Função Matlab lsqcurvefit:

1. initial_guess = [270, 30, 3.5];
- Propósito: Define valores iniciais para os parâmetros do modelo que será ajustado aos dados experimentais.
- Parâmetros:
- 270: Estimativa inicial para $T_{i,\max}$ (temperatura máxima no centro da coluna de plasma).
- 30: Estimativa inicial para $T_{i,\min}$ (temperatura mínima na borda da coluna de plasma).
- 3.5: Estimativa inicial para n (expoente que define o formato do perfil de temperatura).

2. lsqcurvefit (Ajuste Não Linear por Mínimos Quadrados)
A função `lsqcurvefit` ajusta um modelo não linear a dados experimentais minimizando a soma dos quadrados dos resíduos. Sua sintaxe é:

[parâmetros_ajustados, ~, resíduos, ~, ~, ~, Jacobiano] = lsqcurvefit(função_modelo, chute_inicial, dados_x, dados_y, limites_inferiores, limites_superiores, opções);

No código:
- @(p, rr) Ti_function(p, rr):
- Propósito: Define a função modelo $T_i(r)$ que será ajustada.
- p: Vetor de parâmetros a serem ajustados ($T_{i,\max}$, $T_{i,\min}$, n).
- rr: Vetor de pontos do raio r.
- Exemplo de modelo:

$$T_i(r) = (T_{i,\max} - T_{i,\min}) \cdot \left[1 - \left(\frac{r}{a}\right)^2\right]^n + T_{i,\min}$$

- **initial_guess:** Valores iniciais dos parâmetros ($T_{i,\max} = 270$, $T_{i,\min} = 30$, $n = 3{,}5$).

- r_Ti: Dados experimentais do raio r (pontos onde a temperatura foi medida).
- Ti_reconstructed: Dados experimentais da temperatura iônica $T_i(r)$.

- []: Indica que não há limites inferiores ou superiores para os parâmetros.
- optimset('Display', 'off'): Configura o algoritmo para não exibir mensagens de progresso (`'off'`).

3. Saídas da Função
- params_initial: Parâmetros ajustados ($T_{i,\max}$, $T_{i,\min}$, n) que melhor se encaixam nos dados.
- ~: Ignora saídas não utilizadas (ex: norma do resíduo).

- residual_initial: Vetor de resíduos (diferenças entre dados experimentais e modelo ajustado).
- J_initial: Matriz Jacobiana, que contém as derivadas parciais do modelo em relação aos parâmetros.
- Exemplo:

$$J = \begin{bmatrix} \frac{\partial T_i(r_1)}{\partial T_{i,\max}} & \frac{\partial T_i(r_1)}{\partial T_{i,\min}} & \frac{\partial T_i(r_1)}{\partial n} \\ \vdots & \vdots & \vdots \\ \frac{\partial T_i(r_m)}{\partial T_{i,\max}} & \frac{\partial T_i(r_m)}{\partial T_{i,\min}} & \frac{\partial T_i(r_m)}{\partial n} \end{bmatrix}$$

- Uso: Usada para calcular incertezas nos parâmetros (via matriz de covariância).

Resumo do Funcionamento:
1. Passo 1: O algoritmo inicia com os valores de `initial_guess`.
2. Passo 2: Calcula a diferença quadrática entre o modelo e os dados experimentais.
3. Passo 3: Ajusta iterativamente os parâmetros para minimizar essa diferença.
4. Passo 4: Retorna os parâmetros otimizados (`params_initial`), resíduos (`residual_initial`), e Jacobiano (`J_initial`).

Exemplo Prático:
Suponha que os dados experimentais sejam:
- $r = [0.1, 0.2, 0.3]$ m
- $T_i = [250, 200, 150]$ eV

O `lsqcurvefit` ajustará$T_{i,\max}$, $T_{i,\min}$, e n para que a curva do modelo passe o mais próximo possível dos pontos experimentais.

Importância do Jacobiano:
O Jacobiano permite calcular a **matriz de covariância** dos parâmetros:
$\text{Cov} = (J^T J)^{-1} \cdot \text{RMSE}^2$ Essa matriz quantifica a incerteza nos parâmetros ajustados. Por exemplo, se a variância de $T_{i,\max}$ for alta, significa que esse parâmetro é pouco restrito pelos dados.