Projeto 3:
Álgebra Linear Avançada para Computação

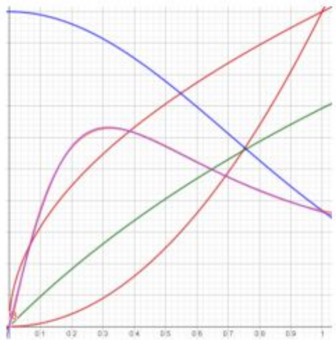
Marcos Heitor Carvalho de Oliveira
Bezaliel da Conceição Silva

Centro de Informática
UFPE
Universidade Federal de Pernambuco

Tema

Aproximação de uma função tomando por base uma função logarítmica, uma quadrática, uma exponencial e raiz quadrada. Utilize as seguintes funções:

●	$f1(x) = x^2$
●	$f2(x) = \ln(x+1)$
●	$f3(x) = \sqrt{x}$
●	$f4(x) = e^{-x^2}$



Para fazer uma aproximação da função $\frac{4t}{1+10t^2}$. Gere 100 pontos do gráfico dessa função com abscissas igualmente espaçadas entre 0 e 1. No final, exiba o erro residual ($e = ||Ax^* - b||$, onde x^* é a solução aproximada encontrada). Projeto em dupla.

Ferramentas usadas

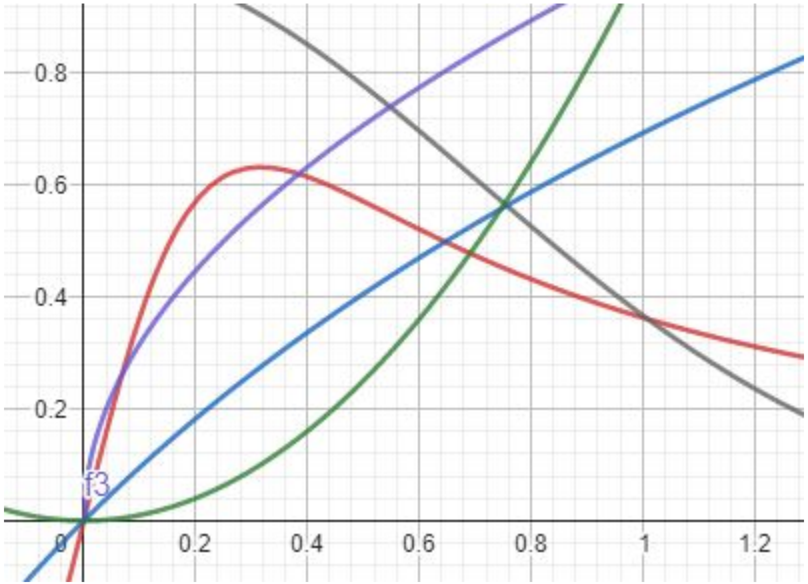
GeoGebra
for Learning Mathematics

python™

Interpretação do problema

$$f(x) = \frac{4x}{1 + 10x^2}$$

Interpretação do problema



$$f(x) = \frac{4x}{1 + 10x^2}$$

$$f1(x) = x^2$$

$$f2(x) = \ln(x + 1)$$

$$f3(x) = \sqrt{x}$$

$$f4(x) = e^{-x^2}$$

g(x) = a0 * f1 + a1 * f2 + a2 * f3 + a3 * f4

Como aproximar g(x) a f(x): MMQ

O **Método dos Mínimos Quadrados (MMQ)**, ou Mínimos Quadrados Ordinários (MQO) ou OLS (do inglês *Ordinary Least Squares*) é uma técnica de otimização matemática que procura encontrar o melhor ajuste para um conjunto de dados tentando minimizar a soma dos quadrados das diferenças entre o valor estimado e os dados observados (tais diferenças são chamadas resíduos).[1]

(x_i, y_i), $i = 1, \ldots, n$, onde x_i é uma variável independente e y_i é uma variável dependente cujo valor é encontrado por observação.

$f(x, \beta)$

$$r_i = y_i - f(x_i, \beta)$$

$$S = \sum_{i=1}^{n} r_i^2$$

Como minimizar S?

- Usando um otimizador
- Usando a matriz Moore-Penrose

Matriz de Moore-Penrose

In mathematics, and in particular linear algebra, the **Moore–Penrose inverse** A^+ of a matrix A is the most widely known generalization of the inverse matrix.[1][2][3][4] It was independently described by E. H. Moore[5] in 1920, Arne Bjerhammar[6] in 1951, and Roger Penrose[7] in 1955. Earlier, Erik Ivar Fredholm had introduced the concept of a pseudoinverse of integral operators in 1903. When referring to a matrix, the term pseudoinverse, without furthe specification, is often used to indicate the Moore–Penrose inverse. The term generalized inverse is sometimes used as a synonym for pseudoinverse.

A common use of the pseudoinverse is to compute a "best fit" (least squares) solution to a system of linear equations that lacks a solution (see below under § Applications). Another use is to find the minimum (Euclidean) norm solution to a system of linear equations with multiple solutions. The pseudoinverse facilitates the statement and proof of results in linear algebra.

The pseudo-inverse of a matrix A, denoted A^+, is defined as: "the matrix that 'solves' [the least-squares problem] $Ax = b$," i.e., if $\bar{x}$ is said solution, then A^+ is that matrix such that $\bar{x} = A^+b$.

It can be shown that if $Q_1 \Sigma Q_2^T = A$ is the singular value decomposition of A, then $A^+ = Q_2 \Sigma^+ Q_1^T$, where $Q_{1,2}$ are orthogonal matrices, Σ is a diagonal matrix consisting of A's so-called singular values, (followed, typically, by zeros), and then Σ^+ is simply the diagonal matrix consisting of the reciprocals of A's singular values (again, followed by zeros). [1]

Matriz de Moore-Penrose

A^+ exists for any matrix A , but, when the latter has full rank (that is, the rank of A is $\min\{m,n\}$), then A^+ can be expressed as a simple algebraic formula.

In particular, when A has linearly independent columns (and thus matrix A^*A is invertible), A^+ can be computed as

$$A^+ = (A^*A)^{-1}A^*.$$

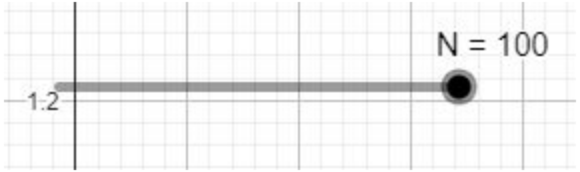
This particular pseudoinverse constitutes a *left inverse*, since, in this case, $A^+A = I$.

When A has linearly independent rows (matrix AA^* is invertible), A^+ can be computed as

$$A^+ = A^*(AA^*)^{-1}.$$

This is a *right inverse*, as $AA^+ = I$.

Aproximação da função



$$l4 = \text{Sequence}\left(\{f(n)\}, n, 0, 1, \frac{1}{N}\right)$$

0
0.03996003996
0.0796812749004
0.1189296333003
0.1574803149606
0.1951219512195
0.2316602316602
0.2669208770257
0.3007518796992
0.3330249768733
0.3636363636364
0.392506690455
0.4195804195804
0.4448246364414
0.4682274247492
0.4897959183673
0.5095541401274
0.5275407292475
0.5438066465257
0.5584129316679
0.5714285714286
0.5829285218598
0.5929919137466
0.6017004578156
0.6091370558376
0.6153846153846
0.6205250596659

<- Matriz (100 x 1) pontos

$$\text{stack1} = \text{Sequence}\left(\{f1(n), f2(n), f3(n), f4(n)\}, n, 0, 1, \frac{1}{N}\right)$$

0	0	0	1
0.0001	0.0099503308532	0.1	0.9999000049998
0.0004	0.0198026272962	0.1414213562373	0.9996000799893
0.0009	0.0295588022415	0.1732050807569	0.9991004048785
0.0016	0.0392207131533	0.2	0.9984012793176
0.0025	0.0487901641694	0.22360679775	0.9975031223975
0.0036	0.058268908124	0.2449489742783	0.996406472231
0.0049	0.0676586484738	0.2645751311065	0.9951119854158
0.0064	0.0769610411361	0.2828427124746	0.9936204363791
0.0081	0.0861776962411	0.3	0.9919327166056
0.01	0.0953101798043	0.3162277660168	0.9900498337492
0.0121	0.1043600153242	0.3316624790355	0.9879729106308
0.0144	0.113328685307	0.3464101615138	0.9857031841224
0.0169	0.1222176327242	0.3605551275464	0.9832420039193
0.0196	0.1310282624064	0.3741657386774	0.9805908312024
0.0225	0.1397619423752	0.3872983346207	0.9777512371933
0.0256	0.1484200051183	0.4	0.9747249016018
0.0289	0.1570037488097	0.4123105625618	0.9715136109703
0.0324	0.1655144384776	0.4242640687119	0.9681192569166
0.0361	0.1739533071234	0.4358898943541	0.9645438342768
0.04	0.182321556794	0.4472135955	0.9607894391523
0.0441	0.1906203596086	0.4582575694956	0.9568582668619
0.0484	0.1988508587452	0.4690415759823	0.9527526098032
0.0529	0.2070141693843	0.4795831523313	0.9484748552256
0.0576	0.2151113796169	0.4898979485566	0.9440274829178
0.0625	0.2231435513142	0.5	0.9394130628135

<- Matriz (100 x 4) pontos

Calculando a matriz de Moore-Penrose

$$\mathrm{m1} = \mathrm{Transpose}(\mathrm{stack1})\ \mathrm{stack1}$$

$$= \begin{pmatrix} 20.5033333 & 18.7801797677022 & 29.0735119873271 & 19.1311743005911 \\ 18.7801797677022 & 19.0725346814991 & 30.7261993475264 & 25.1146957384664 \\ 29.0735119873271 & 30.7261993475264 & 50.5 & 45.5018856042046 \\ 19.1311743005911 & 25.1146957384664 & 45.5018856042046 & 60.3816171931456 \end{pmatrix}$$

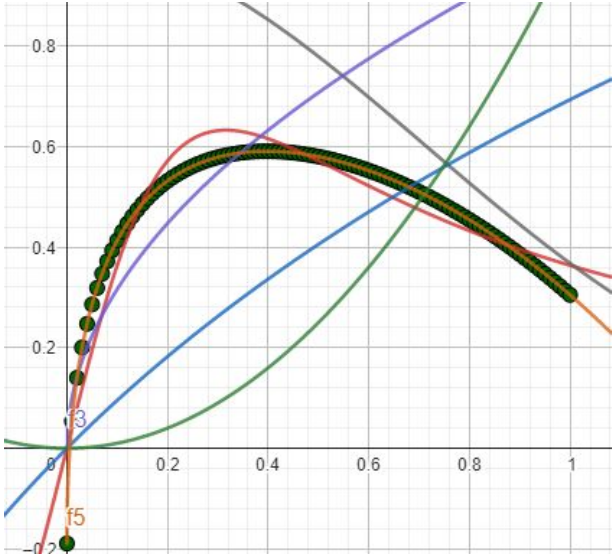
$$\mathrm{m2} = \mathrm{m1}^{-1}\ \mathrm{Transpose}(\mathrm{stack1})$$

$$= \begin{pmatrix} -0.3535995435709 & -0.034598542755 & 0.0538845281361 & 0.1054249171005 & 0.138396 \\ 5.1059781574206 & 2.0499862800612 & 1.1230205772435 & 0.5394145768621 & 0.12965 \\ -3.4435678365284 & -1.5012711891789 & -0.903938175811 & -0.5237305137794 & -0.253578 \\ 0.5998268691938 & 0.3061809178941 & 0.2135632953053 & 0.1534521816665 & 0.109848 \end{pmatrix}$$

Resolvendo o sistema

$$m3 = m2\ l4$$

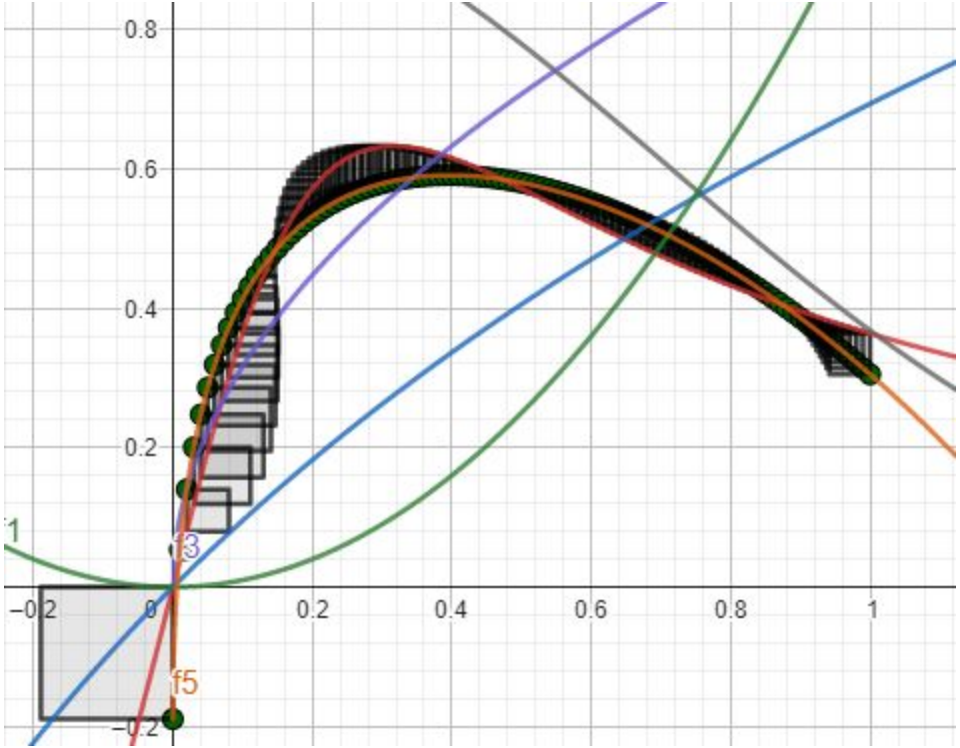
$$= \begin{pmatrix} -0.5363102956739 \\ -2.5668994264523 \\ 2.6901638333399 \\ -0.1893842575761 \end{pmatrix}$$



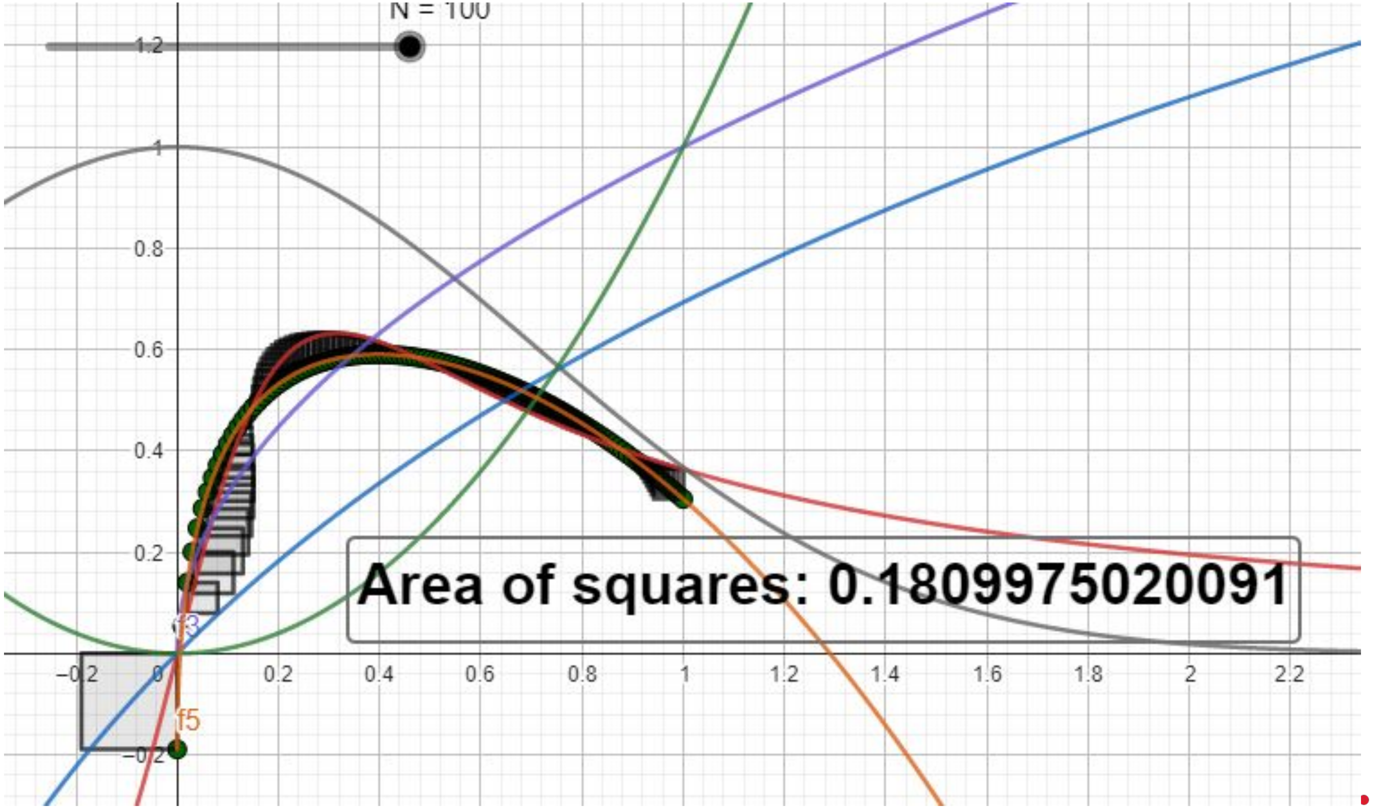
$$f5(x) = MR(1, 1, x)$$

$$= x^2\,(-0.5363102956739) + \ln(x+1)\,(-2.5668994264523) + \sqrt{x}\cdot 2.6901638333399 + e^{-x^2}\,(-0.1893842575761)$$

Erro gerado pela aproximação



Erro gerado pela aproximação



Confirmação em Python (otimizador)

opt.py > ...

```python
import numpy as np
from scipy.optimize import minimize

# Define the functions to be used in the approximation
def f1(x):
    return x**2

def f2(x):
    return np.log(x + 1)

def f3(x):
    return np.sqrt(x)

def f4(x):
    return np.exp(-x**2)

# Define the function to be approximated
def f(x):
    return 4*x / (1 + 10*x**2)

# Define the linear combination of functions
def F(x, a):
    return a[0]*f1(x) + a[1]*f2(x) + a[2]*f3(x) + a[3]*f4(x)

# Define the sum of squared errors
def S(a):
    error = 0
    for i in range(len(x_data)):
        error += (F(x_data[i], a) - y_data[i])**2
    return error
```

```python
# Define the initial guess for the coefficients
a0 = [1, 1, 1, 1]

# Define the data points
x_data = np.linspace(0, 1, 101)
y_data = f(x_data)

# Find the coefficients that minimize the sum of squared errors
res = minimize(S, a0)

# Print the coefficients and the approximate function
print("Coefficients:", res.x)
print("Approximate function: F(x) = {:.3f}x^2 + {:.3f}ln(x+1) + {:.3f}sqrt(x) + {:.3f}e^(-x^2)".format(*res.x))
print(S(res.x))
```

```
Coefficients: [-0.53631113 -2.5668955   2.69016176 -0.18938406]
Approximate function: F(x) = -0.536x^2 + -2.567ln(x+1) + 2.690sqrt(x) + -0.189e^(-x^2)
0.1809975020095418
```

Confirmação em Python (matriz)

optmat.py > ...

```python
import numpy as np

# Define the functions
f1 = lambda x: x**2
f2 = lambda x: np.log(x+1)
f3 = lambda x: np.sqrt(x)
f4 = lambda x: np.exp(-x**2)

f = lambda x: 4*x/(1 + 10*x**2)

# Define the data
x = np.linspace(0, 1, 101)
y = f(x)

# Define the matrix A
A = np.vstack([f1(x), f2(x), f3(x), f4(x)]).T

# Compute the least squares solution for the coefficients
c = np.linalg.pinv(A) @ y

# Construct the approximate function
f = lambda x: c[0]*f1(x) + c[1]*f2(x) + c[2]*f3(x) + c[3]*f4(x)

# Print the coefficients and the approximate function
print("Coefficients:", c)
print("Approximate function:", f)
```

```
Coefficients: [-0.5363103  -2.56689943  2.69016383 -0.18938426]
Approximate function: <function <lambda> at 0x071D1F18>
```

Link do projeto

https://www.geogebra.org/calculator/kmwpwcyc