SEVEN
publicações acadêmicas

Um código em Python para ajuste de dados periódicos de temperatura do solo

https://doi.org/10.56238/sevened2024.003-046

Geilson de Almeida Soares
Universidade Federal de Pelotas
E-mail: geilson.soares@ufpel.edu.br

Jairo Valões de Alencar Ramalho
Universidade Federal de Pelotas
E-mail: jairo.ramalho@ufpel.edu.br

Ana Maria Bersch Domingues
Universidade Federal de Pelotas
E-mail: ambdomingues@ufpel.edu.br

Honório Joaquim Fernando
Universidade Federal Fluminense
E-mail: honoriofernando@id.ufl.br

RESUMO
Este trabalho objetiva apresentar um código em linguagem Python para ajuste de curvas periódicas a partir de dados de simulação de temperatura do solo. Apresenta-se uma metodologia para obter uma função senoidal via método dos mínimos quadrados que representa a temperatura do solo de uma determinada localidade. O código usa também o método de Newton para resolver o sistema não linear resultante de equações. Foram empregados dados de simulação de temperatura do solo em uma cidade do Rio Grande do Sul. Os resultados destas curvas podem auxiliar no estudo e na definição de condições de contorno de novos modelos computacionais de trocadores de calor solo-ar (TCSA). Observa-se que o método dos mínimos quadrados não é uma novidade, mas há uma lacuna na literatura relativa à apresentação de códigos para ajuste de funções trigonométricas como é apresentado aqui.

Palavras-chave: Ajuste de Funções Trigonométricas, Mínimos Quadrados, Trocadores de Calor Solo-Ar, Python.

1 INTRODUÇÃO

Uma maneira de reduzir o consumo de energia em sistemas de ar condicionado é por meio do uso de trocadores de calor solo-ar (TCSA). Esses dispositivos são compostos por ventiladores e dutos enterrados, que insuflam o ar externo no interior dos dutos, permitindo trocas de calor entre o ar e o solo. Como resultado, o ar sai a uma temperatura mais branda. Isso ocorre porque o solo funciona como um reservatório térmico, armazenando calor no verão e liberando-o no inverno. Assim, a temperatura do solo é geralmente maior do que a do ar exterior no inverno, per- mitindo um aquecimento do ar circulante em tubos enterrados graças às trocas de calor entre o solo e o duto. No verão, o oposto ocorre, quando a temperatura do solo é menor que a do ar atmosférico. Dentre algumas referências sobre TCSA, pode-se citar (DOMINGUES *et al.*, 2021; RAMALHO *et al.*, 2022; VAZ *et al.*, 2011)

Para simular TCSA, é comum utilizar modelos para as temperaturas do ar e do solo de uma determinada região. Este trabalho avalia o ajuste por funções trigonométricas dos dados de simulação de temperaturas em uma cidade do estado do Rio Grande do Sul, onde o clima é subtropical e a variação anual de temperaturas segue um padrão periódico.

Há poucos trabalhos apresentando códigos para ajustes envolvendo funções periódicas, como é o caso de (BRUM *et al.*, 2011), onde os autores adotaram uma função real contínua $f(x) = Asen(Bx + C) + D$ (com A, B, C e D sendo constan- tes reais) para modelar, através do método dos mínimos quadrados, um conjunto discreto de pares ordenados $U = (x_i, y_i)$, $i = 1, 2, ..., n$, onde n é um número natu- ral. Por outro lado, em diversas situações, o período T do fenômeno é conhecido e, consequentemente, sabe-se também a frequência angular $B = 2\pi/T$. Por exemplo, a temperatura do solo no Rio Grande do Sul segue uma variação periódica anual, ou seja, podemos tomar T como sendo 365 dias. Assim, o objetivo deste trabalho é apresentar uma nova versão do código proposto em (BRUM *et al.*, 2011), para determinar apenas os coeficientes A, C e D.

2 METODOLOGIA

2.1 MÍNIMOS QUADRADOS DISCRETO

Pode-se usar funções trigonométricas do tipo $f(x) = Asen(Bx + C) + D$ para modelar a temperatura do solo. Sabendo-se o valor de $B = 2\pi/T$, as outras cons- tantes A, C e D podem ser determinadas pelo método dos mínimos quadrados, de modo a minimizar a função erro:

$$Erro(A,C,D) := \sum_{i=1}^{N} [f(x_i) - y_i]^2 = \sum_{i=1}^{N} \left[A \operatorname{sen}\left(\frac{2\pi}{T} x_i + C\right) + D - y_i \right]^2 \qquad (1)$$

Onde N é o número de dados.

Sob suposições como as consideradas em (BRUM *et al.*, 2011), pode-se afirmar que o valor mínimo da função em (1) é alcançado quando seu gradiente é nulo. Fazendo isso, somos conduzidos a

$$\begin{cases} \displaystyle\sum_{i=1}^{N}\left[A\,\mathrm{sen}\left(\frac{2\pi}{T}x_i+C\right)+D-y_i\right]\mathrm{sen}\left(\frac{2\pi}{T}x_i+C\right)=0\,, \\ \displaystyle\sum_{i=1}^{N}\left[A\,\mathrm{sen}\left(\frac{2\pi}{T}x_i+C\right)+D-y_i\right]A\cos\left(\frac{2\pi}{T}x_i+C\right)=0\,, \\ \displaystyle\sum_{i=1}^{N}\left[A\,\mathrm{sen}\left(\frac{2\pi}{T}x_i+C\right)+D-y_i\right]=0\,. \end{cases} \qquad (2)$$

Para resolver o sistema não linear em (2), foi adotado o método de Newton.

2.2 MÉTODO DE NEWTON

Dado um campo vetorial não linear e diferenciável

$$\vec{F}(A,C,D)=\begin{pmatrix} u(A,C,D) \\ v(A,C,D) \\ w(A,C,D) \end{pmatrix}, \qquad (3)$$

Pode-se aproximar $\vec{F}$, na vizinhança de um ponto $x_0 = (A_0,C_0,D_0)$, pelo campo vetorial linear:

$$\vec{L}(A,C,D)=\begin{pmatrix} u(x_0)+u_A(x_0)(A-A_0)+u_C(x_0)(C-C_0)+u_D(x_0)(D-D_0) \\ v(x_0)+v_A(x_0)(A-A_0)+v_C(x_0)(C-C_0)+v_D(x_0)(D-D_0) \\ w(x_0)+w_A(x_0)(A-A_0)+w_C(x_0)(C-C_0)+w_D(x_0)(D-D_0) \end{pmatrix}. \qquad (4)$$

Logo, ao invés de buscar solução para $\vec{F}(A,C,D)=0$, resolve-se o problema aproximado $\vec{L}(A,C,D)=0$ que possui a representação matricial

$$J(x_0)x = J(x_0)x_0 - \vec{F}(x_0)\,, \qquad (5)$$

Onde

$$J(s) = \begin{bmatrix} u_A(s) & u_C(s) & u_D(s) \\ v_A(s) & v_C(s) & v_D(s) \\ w_A(s) & w_C(s) & w_D(s) \end{bmatrix}, \quad x = \begin{pmatrix} A \\ C \\ D \end{pmatrix}, \quad \text{e} \quad x_0 = \begin{pmatrix} A_0 \\ C_0 \\ D_0 \end{pmatrix}, \qquad (6)$$

Com $J(s)$ denotando a matriz jacobiana do campo vetorial em (3) avaliada no ponto $s \in \mathrm{R}^3$. Portanto, cada entrada $z_p(s)$, $z \in \{u,v,w\}$ e $p \in \{A,C,D\}$ de $J(s)$, indica a derivada parcial de z com respeito a p.

Assim, quando se deseja achar um valor aproximado da raiz x_r de um campo vetorial, isto pode ser feito iterativamente escolhendo uma aproximação inicial x_0 e calculando aproximações sucessivas através da resolução do sistema de equações lineares

$$J(x_k)x_{k+1} = J(x_k)x_k - \vec{F}(x_k)\,, \qquad (7)$$

Com $k = 0, 1, 2,...$, etc. Esta técnica de resolução é chamada de Método de Newton (BURDEN; FAIRES; BURDEN, 2015).

Visando a redução do custo computacional associado a implementação direta de (7), introduz-se a variável auxiliar $t_{k+1} := x_{k+1} - x_k$ que transforma o problema na seguinte sequência:

1. Determinar t_{k+1} em $J(x_k)t_{k+1} = -\vec{F}(x_k)$,
2. Calcular $x_{k+1} = x_k + t_{k+1}$,

Evitando-se com isso o cálculo do primeiro termo no lado direito de (7) que pode demandar um custo computacional significativo em todo processo de busca pela raíz x_r de $\vec{F}$.

O método de Newton pode convergir relativamente rápido ou divergir depen- dendo de certas condições, como a aproximação inicial escolhida e os valores das derivadas parciais na matriz Jacobiana. Há uma discussão sobre esse assunto dis- ponível nos trabalhos (BURDEN; FAIRES; BURDEN, 2015; BRUM *et al.*, 2011).

3 CÓDIGO UTILIZADO COMENTADO

Com a finalidade de mostrar a forma como o sistema não linear é resolvido numericamente, apresenta-se a seguir o código desenvolvido, que foi implementado na linguagem de programação Python (versão 3.11.2). No Python, textos à direita do símbolo #, assim como os textos limitados por 3 aspas (' ' '), são comentários. NumPy é uma biblioteca da linguagem Python, que tem capacidade para manipular arranjos (matrizes multidimensionais) de grandes dimensões, além de oferecer uma ampla variedade de funções matemáticas avançadas para trabalhar com esses arranjos. As funções utilizadas no código são descritas a seguir:

- array() : define uma matriz;
- max() : localiza o valor máximo em uma matriz;
- mean() : calcula a média aritmética;
- argmax() : retorna o posição do valor máximo;
- linalg.solve() : retorna a solução de um sistema de equações lineares;
- abs(): retorna o valor absoluto;
- sum() : retorna o somatório.

Portanto, a linguagem Python disponibiliza uma ampla gama de recursos pron- tos que facilitam a implementação dos mais variados algorítmos computacionais. Explanações detalhadas sobre a linguagem de programação Python podem ser en- contradas em (LANGTANGEN, 2011). A listagem a seguir, apresenta integralmente o código desenvolvido neste trabalho.

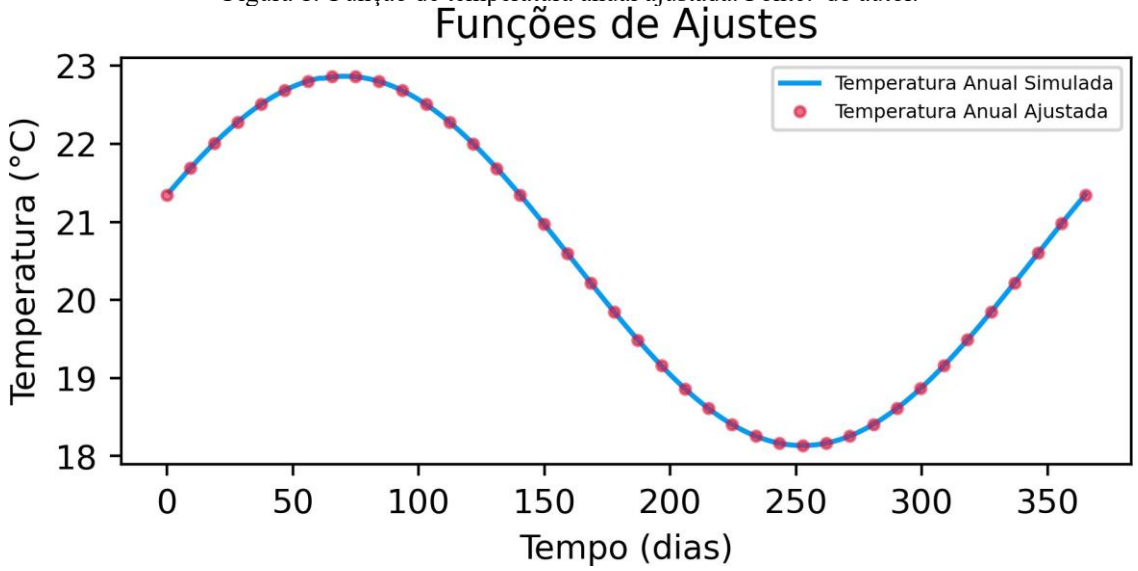
```
import numpy as np

'''
Esta função calcula os coeficientes a,b,c,d para uma função do tipo
a*sin(bx+c)+d que ajusta a solução da temperatura anual.
'''
def ajustes(lista_dias, dados):
    # Converter lista_dias para um array numérico
    x = np.array(lista_dias).astype(float)
    # Atribuindo os dados ao y
    y = dados

    # Estimativas iniciais para os valores dos coeficientes da função seno
    y_max = np.max(y)
    dia_max = np.argmax(y)
    d = np.mean(y)  # Média aritmética dos elementos da matriz
    a = y_max - d
    b = 2*np.pi/365
    c = np.pi/2 - 2*np.pi*dia_max/365

    # Tolerância e número máximo de iterações
    tol = 1e-5
    nMax = 200

    # Intervalos de tempo (x) e Valores de temperatura (y)
    # Número de intervalos de tempo
```

```
    n = max(x)

    cont = 1  # Contador para o número de iterações

    while cont < nMax:

        # Computação da Matriz Jacobiana
        jac = np.array([
            [np.sum(np.sin(b*x+c)**2), np.sum(np.cos(b*x+c)*(2*a*np.sin(b*x+c)+
    d-y)), np.sum(np.sin(b*x+c))],
            [np.sum(np.sin(b*x+c)*np.cos(b*x+c)), np.sum((a*(np.cos(b*x+c))
    **2-(a*np.sin(b*x+c)+d-y)*np.sin(b*x+c))), np.sum(np.cos(b*x+c))],
            [np.sum(np.sin(b*x+c)), np.sum(a*np.cos(b*x+c)), n]
        ])

        # Computação do Campo Vetorial F
        F = np.array([
            [np.sum((a*np.sin(b*x+c)+d-y)*np.sin(b*x+c))],
            [np.sum((a*np.sin(b*x+c)+d-y)*np.cos(b*x+c))],
            [np.sum(a*np.sin(b*x+c)+d-y)]
        ])

        # Resolvendo o sistema linear jac t =-F
        t = np.linalg.solve(jac, -F)

        # Verificar a convergência na norma do infinito
        if np.max(np.abs(t)) < tol: break
        else:
            cont += 1

            # Atualiza a solução
            a += t[0]
            c += t[1]
            d += t[2]

    # Retorno dos coeficiêntes
    return a[0], c[0], d[0]
```

4 RESULTADOS

Neste estudo, foram empregados dados simulados das pesquisas realizadas no trabalho (RAMALHO *et al.*, 2022). Os dados referem-se as temperaturas médias diárias do solo na cidade de Viamão/RS, ao longo de um ano. Assim, procurou-se determinar a melhor função senoidal do tipo $f(x) = Asen((2\pi/T)x + C) + D$ que se adequasse aos dados de todos os 365 dias do ano. Para isso, utilizou-se a metodologia descrita na seção anterior. A Figura 1 apresenta um gráfico que ilustra tanto os dados simulados como também a curva de ajuste da temperatura simulada ao longo de um ano. Como pode ser observado, a função senoidal é uma representação adequada e natural da variação da temperatura ao longo das estações, alcançando seu extremo máximo no verão e mínimo no inverno, além de acompanhar a queda e aumento gradual das temperaturas durante o outono e a primavera, respectivamente. De fato, como evidenciado na mesma figura, a curva ajustada apresentou uma alta proximidade com os dados simulados.

Figura 1. Função de temperatura anual ajustada. Fonte: do autor.



5 CONCLUSÕES

O objetivo deste estudo foi desenvolver um código computacional na linguagem de programação Python para ajustes de curvas periódicas. Inspirado em um trabalho similar feito na linguagem de programação MatLab, foi desenvolvido uma versão simplificada para o caso em que o período do fenômeno é conhecido com a finalidade de reduzir o custo computacional. Em particular, utilizaram-se dados simulados de temperatura do solo de uma cidade do estado do Rio Grande do Sul.

Foi observado que o ajuste do código em Python apresentou uma aproximação bastante satisfatória em relação aos dados simulados. A análise da qualidade do ajuste será feita em trabalhos futuros. Além disso, este estudo contribuiu para preencher uma lacuna na literatura em relação a implementação computacional do método de mínimos quadrados para ajuste de funções trigonométricas.

AGRADECIMENTOS

Geilson de Almeida Soares agradece o CNPq pela bolsa de iniciação científica. De igual modo, Ana Maria Bersch Domingues agradece o suporte financeiro da Capes.

REFERÊNCIAS

Brum, R. S., Ramalho, J. V. A., Rocha, L. A. O., Isoldi, L. A., Santos, E. D., "A Matlab code to fit periodic data". Em: Revista Brasileira de Computação Aplicada 7 (2015), pp. 16–25.

Burden, R. L., Faires, J. D., Burden, A. M. Análise Numérica, 10a. ed. São Paulo: Cengage Learning, 2015. ISBN: 978-85-2212-341-4.

Domingues, A.M.B., Nóbrega, E.S.B., Ramalho, J.V.A., Brum, R.S., Quadros, R.S., "Parameter analysis of Earth-air heat exchangers over multi-layered soils in South Brazil". Em: Journal Geothermics 93 (2021), pp. 1–14.

Ramalho, J. V. A., Fernando, H. J., Brum, R. S., Domingues, A. M. B., Pastor, N. R. N., Olivera, M. R. B., "Accessing the thermal performance of Earth-air heat exchangers surrounded by galvanized structures". Em: Sustainable Energy Technologies and Assessments 54 (2022), pp. 1–11.

Vaz, J., Sattler, M.A., Santos, E.D., Isoldi, L.A., "Experimental and numerical analysis of an earth-air heat exchancher". Em: Journal Energy and Buildings 43 (2011), pp. 2476–2482.

Langtangen, H. P. A Primer on Scientific Programming with Python, 2nd Edition, Springer-Verlag, Berlin, Heidelber, 2011.