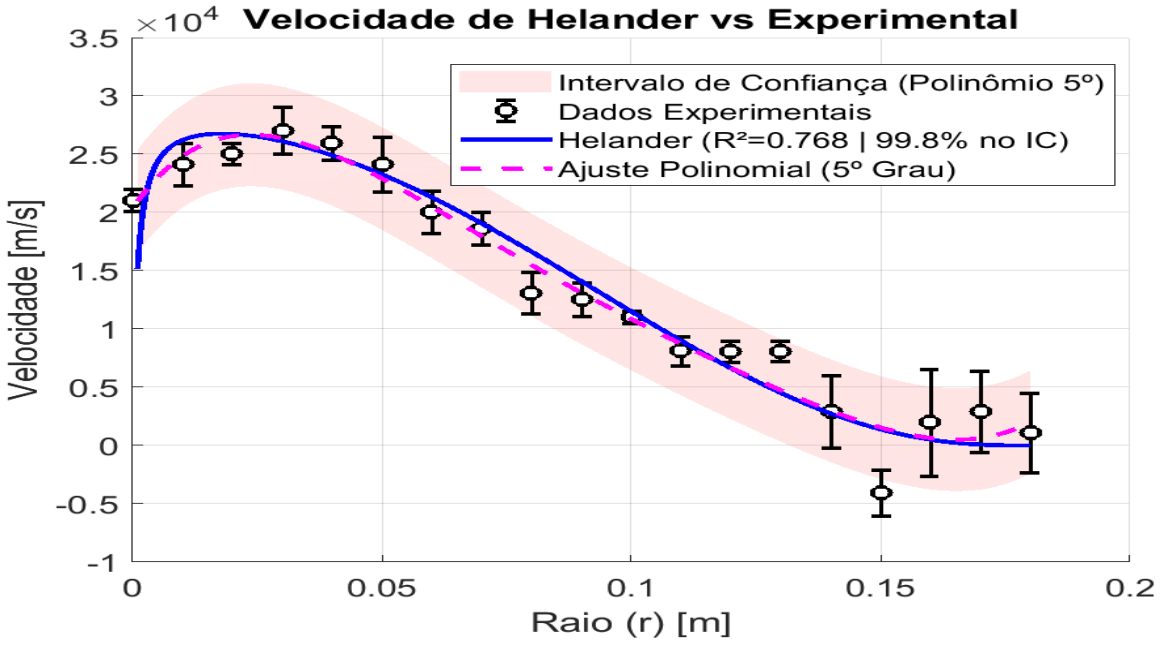
- Velocidade de Helander (linha azul).

- Dados experimentais com barras de erro.

- Intervalo de confiança do ajuste polinomial (área vermelha).

- Legenda com R^2 e porcentagem de pontos da Helander dentro do intervalo.

Exemplo de Gráfico:



Instruções para o Relatório:

1. Inclua o gráfico comparativo da velocidade.

2. Discuta:

- A discrepância entre o modelo de Helander e os dados experimentais.

- A significância da porcentagem de pontos dentro do intervalo.

3. Apresente os resultados numéricos em uma tabela:

Métrica	**Valor**
R^2 **(Helander)**	
Quantidade de pontos no Intervalo de Confiança (%)	

4. Anexe o script MATLAB utilizado.

Observações:

- O intervalo de confiança é calculado para o ajuste polinomial, servindo como referência para avaliar a aderência do modelo físico.

- A velocidade de Helander é sensível ao gradiente de temperatura. Pequenas variações nos parâmetros do ajuste canônico podem alterar significativamente o perfil.

Parte 4: Perturbação dos Dados de Temperatura e Análise de Sensibilidade

Objetivo: Perturbar os dados experimentais de temperatura dentro das incertezas, reajustar o modelo canônico, calcular a velocidade de Helander otimizada e compará-la com o ajuste polinomial experimental.

Passos:

I) Carregar os dados experimentais de velocidade e temperatura e crie um novo script chamado script_04:

Matlab

```
r_Exp = [0; 0.01; 0.02; ... ]; % Raio experimental (m)

Vphi_Exp = [24.4725; 22.2933; ... ]; % Velocidade experimental (m/s)

Vphi_Exp_err = [0.9428; 1.8117; ... ]; % Incertezas experimentais (m/s)


r_data = [0.1021, 4.5631, ..., 17]/100; % Raio experimental (m)

Ti_reconstructed = [269.93, 212.63, ..., 32]; % Temperatura (eV)

Ti_data_err = 1*[12.54, 8.897, ..., 11]; % Incertezas (eV)
```

II) Crie uma tabela com os principais dados que serão usados nos cálculos posteriores como:

```
a = 0.18;              % Raio menor do plasma [m]

Rcentr = 0.61;         % Raio maior central [m]

epsilon = a / Rcentr;  % Razão de aspecto inversa
```

```
mu0 = 4 * pi * 1e-7;     % Constante magnética
I_P = 90e3;              % Corrente de plasma [A]
r = linspace(1e-3, a, 1000)';  % Malha radial
```

III) **Definir a função canônica:**

Matlab

```
Ti_function = @(params, rr) (params(1) - params(2)) .* (1 - (rr/a).^2).^params(3) + params(2);
```

IV) Ajustar os parâmetros do perfil de temperatura, sem perturbação usando a função `lsqcurvefit`:

Matlab

```
initial_guess = [270, 30, 3.5];
[params_initial, ~, residual_initial, ~, ~, ~, J_initial] = lsqcurvefit(...
    @(p, rr) Ti_function(p, rr), initial_guess, r_Ti, Ti_reconstructed, [], [], ...
    optimset('Display', 'off'));
```

V) Perturbar dados e otimizar via Monte Carlo:

Matlab

```
n_iterations = 1000;
best_diff = Inf;
best_params_temp = [];

for i = 1:n_iterations
    Ti_perturbed = Ti_reconstructed + randn(size(Ti_reconstructed)) .* Ti_data_err;
     params = lsqcurvefit(@(p, rr) Ti_function(p, rr), initial_guess, r_Ti, Ti_perturbed, [], [], ...
       optimset('Display', 'off'));
```

```
    % Cálculo da velocidade de Helander
    Bpol = -(mu0 * (I_P/(2*pi)) * (r./a.^2) .* (2 - (r/a).^2) + 0.001);
     dTidr = (params(1) - params(2)) * params(3) .* (1 - (r/a).^2).^(params(3)-1) .* (-2*r/
a^2);
    V_helander = (2 * epsilon) ./ Bpol .* dTidr;

    % Ajuste polinomial de 5º grau
    poly5_val = polyval(polyfit(r_Exp, Vphi_Exp, 5), r);
    diff = sum((V_helander - poly5_val).^2);

    if diff < best_diff
        best_diff = diff;
        best_params_temp = params;
    end
end
```

VI) Cálculos finais sobre a velocidade

```
Bpol = -(mu0 * (I_P/(2*pi)) * (r./a.^2) .* (2 - (r/a).^2) + 0.001);
dTidr = (best_params_temp(1) - best_params_temp(2)) * best_params_temp(3) .* ...
        (1 - (r/a).^2).^(best_params_temp(3)-1) .* (-2*r/a^2);
V_helander_optimized = (2 * epsilon) ./ Bpol .* dTidr;

p5 = polyfit(r_Exp, Vphi_Exp, 5);
poly5_val = polyval(p5, r);

SSres_poly = sum((Vphi_Exp - polyval(p5, r_Exp)).^2);
RMSE_poly = sqrt(SSres_poly / (length(r_Exp) - 6));
```

```
conf_range_poly = 1.96 * RMSE_poly;

upper_conf_poly = poly5_val + conf_range_poly;

lower_conf_poly = poly5_val - conf_range_poly;


V_helander_Exp = interp1(r, V_helander_optimized, r_Exp, 'spline');

SSres_helander = sum((Vphi_Exp - V_helander_Exp).^2);

SStot_helander = sum((Vphi_Exp - mean(Vphi_Exp)).^2);

R2_helander = 1 - SSres_helander/SStot_helander;
```

VII) **Da mesma forma que você fez no item 3., calcule a Razão de aderência ao intervalo do polinômio (Quantifica o número de pontos dentro do intervalo de confiança):**

VIII) **Ajuste final e cálculo do intervalo de confiança da temperatura**

```
[~, ~, residual, ~, ~, ~, J] = lsqcurvefit(...

    @(p, rr) Ti_function(p, rr), best_params_temp, r_Ti, Ti_reconstructed, [], [], ...

    optimset('Display', 'off'));

cov_matrix = inv(J' * J) * (sum(residual.^2)/(length(Ti_reconstructed) - length(best_params_temp)));

r_fine = linspace(0, a, 1000);

Ti_fine_initial = Ti_function(params_initial, r_fine);

Ti_fine_optimized = Ti_function(best_params_temp, r_fine);


% Cálculo da variância em cada ponto de r_fine

pred_var = zeros(size(r_fine));

for i = 1:length(r_fine)

    X = 1 - (r_fine(i)/a)^2;

    % EVITA log(0) - clamp

    if X <= 1e-16

        X_for_log = 1e-16;
```

```
    else
        X_for_log = X;
    end
    dTmax = X^best_params_temp(3);
    dTmin = 1 - X^best_params_temp(3);
        dn = (best_params_temp(1) - best_params_temp(2)) * X^best_params_temp(3) *
log(X_for_log);
    J_val = [dTmax, dTmin, dn];
    pred_var(i) = J_val * cov_matrix * J_val';
end
RMSE_temp = sqrt(mean(residual.^2));
conf_range = 1.96 * sqrt(pred_var + RMSE_temp^2 + mean(Ti_data_err)^2);
upper_conf = Ti_fine_optimized + conf_range;
lower_conf = Ti_fine_optimized - conf_range;
```

6. Gerar gráfico de temperatura:

Matlab

```
figure;
hold on;
% 1. Intervalo de confiança (área preenchida)
fill([r_fine, fliplr(r_fine)], ...
     [upper_conf, fliplr(lower_conf)], ...
     [0.2 0.6 1], 'FaceAlpha', 0.2, 'EdgeColor', 'none', ...
     'DisplayName', 'Intervalo de 95%');
% 2. Dados experimentais
errorbar(r_Ti, Ti_reconstructed, Ti_data_err, 'ko', ...
    'LineWidth', 1.5, 'MarkerFaceColor', 'w', 'CapSize', 6, 'MarkerSize', 8, ...
```

```
    'DisplayName', 'Dados Experimentais');
% 3. Ajuste Inicial
R2_initial = calc_R2(Ti_function(params_initial, r_Ti), Ti_reconstructed);
Ti_initial_label = sprintf('Ajuste Inicial (R²=%.3f)', R2_initial);
plot(r_fine, Ti_fine_initial, 'b--', 'LineWidth', 2, 'DisplayName', Ti_initial_label);
% 4. Ajuste Otimizado
R2_optimized = calc_R2(Ti_function(best_params_temp, r_Ti), Ti_reconstructed);
% Razão da função otimizada no IC da temperatura
razao_otimizada_no_IC = sum((Ti_fine_optimized >= lower_conf) & ...
                        (Ti_fine_optimized <= upper_conf)) / length(r_fine);

%: Incluímos a fração do ajuste otimizado dentro do IC
Ti_optimized_label = sprintf('Ajuste Otimizado (R²=%.3f | %.1f%% no IC)', ...
    R2_optimized, 100*razao_otimizada_no_IC);

plot(r_fine, Ti_fine_optimized, 'r-', 'LineWidth', 2, ...
    'DisplayName', Ti_optimized_label);
xlabel('Raio (r) [m]', 'FontSize', 12);
ylabel('Temperatura Iônica [eV]', 'FontSize', 12);
title('Perfil de Temperatura com Intervalo de Confiança (95%)', 'FontSize', 14);
grid on;
xlim([0 a]);
set(gca, 'FontSize', 12);
legend('Location', 'northeast');
```

- **Velocidade:**

Matlab

```
figure;
hold on;

% Intervalo de confiança do polinômio
fill([r; flipud(r)], [upper_conf_poly; flipud(lower_conf_poly)], ...
     'r', 'FaceAlpha', 0.1, 'EdgeColor', 'none', ...
     'DisplayName', 'Intervalo de Confiança (95%)');

% Dados experimentais
errorbar(r_Exp, Vphi_Exp, Vphi_Exp_err, 'ko', ...
    'LineWidth', 1.5, 'MarkerFaceColor', 'w', 'CapSize', 6, ...
    'DisplayName', 'Dados Experimentais');
R2_poly = 1 - SSres_poly / SStot_helander;
plot(r, poly5_val, 'm--', 'LineWidth', 2, ...
    'DisplayName', sprintf('Ajuste Polinomial (R²=%.3f)', R2_poly));

%: Cálculo a fração de Helander no IC
vel_hel_label = sprintf('Velocidade de Helander (R²=%.3f | %.1f%% no IC)', ...
    R2_helander, 100*razao_aderencia);
plot(r, V_helander_optimized, 'b-', 'LineWidth', 2, ...
    'DisplayName', vel_hel_label);
xlabel('Raio (r) [m]', 'FontSize', 12);
ylabel('Velocidade [m/s]', 'FontSize', 12);
title('Perfil de Velocidade: Experimental vs Modelos', 'FontSize', 14);
grid on;
xlim([0 a]);
set(gca, 'FontSize', 12);
```

```
legend('Location', 'northeast');
```

% Impressão dos resultados numéricos

```
fprintf('=== Resultados ===\n');
fprintf('Temperatura:\n');
fprintf('  R² Inicial:   %.3f\n', R2_initial);
fprintf('  R² Otimizado: %.3f\n', R2_optimized);
fprintf('  Razão Otimizada no IC: %.1f%%\n', 100*razao_otimizada_no_IC);
fprintf('\nVelocidade:\n');
fprintf('  R² Polinômio: %.3f\n', R2_poly);
fprintf('  R² Helander:  %.3f\n', R2_helander);
fprintf('  Helander no IC: %.1f%%\n', 100*razao_aderencia);
```

% Função auxiliar para R²

```
function r2 = calc_R2(model_values, data_values)
    SSres = sum((data_values - model_values).^2);
    SStot = sum((data_values - mean(data_values)).^2);
    r2 = 1 - SSres / SStot;
end
```

Saída Esperada:

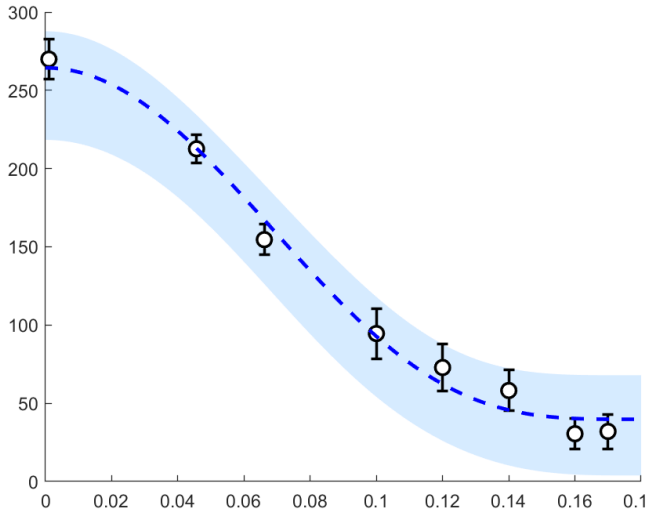
- Gráficos:

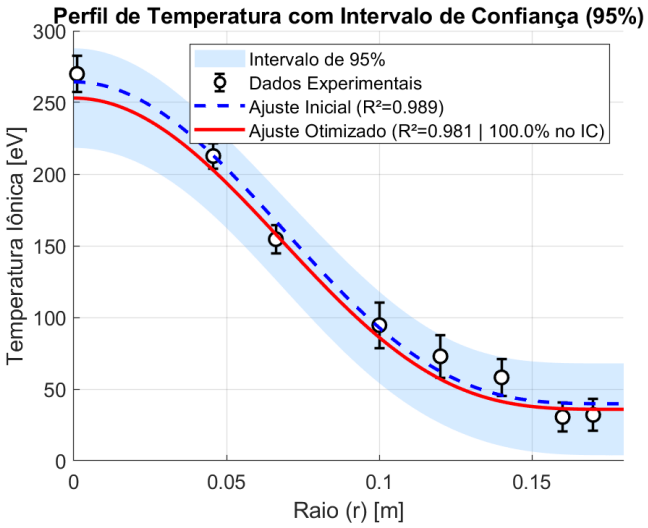
- Temperatura: Dados experimentais, ajuste inicial, ajuste otimizado e intervalo de confiança.

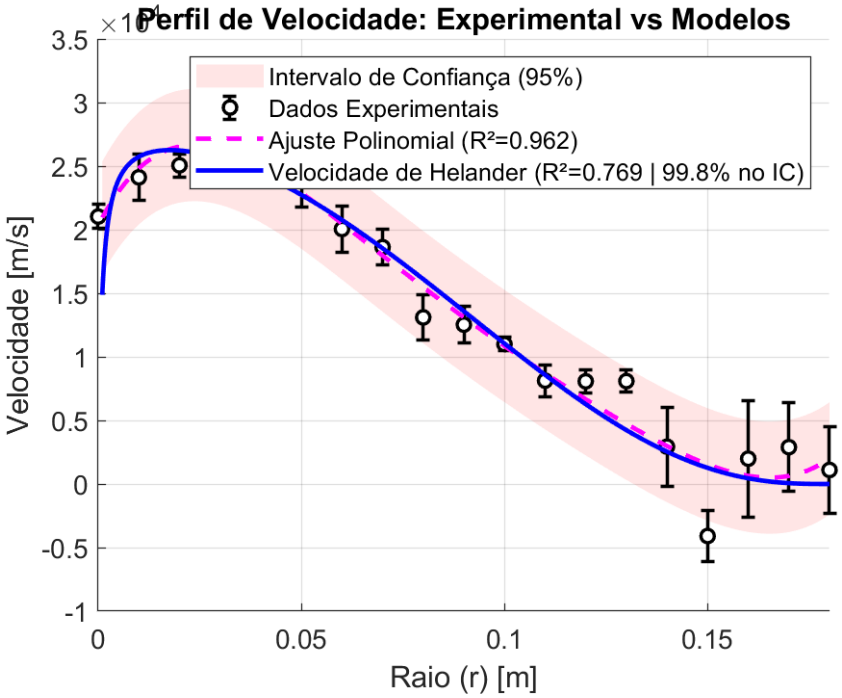
- Velocidade: Dados experimentais, ajuste polinomial, velocidade de Helander e intervalo de confiança.

Exemplo de Gráfico:

Modelo	R^2	% no IC
Canônica antes da Pertubação.	0,981	100
Canônica após a Pertubação	0,989	100
Ajuste Polinomial antes da Perturbação	0,779	99,8%
Ajuste Polinomial depois da Perturbação	0,962	100%







Instruções para o Relatório:

1. Inclua os gráficos de temperatura e velocidade.

2. Documente os parâmetros ajustados:

Parâmetro	Valor (Antes da Perturbação)	Valor (Após Perturbação)
$T_{i,max}$	264.37 eV	266.82 eV
$T_{i,min}$	39.77 eV	4.89 eV
n	3.91	2.94

3. Discuta:

- Como a perturbação afetou os parâmetros do modelo canônico.

- A compatibilidade da velocidade de Helander com o intervalo de confiança experimental.

4. Anexe o script MATLAB ao relatório.

Observações:

- O intervalo de confiança considera incertezas experimentais e variância do modelo.

- A velocidade de Helander é sensível ao gradiente de temperatura: pequenas perturbações podem alterar significativamente o perfil.