Derivada da exponencial

Calculando graficamente a derivada da exponencial

Praciano-Pereira, Tarcisio
Universidade Estadual Vale do Acaraú
tarcisio@sobralmatematica.org

14 de setembro de 2026
Preprints da Sobral Matemática no. 2025.01
Editor: Tarcisio Praciano-Pereira
tarcisio@sobralmatematica.org

Resumo

Este artigo se baseia em cinco gráficos, da primeira bissetriz dos eixos, do gráfico do logaritmo neperiano, da exponencial que vou obter por simetria do logaritmo neperiano em torno da primeira bissetriz dos eixos, das retas tangentes aos gráficos do logaritmo e da exponencial. O logaritmo foi construído usando somas de Riemann uniformes com um programa escrito em `Python`. Como *bi-produto* estou usando as *somas de Riemann uniformes* para mostrar que a *propriedade distributiva do produto relativamente à soma*, longe de ser um item abstrato, é um método para otimizar alguns algoritmos.

palavras chaves: derivada da função inversa, exponencial, logaritmo neperiano, propriedade distributiva do produto relativamente à soma, somas de Riemann uinformes

1 Introdução

Este artigo se baseia em cinco gráficos:

1. O gráfico da primeira bissetriz dos eixos,
2. o gráfico da hipérbole padrão, que para mim é $y = f(x) = \frac{1}{x}$ que, sobre qualquer intervalo fechado do seu domínio, é uniformemente contínua o que me permite calcular uma aproximação de sua integral de Riemann usando apenas *somas de Riemann uniformes* com as quais posso deduzir a propriedade fundamental do logaritmo neperiano. Este gráfico não vai aparecer, mas vai ser usada a função $y = f(x) = \frac{1}{x}$ nos programas. Alguns exemplos de *somas de Riemann uniformes* você pode encontrar em (PRACIANO-PEREIRA, 2025) voltadas para calcular aproximadamente a integral de polinômos.
3. O gráfico do logaritmo neperiano obtido calculando alguns pontos com a definição integral, aproximadamente, usando somas de Riemann uniformes, com um programa escrito em `Python`,
4. o gráfico da exponencial obtido por simetria em torno da primeira bissetriz dos eixos usando a tabela de logaritmos construída pelo programa em `Python`,
5. os gráficos das retas tangentes ao logaritmo e à exponencial.

Para obter o gráfico do logaritmo neperiano eu construí uma tabela usando um programa escrito em `Python` e apresento o programa no momento certo. O gráfico da exponencial foi obtido usando o editor de textos `joe` que me oferece o `modo coluna` e me permite trocar as duas colunas do gráfico do logaritmo para assim produzir o gráfico da exponencial. A troca de colunas corresponde a fazer uma simetria em torno da primeira bissetriz dos eixos o que é válido porque a exponencial é a função inversa do logaritmo.

Escolhi um ponto arbitrário $(a, \ln(a))$ sobre o gráfico do logaritmo e vou passar por este ponto o gráfico da reta tangente ao logaritmo também construído a partir duma tabela e, novamente, usando `joe` troquei as duas colunas do gráfico da tangente para obter o gráfico da reta tangente ao gráfico da exponencial no ponto $(\ln(a), a)$ é quando concluí o cálculo da derivada da exponencial. Claro, estou usando o *teorema da derivada da função inversa*, apenas o estou fazendo graficamente o que pode ser usado para compreender melhor o teorema e inclusive usar o mesmo método geométrico com outro par de funções inversas.

Como *bi-produto* estou usando o programa que calcula as *somas de Riemann uniformes* para dar um *exemplo de uso prático* da *propriedade distributiva do produto relativamente à soma* como um método de otimização dum algoritmo, neste caso da *soma de Riemann uniforme*.

Eu não usei, mas me inspirei em (COURANT, 1961), em que aprendi Cálculo começando pela *integral* e usando *somas de Riemann*, é o método que estou usando para construir o logaritmo neperiano.

Em resumo, neste artigo o leitor encontra

1. *somas de Riemann* inspirado no livro do Courant(COURANT, 1961),
2. dois programas escritos em `Python`, aqui apresentados, para construir as tabelas das funções cujos gráficos serão exibidos,
3. a formulação gráfica do *teorema da função inversa* provando, graficamente, que $\frac{de^x}{dx} = e^x$,
4. que a *propriedade distributiva do produto relativamente à soma* não é somente um item teórico mas também um instrumento prático, serve para otimizar as *somas de Riemann uniformes* quando estas puderem ser usadas.

O meu objetivo é mostrar que alguns dos conceitos mais difíceis do Cálculo podem ser apresentados com extremo rigor e ao mesmo tempo de forma didática mostrando que o Cálculo não precisa ser o monstro que costuma ser.

Apresentando os programs no texto, incluindo comentários, dou condições para que o leitor reproduza todo o conteúdo do artigo fazendo as alterações que lhe parecerem oportunas.

2 O logaritmo neperiano

Como o logaritmo neperiano é diferenciável e estritamente crescente então tem uma função inversa, a exponencial, que vou obter, graficamente, fazendo uma simetria em torno da primeira bissetriz. Sendo a exponencial inversa do logaritmo ela herda entre outras propriedades a diferenciabilidade e vou calcular sua derivada graficamente. Estou apenas construindo geometricamente o *teorema da derivada da função inversa*, a novidade é que o estou fazendo de forma geométrica portanto acrescentando uma visão gráfica do teorema para o caso

da exponencial e também algoritmicamente, estou usando pequenos programas escritos em `Python`, inseridos no texto,

Como a exponencial é inversa do logaritmo, então vou começar por lembrar a forma *moderna* de definir logaritmo que aparece nos livros de Cálculo desde início do século passado, confira (COURANT, 1961) que foi publicado por volta de 1940.

Esta "*forma moderna*" passa pela definição:

$$\ln(x) = \int_1^x \frac{1}{t} dt \tag{1}$$

com a qual é bem simples provar a *lei fundamental dos logaritmos*,

$$ln(ab) = ln(a) + ln(b)$$

para dois números reais positivos quaisquer, e, por definição mesmo,

$$\ln(1) = 0 = \int_1^1 \frac{1}{t} dt; \tag{2}$$

Esta *integral* é daquelas para as quais não temos regras para o cálculo, então a única forma de obter o valor

$$ln(x) = \int_1^x \frac{1}{t} dt \tag{3}$$

é com o cálculo aproximado desta integral e dos métodos menos efetivos são as *somas de Riemann* mas, neste caso, me leva rapidamente a demonstrar a *lei fundamental dos logaritmos* e como consequência imediata vai me permitir a definição da exponencial, graficamente.

Se o leitor ainda não tiver estudado *integral* vou lhe mostrar o que é uma *soma de Riemann* e em seguida aplicar no estudo do logaritmo, confira exemplos de *soma de Riemann*(PRACIANO-PEREIRA, 2025).

A figura (fig. 1), na página 3 representa uma função cuja *integral aproximada* está sendo descrita por coleção de retângulos, alguns contidos no gráfico da função e alguns outros levemente se sobrepondo ao gráfico de modo que o que se perde em alguns casos e compensado por outros. Mas esta compensação é secundária, o que leva a *aproximação* é o *refinamento* do conjunto das bases dos retângulos.

A *soma de Riemann* é a soma das áreas dos retângulos que cobrem a região limitada pelo gráfico da função e o eixo OX, no caso de funções univariadas. A figura (fig. 1, na página 3 lhe mostra o gráfico de uma função $y = f(x)$ sobre o

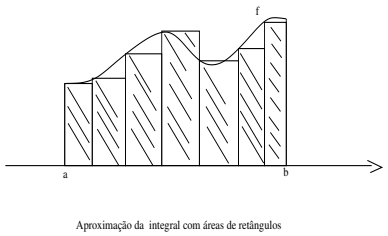


Figura 1: Soma de Riemann

intervalo $[a, b]$ que foi *particionado* por uma sucessão de pontos

$$a = x_0, \ldots, x_k, \ldots, x_N = b; \tag{4}$$

$$f(x_0)\Delta x_0 + \cdots + f(x_{N-1})\Delta x_{N-1}; \tag{5}$$

$$\int_a^b f(x)dx \approx \sum_{k=0}^{N-1} f(x_k)\Delta x_k; \tag{6}$$

O valor aproximado da integral é a soma das áreas dos retângulos.

Para o *cálculo aproximado* de integrais há outros métodos mais efetivos do que *somas de Riemann*, mas para alguns tipos de função uma particularização do método, das *somas de Riemann uniformes*, consigo resultados muito bons e inclusive *fórmulas precisas* para o cálculo da integral, é o caso das funções polinomiais. No caso da função $f(x) = \frac{1}{x}$ eu vou obter um bom resultado com as *somas de Riemann uniformes* e vou lhe mostrar isto no momento certo.

A diferença entre este *caso particular*, das *somas de Riemann uniformes*, e o caso geral é que o intervalo em que a função está definida é dividido em subintervalos de mesma medida e é quando as *somas de Riemann* se torna mais efetivas no cálculo com programas de computador porque posso usar a *propriedade distributiva do produto relativamente à soma* e reduzir o tempo de computação significativamente. A *propriedade distributiva* está ligada a duas operações neste caso da multiplicação relativamente à soma, mas vou resumir a frase evitando o complemento uma vez que é o único caso em que a vou usar neste contexto.

2.1 Propriedade distributiva, teste computacional

Eu calculei a soma de Riemann para o calculo aproximado da integral de $f(x) = x^2$ no intervalo $[0, 1]$ que vale $\frac{1}{3}$

$$\int_0^1 t^2 dt = \frac{1}{3} \tag{7}$$

com $N = 100000000$ e joguei o programa dentro do aplicativo "`time`" que calcula o tempo de processamento obtendo os resultados abaixo em dois computadores diferentes:

- `Python, sem propriedade distributiva, N= 1000000000`
  `0.33333333383340125`
  `real 2m17,892s`

```
user    2m17,677s
sys     0m0,212s
-------------------------
real    2m33,569s
user    2m33,488s
sys     0m0,045s
```

- `Python,com propriedade distributiva, N=  1000000000`

```
0.3333333338332974
real    2m11,669s
user    2m11,619s
sys     0m0,033s
--------------------
real    2m29,478s
user    2m29,452s
sys     0m0,016s
-----------------
real    2m10,995s
user    2m10,957s
sys     0m0,028s
```

Tive que usar $N = 1000000000$, bastante grande, do contrário não haveria diferença significativa num cálculo trivial como este. Importante fazer este registro porque as propriedades *comutativa, distributivas* e *associativa* são, com frequência, consideradas itens de *abstração inútil* e este exemplo mostra uma diferença significativa entre *fazer ou não* uso da *propriedade distributiva* tem diferença, mesmo num pequeno programa como este do cálculo de *soma de Riemann*. A terceira estatística, "com propriedade distributiva", eu rodei num computador em que, simultaneamente, fazia uma renovação (upgrade) da minha distribuição, portanto com muito mais coisa ocupando o sistema, `Debian/gnu/linux`, e ela se revelou a melhor das três, aqui entra a vantagem de usar uma *distribuição Linux*...

O código em `Python` para calcular *somas de Riemann uniformes* é

```
#! /usr/bin/python3
def f(x):
    if (x*(-x) == 0):
          return 0;
    else:
          return 1/x;
## soma de Riemann sem a propriedade distributiva
def SomaRiemannSemPD(f,a,b,N):
    I = range(0,N+1); soma = 0;
    Delta=(b-a)/N;
    for s in I:
        soma += f(a+s*Delta)*Delta;
    return soma;

## soma de Riemann com a propriedade distributiva
def SomaRiemann(f,a,b,N):
    I = range(0,N+1); soma = 0;
    Delta=(b-a)/N;
    for s in I:
        soma += f(a+s*Delta);
    return soma*Delta;

print(SomaRiemann(f,1,2,1000000));
```

Este exemplo lhe mostra uma vantagem de `Python` sobre a uma grande maioria de linguagens de programação, eu posso passar o símbolo da função f como uma *informação* para

o *método* `SomaRiemann(f,a,b,N)`. Se você tiver este código no arquivo `SomaRiemann.py` basta escrever na `linha de comandos`:

`SomaRiemann(f,1,2,1000000)`

e se você estiver usando uma `distribuição Linux`, com quase toda certeza `Python3` estará instalado e pode executar, na linha de comandos;

`time SomaRiemann(f,1,2,1000000)`

para obter o tempo de processamento do programa.

Retornando à função que me interessa, $f(x) = \frac{1}{x}$, a soma de Riemann fica descrita pela seguinte lista de equações depois da qual vou fazer comentários:

$$\int_a^b f(x)dx \approx \sum_{k=0}^{N-1} f(x_k)\Delta; \tag{8}$$

$$\Delta = \tfrac{b-a}{N}; x_k = a + k\Delta; \text{ uma p.a.} \tag{9}$$

$$\int_a^b f(x)dx \approx \sum_{k=0}^{N-1} f(x_k)\Delta = \Delta \sum_{k=0}^{N-1} f(a + k\Delta); \tag{10}$$

$$\int_a^b \tfrac{1}{x}dx \approx \sum_{k=0}^{N-1} \tfrac{1}{a+k\Delta}\Delta = \Delta \sum_{k=0}^{N-1} \tfrac{1}{a+k\Delta}; \tag{11}$$

- Na equação (eq. 8) eu escrevi a soma de Riemann para uma função genérica $f(x)$ sobre o intervalo $[a, b]$, e estou usando uma notação diferente dos livros de Cálculo, Δx, porque agora as bases dos retângulos têm a mesma medida, Δ,
- na (eq. 9) descrevi o *salto* $\Delta = \frac{b-a}{N}$ em que N é a quantidade de subintervalos em que o intervalo $[a, b]$ está dividido, sendo Δ a medida comum a todos os subintervalos o que me permite na
- (eq. 10) colocar Δ em evidência, *propriedade distributiva*. Então o cálculo se resume em somar todas as alturas $f(a + k\Delta)$ para depois multiplicar por Δ, pela *propriedade distributiva*.

Vou aplicar esta técnica com a função $f(x) = \frac{1}{x}$ me restringindo à semirreta positiva, com zero excluído:

$$\int_a^b \tfrac{1}{x}dx \approx \sum_{k=0}^{N-1} \tfrac{1}{a+k\Delta}\Delta = \tag{12}$$

$$= \Delta \sum_{k=0}^{N-1} \tfrac{1}{a+k\Delta} = \tfrac{\Delta}{a} \sum_{k=0}^{N-1} \tfrac{1}{1+k\Delta/a}; \tag{13}$$

$$\Delta' = \tfrac{\Delta}{a} \implies \Delta' \sum_{k=0}^{N-1} \tfrac{1}{1+k\Delta'} \approx \int_1^{b/a} \tfrac{1}{x}dx \tag{14}$$

1. Na equação (eq. 12) escrevi a integral que desejo calcular e a expressão aproximada da mesma por uma soma de Riemann uniforme com $\Delta = \frac{b-a}{N}$.
2. Na equação (eq. 13) coloquei $\frac{\Delta}{a}$ em evidência, sem alterar o valor da soma de Riemann, mas agora o ponto inicial é 1 e vai ser expresso sob forma de integral.
3. Na equação (eq. 14) redefini o salto Δ com o novo salto $\Delta' = \frac{\Delta}{a}$, lembrando que $a > 0$, redefini assim a soma de Riemann para o intervalo $[1, \frac{b}{a}]$, ainda igual à soma de Riemann anterior.

Com estes cálculos obtive uma propriedade das funções do tipo $f(x) = \frac{K}{x}$, obviamente, usei $K = 1$, mas você pode refazer os mesmo cálculos com qualquer valor $K \neq 0$ e agora me interessa $K = 1$.

$$\int_a^b \frac{1}{x}dx = \int_1^{b/a} \frac{1}{x}dx \tag{15}$$

estabelecendo que na integral $\int_a^b \frac{1}{x}dx$ posso *cancelar* o limite superior b da integral. Contas semelhantes mostram que também posso *cancelar* o limite inferior a da integral, mas isto não

me interessa aqui. Esta propriedade vale para qualquer função do tipo $f(x) = \frac{K}{x}$, $K \neq 0$ mas somente me interessam aquelas em $K > 0$ e neste momento estou trabalhando com $K = 1$.

Estes cálculos me levam à propriedade fundamental do logaritmo:

$$ln(x) = \int_1^x \frac{1}{t} dt \text{ por definição ;} \tag{16}$$

$$ln(ab) = \int_1^{ab} \frac{1}{t} dt = \int_1^a \frac{1}{t} dt + \int_a^{ab} \frac{1}{t} dt = \int_1^a \frac{1}{t} dt + \int_1^b \frac{1}{t} dt = ln(a) + ln(b); \tag{17}$$

que mostra que o logaritmo é um *morfismo* do grupo multiplicativo $(\mathbf{R}^{**}, \cdot)$ sobre o grupo aditivo $(\mathbf{R}, +)$, um *isomorfismo* de grupos. Aliás, esta *propriedade dos logaritmos* já salvariam a vida dos logaritmos dentro do *Ensino Médio* deixando de ser a *ultrapassada máquina de calcular*. Mudando a base, digo, mudando o valor de K, um novo *isomorfismo de grupo multiplicativo* estaria sendo apresentado! E a fórmula de mudança de base é simples,

$$\int_1^A \frac{K}{t} dt = K \int_1^A \frac{1}{t} dt = K \ln(A); \tag{18}$$

$$K \int_1^A \frac{1}{t} dt = 1 \implies K = \frac{1}{\ln(A)}; \tag{19}$$

mostra o valor de K para transformar o logaritmo neperiano no logartimo na base de sua escolha. $K = \frac{1}{\ln(10)}$ transforma l logaritmo neperiano no logaritmo decimal.

3 A derivada da exponencial

O meu objetivo, neste artigo é calcular graficamente a derivada da função exponencial e passo pela construção do logaritmo neperiano como função inversa da exponencial. No algoritmo usei a propriedade fundamental da função $f(x) = \frac{1}{x}$. O código em `Python` é

```
#! /usr/bin/python3
def f(x):
        if (x*(-x) == 0):
                return 0;
        else:
                return 1/x;

def SomaRiemann(f,a,b,N):
        I = range(0,N+1); soma = 0;
        Delta=(b-a)/N;
        for s in I:
                soma += f(a+s*Delta);
        return soma*Delta;

I = range(1,100);
a=1;b=2;base=0;N=1000000;
for a in I:
        print(a,base);
        base = base + SomaRiemann(f,1,b/a,N);
        a=a+1;b=b+1;
```

`SomaRiemann(f,1,b/a,N)` está calculando, aproximadamente, a integral

$$\int_1^{b/a} \frac{1}{t} dt = \int_a^b \frac{1}{t} dt \tag{20}$$

O programa memoriza o valor do trecho anterior da integral na variável `base` e atualiza as variáveis `a,b` para o próximo intervalo percorrendo os inteiros $\{1, 2, \ldots, 99\}$, sempre `b=a+1`.

No código usei `I = range(1,100);` que me produziu uma tabela de logaritmos para os inteiros $\{1, \cdots, 99\}$ em 14 milissegundos e testei a acuracidade do cálculo com

```
from math import exp
exp(4.595124522461644)
99.00046256145896
```

porque a última linha de minha tabela de logaritmos é

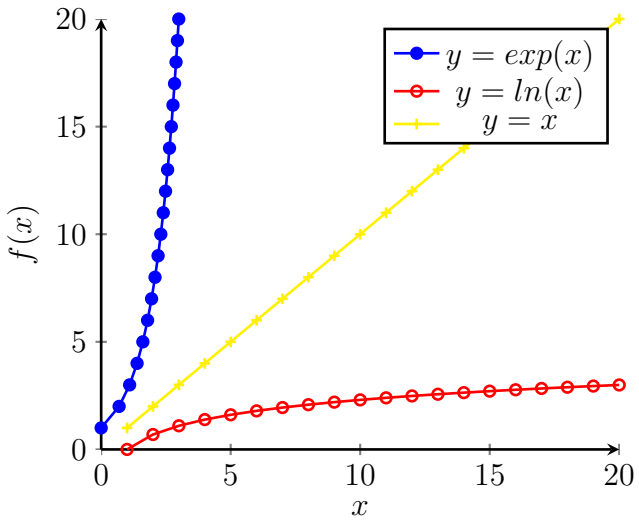
```
99 4.595124522461644
```

com um erro inferior a 0.0005.

4 A função inversa do ln

Fiz algumas modificações no código com que calculei as somas de Riemann para, usando um único código, produzir tabelas de *logaritmo, primeira bissetriz dos eixos, inversa do logaritmo*. Razão, produzir facilmente tabelas com o mesmo número de linhas e obter um gráfico decente usando LaTeX.

No caso da *inversa do logaritmo* usei o editor `joe` que me oferece o `modo coluna` e assim pude trocar as posições das colunas na tabela do *logaritmo* para obter o gráfico da função inversa e coloquei estas tabelas num código `PGFPlots` que me produziu o gráfico seguinte.



Observe, neste ponto, que tenho, e lhe ofereço, um algoritmo para *construir* tanto o logaritmo como a exponencial. Construo o logaritmo com *soma de Riemann uniforme* e uso um editor para trocar as colunas duma tabela para obter a exponencial...

5 A derivada da exponencial

Como estou usando gráficos como expressão da Matemática, vou agora completar a figura anterior incluindo a reta tangente ao gráfico do *logaritmo*. A equação da reta tangente é a reta que passa no ponto $(a, ln(a)$ com coeficiente angular $m = \frac{1}{a}$

$$y = ln(a) + m(x - a); m = \frac{dln(x)}{dx}_{x=a}; y = ln(a) + \frac{1}{a}(x - a); a = 3; \tag{21}$$

como se trata dum gráfico não posso trabalhar com a expressão formal em que aparece um *ponto arbitrário* $(a, ln(a)$, tenho que usar $a = 3$. Entretanto fica claro que a construção vale

para qualquer que seja a no domínio do logaritmo, e você tem o código fonte e pode escolher o valor de a, dentro do domínio do logaritmo, o que me permite dizer que estou trabalhando com uma *fórmula geral*.

Usei o seguinte código em `Python` para criar uma tabela a ser repassada para `pgfplot`

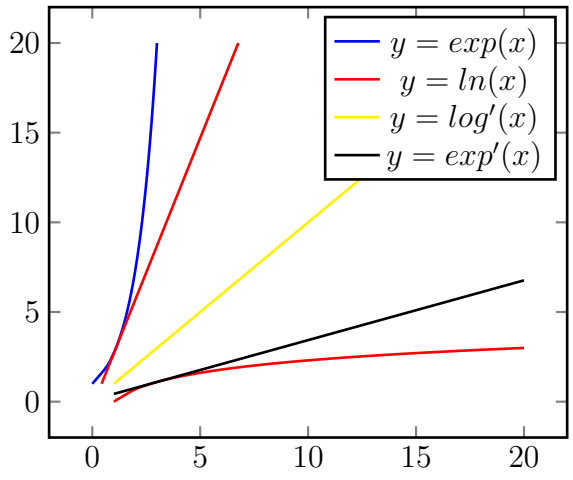
```
def f(x):
    if (x*(-x) == 0):
       return 0;
    else:
       return 1/x;

def RetaLog(x):
    from math import log;
    return ln(3) + (1/3.0)*(x-3);

I = range(1,21)
for a in I:
    print(a,RetaLog(a));
```

este código vai criar uma tabela com duas entradas por 20 linhas com os valores da reta tangente ao gráfico de $y = \ln(x)$ no ponto $(3, \ln(3))$.

Usando o editor `joe` no modo `coluna` troquei a posição das duas colunas criando uma nova tabela que vai me produzir uma reta passando pelo ponto $(\ln(3), 3)$ simétrica a anterior relativamente à primeira bissetriz, e, consequentemente, tangente ao gráfico da exponencial no ponto $(\ln(3), 3)$ e com coeficiente angular $m = 3$ devido a que estas retas são simétricas relativamente à primeira bissetriz.

Neste ponto já tenho o resultado, o valor da derivada da exponencial no ponto $(\ln(3), 3)$ é $m = 3$ que coincide com o valor da exponencial no ponto de tangência. Como este resultado vale para qualquer seja a no domínio do logaritmo, então a derivada de $y = e^x$ no ponto a é e^a provando que

$$\frac{de^x}{dx} = e^x; \tag{22}$$



6 Comentários finais

Estou apresentando como referência bibliográfica o livro do Courant (COURANT, 1961), que é um texto muito didático e começa o estudo das funções pela integração que considero muito

sugestivo uma vez que um simples programa, como algum destes que apresentei, permite que o estudante já consiga calcular integrais aproximadamente e, no caso das funções polinomiais chegue rapidamente ao limite com uma fórmula exata. O livro (COURANT, 1961) fez isto em 1940 quando não tinha o poder computacional que temos hoje, estudei no livro do Courant na década de 60 e calculei integrais, aproximadamente, usando um instrumento hoje praticamente desconhecido, *régua de cálculo*.

Apresento um *algoritmo*, um programa em `Python`, para construir o *logaritmo neperiano* e, consequentemente, estou também mostrando como construir a *exponencial* como alternativa a uma série de potências.

referências

COURANT, Richard. **Differential and Integral Calculus I**. Edição: Blackie e Son Limited. [S.l.]: Blackie e Son Limited - London, 1961.

PRACIANO-PEREIRA, Tarcisio. Somas polinomiais notáveis. http://www.sobralmatematica.org/preprints/2025/preprint_2025_02.pdf. [S.l.], 18 abr. 2025. Disponível em: ¡http://www.sobralmatematica.org/preprints/2025/preprint_2025_02.pdf¿.