Documentação do Código para Estimação de Parâmetros em um Pêndulo Amortecido Forçado

Autor: Thaisse Dias Paes

30 de março de 2025

1 Introdução

Este código tem como objetivo estimar os parâmetros de um pêndulo amortecido forçado utilizando duas abordagens de otimização:

- **Algoritmos Genéticos (AGs):** Implementados com a biblioteca DEAP, os quais exploram de forma estocástica o espaço de busca dos parâmetros.
- **Mínimos Quadrados Não Lineares (MQNL):** Implementados por meio da função `curve_fit` da biblioteca SciPy, que ajusta o modelo aos dados simulados.

O modelo físico é descrito por uma equação diferencial ordinária (EDO) que representa o movimento de um pêndulo submetido a amortecimento e forçamento externo. Os parâmetros a serem estimados são:

- A massa m.
- O coeficiente de amortecimento c.
- A amplitude do forçamento F.

2 Estrutura do Código

O código está organizado nas seguintes seções:

2.1 Configuração Inicial

- Definição de constantes físicas: a aceleração da gravidade g e o comprimento do pêndulo l.
- Especificação dos parâmetros reais (m_{real}, c_{real}, F_{real}) que serão utilizados para gerar os dados de referência.
- Definição do intervalo de tempo para a simulação, com a discretização obtida por meio do vetor `t_data`.

2.2 Definição do Modelo Físico

A função `pendulo_forcado` define a EDO do pêndulo amortecido forçado:

```
def pendulo_forcado(t, y, m, c, F):
    theta, omega = y
    return [omega, -(g/l)*np.sin(theta) - (c/m)*omega + (F/m)*np.cos(2*t)]
```

- **Entradas:** Tempo t, vetor de estado $y = [\theta, \omega]$ e os parâmetros m, c e F.
- **Saídas:** Derivadas de θ e ω, de acordo com o modelo físico.

2.3 Função de Simulação

A função `simulate_pendulum` utiliza o solver `solve_ivp` do SciPy para simular o comportamento do pêndulo:

```
def simulate_pendulum(m, c, F, t_eval=t_data, y0=[0.1, 0], t_span=t_span):
    """
    Simula o p ndulo amortecido for ado e retorna o vetor de  ngulos.
    Para acelerar a simula  o , utilizam-se toler ncias relaxadas e n  o  se usa dense_output.
    """
    sol = solve_ivp(
        pendulo_forcado,
        t_span,
        y0,
        args=(m, c, F),
        t_eval=t_eval,
        method='RK45',
        rtol=1e-4,
        atol=1e-6
    )
    return sol.y[0]
```

Esta função retorna o vetor de ângulos ao longo do tempo.

2.4 Geração dos Dados "Reais"

Utilizando a função de simulação, os dados de referência são gerados com a adição de ruído (1%) para simular erros de medição:

```
theta_real = simulate_pendulum(m_real, c_real, F_real) + np.random.normal(0, 0.01, len(t_data))
```

2.5 Otimização via Algoritmo Genético

Nesta seção, define-se a estrutura dos indivíduos e da população utilizando a biblioteca DEAP:

- Criação das classes `FitnessMin` e `Individual`.
- Registro dos atributos individuais dentro de intervalos plausíveis para m, c e F.

- Definição dos operadores genéticos: cruzamento (`mate`), mutação (`mutate`) e seleção (`select`).

A função de fitness calcula o erro médio quadrático entre os dados simulados e os dados reais, aplicando penalizações para indivíduos com parâmetros fora dos limites esperados:

```
def fitness(individual):
    m, c, F = individual
    penalty = 0
    if not (1.0 <= m <= 2.0): penalty += 10 * abs(m - 1.5)
    if not (0.1 <= c <= 0.5): penalty += 10 * abs(c - 0.3)
    if not (0.5 <= F <= 1.2): penalty += 10 * abs(F - 0.8)

    try:
        theta_pred = simulate_pendulum(m, c, F)
        error = np.mean((theta_pred - theta_real) ** 2)
        return error + penalty,
    except Exception:
        return 1e6,
```

O algoritmo evolutivo é executado com uma população inicial de 150 indivíduos por 200 gerações, e o melhor indivíduo é selecionado.

2.6 Ajuste via Mínimos Quadrados

A função `modelo_para_ajuste` é definida para que o `curve_fit` possa ajustar o modelo aos dados:

```
def modelo_para_ajuste(t, m, c, F):
    return simulate_pendulum(m, c, F, t_eval=t)
```

Os parâmetros são ajustados a partir de palpites iniciais e limites pré-definidos.

2.7 Visualização dos Resultados

São calculadas as soluções dos métodos (AG e MQNL) e, em seguida, os resultados são plotados:

- A curva preta representa os dados reais.
- A curva vermelha (tracejada) corresponde à solução obtida pelo Algoritmo Genético.
- A curva azul (pontilhada) corresponde à solução dos Mínimos Quadrados.

Além do gráfico, os resultados são impressos em formato tabular no terminal.

3 Conclusão

Este código integra técnicas de simulação e otimização para a estimação de parâmetros de um pêndulo amortecido forçado. A combinação do uso de Algoritmos Genéticos e do método dos Mínimos Quadrados Não Lineares possibilita uma comparação robusta entre duas abordagens distintas. Enquanto os AGs oferecem maior robustez na presença de ruído e em sistemas não lineares complexos, o método dos Mínimos Quadrados proporciona uma solução computacionalmente mais eficiente em cenários com condições iniciais bem definidas.

Esta documentação serve como guia para a compreensão da estrutura e funcionamento do código, facilitando a manutenção, a replicação dos experimentos e a eventual extensão para a análise de outros sistemas dinâmicos.