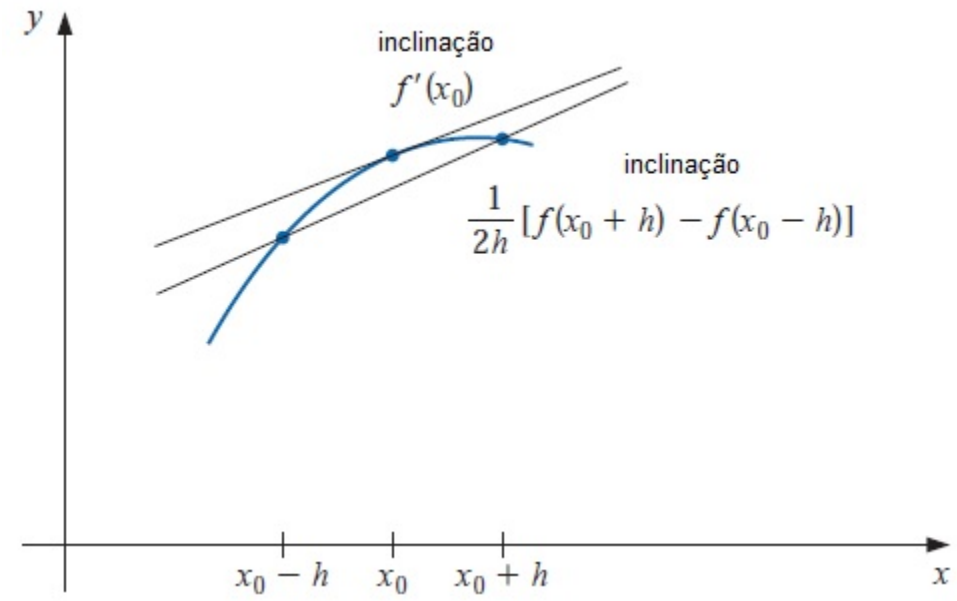
derivacao_numerica

May 21, 2023

#

Derivação Numérica



A derivada de uma função f em x_0 é

$$f'(x_0) = lim_{h\to 0}\frac{f(x_0+h)-f(x)}{h}$$

Se considerarmos a fórmula acima sem o limite, temos uma forma óbvia de gerar aproximações para $f'(x_0)$. Para tanto, supomos x_0 e x_1 tais que $x_0 \in (a,b)$ em que $f \in C^2[a,b]$ e $x_1 = x_0 + h$ para algum $h \neq 0$ que seja suficientemente pequeno para garantir que $x_1 \in [a,b]$.

Analisemos o primeiro polinômio de Lagrange $P_{0,1}(x)$ de f determinado por x_0 e x_1 juntamente com seu erro:

$$f(x) = P_{0,1}(x) + \frac{(x-x_0)(x-x_1)}{2!}f''(\xi(x)) =$$

$$= f(x_0)\frac{x-x_0-h}{-h} + f(x_0+h)\frac{x-x_0}{h} + \frac{(x-x_0)(x-x_0-h)}{2}f''(\xi(x)),$$

para algum $\xi(x)$ em $[a, b]$. A derivação resulta em

$$f'(x) = \frac{f(x_0 + h)}{h} + \frac{f(x_0)}{-h} + \frac{2(x - x_0) - h}{2} f''(\xi(x)) + \frac{(x - x_0)(x - x_0 - h)}{2} D_x(f''(\xi(x)))$$

de modo que

$$f'(x) \approx \frac{f(x_0 + h) - f(x_0)}{h}.$$

Um problema com a estimativa

$$f'(x) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

é que não temos informação sobre $D_x(f''(\xi(x)))$ e, portanto, o erro de truncamento não pode ser estimado. Contudo, fazendo $x = x_0$, temos que

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{|h|}{2} f''(\xi).$$

Tal fórmula é conhecida como **fórmula de diferenças progressivas**. Além disso podemos considerar tanto $h > 0$ como $h < 0$.

0.0.1 Exemplo:

Seja $f(x) = ln(x)$ e $x_0 = 1.8$. Apresente as estimativas para para $f'(1.8)$ considerando $h = 0.1, 0.01$ e 0.001 juntamente com os erros exato e limitante $\left(\frac{|h|}{2} M, M_h > |f''(\xi(x))_h|\right)$.

Solução: Temos que

$$f(x) = ln(x) \Rightarrow f'(x) = \frac{1}{x} \Rightarrow f'(1.8) = \frac{1}{1.8} = 0.5555555555555556.$$

Além disso, temos que $f''(x) = -\frac{1}{x^2}$. Os resultados estão expressos na tabela a seguir:

h	$\frac{f(1.8+h)-f(1.8)}{h}$	$e_{max} = \frac{h}{2} M_h$	e_{ext}
0.1	0.5406722127027574	0.0173010380622837	0.0148833428527981
0.01	0.5540180375615322	0.0015605006085952	0.0015375179940234
0.01	0.5554012916999529	0.0001544925984141	0.0001542638556027

Note que e_{max} e e_{ext} se aproximam à medida que h diminui. Consideramos que M_h ocorre em $x_0 - h$ uma vez que $|f''(x)|$ é monótona decrescente para $x > 0$.

[ ]:
```
# Definindo as funções
import numpy as np

x0,h = 1.8,0.001
```

```
def f(x):
    return np.log(x)

def d_1f(x):
    return 1/x

def d_2f(x):
    return -1/x**2
```

[ ]:
```
estimativa = (f(x0+h)-f(x0))/h
print(estimativa)

e_max = (h/2)*abs(d_2f(x0-h)) # x0-h caso h > 0 e x0+h caso h<0 (execute a célula anterior) (f monótona)
print(round(e_max,16))

e_ext = abs(d_1f(x0)-estimativa)
print(round(e_ext,16))
```

Seguindo as mesmas ideias postas acima podemos deduzir mais fórmulas para aproximarmos a derivada de f em um ponto x_0. Algumas das mais conhecidas são:

0.1 Fórmulas de Três Pontos

Considerando o ponto x_0 e incremento h, temos:

$$f'(x_0) = \frac{1}{2h}(-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)) + \frac{h^2}{3}f^{(3)}(\xi_0)$$

em que ξ_0 entre x_0 e $x_0 + 2h$

e

$$f'(x_0) = \frac{1}{2h}(f(x_0 + h) - f(x_0 - h)) - \frac{h^2}{6}f^{(3)}(\xi_1)$$

em que ξ_1 entre $x_0 - h$ e $x_0 + h$.

0.1.1 Exemplo:

Considere a função $f(x) = ln(x)$. Use as fórmulas de três pontos determine $f'(1.8)$ considerando $h = 0.1$ e $h = -0.1$. Obtenha cotas para o erro e compare com o valor real.

Solução: Temos que

$$f(x) = ln(x) \Rightarrow f'(x) = \frac{1}{x} \Rightarrow f'(1.8) = \frac{1}{1.8} = 0.5555555555555556.$$

Além disso, temos que $f^{(3)}(x) = \frac{2}{x^3}$. Os resultados estão expressos nas tabelas a seguir:

Considerando

$$f'(x_0) = \frac{1}{2h}(-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)) + \frac{h^2}{3}f^{(3)}(\xi_0)$$

temos

h	$\approx f'(x_0)$	$e_{max} = \frac{h^2}{3}M_h$	e_{ext}
0.1	0.5545418471163838	0.0013569441617477	0.0010137084391718
-0.1	0.5542530985170565	0.0013569441617477	0.0013024570384991

[ ]:
```
# Definindo as funções
import numpy as np

x0,h = 1.8,0.1

def f(x):
    return np.log(x)

def d_1f(x):
    return 1/x

def d_3f(x):
    return 2/x**3
```

[ ]:
```
estimativa = (-3*f(x0) + 4*f(x0+h)-f(x0+2*h))/(2*h)
print(estimativa)

e_max = ((h**2)/3)*abs(d_3f(x0-h)) # x0-h caso h > 0 e x0+h caso h<0 (execute a célula anterior) (f monótona)
print(round(e_max,16))

e_ext = abs(d_1f(x0)-estimativa)
print(round(e_ext,16))
```

Considerando

$$f'(x_0) = \frac{1}{2h}(f(x_0 + h) - f(x_0 - h)) - \frac{h^2}{6}f^{(3)}(\xi_1)$$

temos

h	$\approx f'(x_0)$	$e_{max} = \frac{h^2}{6}M_h$	e_{ext}
0.1	0.5561281755511222	0.0006784720808739	0.0005726199955666
-0.1	0.5561281755511222	0.0006784720808739	0.0005726199955666

Obviamente, os valores são os mesmos uma vez que a fórmula de estimativa é simétrica em relação a h.

[ ]:
```
# Definindo as funções
import numpy as np

x0,h = 1.8,0.1

def f(x):
    return np.log(x)

def d_1f(x):
    return 1/x

def d_3f(x):
    return 2/x**3
```

[ ]:
```
estimativa = (f(x0+h) - f(x0-h))/(2*h)
print(estimativa)

e_max = ((h**2)/6)*abs(d_3f(x0-h)) # x0-h caso h > 0 e x0+h caso h<0 (execute a célula anterior) (f monótona)
print(round(e_max,16))

e_ext = abs(d_1f(x0)-estimativa)
print(round(e_ext,16))
```

0.2 Fórmulas de Cinco Pontos

Considerando o ponto x_0 e incremento h, temos:

$$f'(x_0) = \frac{1}{12h}(f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)) + \frac{h^4}{30}f^{(5)}(\xi_0)$$

em que ξ_0 entre $x_0 - 2h$ e $x_0 + 2h$

e

$$f'(x_0) = \frac{1}{12h}(-25f(x_0) + 48f(x_0 + h) - 36f(x_0 + 2h) + 16f(x_0 + 3h) - 3f(x_0 + 4h)) - \frac{h^4}{5}f^{(5)}(\xi_1)$$

em que ξ_1 entre x_0 e $x_0 + 4h$.

0.2.1 Exemplo:

Considere a função $f(x) = ln(x)$. Use as fórmulas de três pontos determine $f'(1.8)$ considerando $h = 0.1$. Obtenha cotas para o erro e compare com o valor real.

Solução: Temos que

$$f(x) = ln(x) \Rightarrow f'(x) = \frac{1}{x} \Rightarrow f'(1.8) = \frac{1}{1.8} = 0.5555555555555556.$$

Além disso, temos que $f^{(5)}(x) = \frac{24}{x^5}$. Os resultados estão expressos nas tabelas a seguir:

Considerando

$$f'(x_0) = \frac{1}{12h}(f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)) + \frac{h^4}{30}f^{(5)}(\xi_0)$$

temos

h	$\approx f'(x_0)$	$e_{max} = \frac{h^4}{30}M_h$	e_{ext}
0.5555512746396548 0.1	0.555551274639654	0.0000056343702218	0.0000042809159008

[ ]:
```
# Definindo as funções
import numpy as np

x0,h = 1.8,0.1

def f(x):
    return np.log(x)

def d_1f(x):
    return 1/x

def d_5f(x):
    return 24/x**5
```

[ ]:
```
estimativa = (f(x0-2*h) -8*f(x0-h) + 8*f(x0+h) - f(x0+2*h))/(12*h)
print(estimativa)

e_max = ((h**4)/30)*abs(d_5f(x0-h)) # x0-h caso h > 0 e x0+h caso h<0 (execute
 ↪a célula anterior) (f monótona)
print(round(e_max,16))

e_ext = abs(d_1f(x0)-estimativa)
print(round(e_ext,16))
```

Considerando

$$f'(x_0) = \frac{1}{12h}(-25f(x_0) + 48f(x_0 + h) - 36f(x_0 + 2h) + 16f(x_0 + 3h) - 3f(x_0 + 4h)) - \frac{h^4}{5}f^{(5)}(\xi_1)$$

temos

h	$\approx f'(x_0)$	$e_{max} = \frac{h^4}{5}M_h$	e_{ext}
0.1	0.5555390401176418	0.0000338062213307	0.0000165154379138

[ ]:
```
# Definindo as funções
import numpy as np

x0,h = 1.8,0.1

def f(x):
    return np.log(x)

def d_1f(x):
    return 1/x

def d_5f(x):
    return 24/x**5
```

[ ]:
```
estimativa = (-25*f(x0) + 48*f(x0+h) - 36*f(x0+2*h) + 16*f(x0+3*h)
 ↪-3*f(x0+4*h))/(12*h)
print(estimativa)

e_max = ((h**4)/5)*abs(d_5f(x0-h)) # x0-h caso h > 0 e x0+h caso h<0 (execute a
 ↪célula anterior) (f monótona)
print(round(e_max,16))

e_ext = abs(d_1f(x0)-estimativa)
print(round(e_ext,16))
```

0.3 Limitações da Máquina

Considere a diminuição sucessiva de h, digamos para $h = 10^{-1}, 10^{-2}, 10^{-3}, ...$ com relação à estimativa de $f'(1.8)$ em que $f(x) = ln(x)$. Os resultados estão expressos na tabela a seguir:

h	$\approx f'(x_0)$	e_{ext}
10^{-1}	0.5561281755511222	0.0005726199955666
10^{-2}	0.5555612712535352	0.0000057156979796
10^{-3}	0.5555556127114225	0.0000000571558669
10^{-4}	0.5555555561270742	0.0000000005715186
10^{-5}	0.5555555555647462	0.0000000000091906
10^{-6}	0.5555555555036839	0.0000000000518717
10^{-7}	0.5555555560032842	0.0000000004477286
⋮	⋮	⋮
10^{-15}	0.6106226635438361	0.0550671079882805
10^{-16}	0.0000000000000000	0.5555555555555556
10^{-17}	0.0000000000000000	0.5555555555555556
⋮	⋮	⋮

Note que a melhor aproximação da derivada se deu para $h = 10^{-5}$ ao passo que começou a divergir a partir deste valor.

h	$\approx f'(x_0)$	e_{ext}
⋮	⋮	⋮
10^{-4}	0.5555555561270742	0.0000000005715186
10^{-5}	0.5555555555647462	0.0000000000091906
10^{-6}	0.5555555555036839	0.0000000000518717
⋮	⋮	⋮

A razão de a diminuição de h não necessariamente produzir uma aproximação de $f'(x_0)$ cada vez melhor pode ser explicada pela necessidade de dividir por uma potência de h. A divisão por números pequenos tende a piorar os erros de arredondamento e essa opação deve ser evitada se possível. No caso da derivada numérica, é impossível evitar tal problema inteiramente, embora métodos de ordem superior reduzem esse impasse. Teste outros exemplos e outras das formas de aproximação apresentadas.

[ ]:
```python
# Definindo as funções
import numpy as np

def f(x):
    return np.log(x)

def d_1f(x):
    return 1/x
```

[ ]:
```python
x0,h = 1.8,0.1
for k in range(1,20):
    estimativa = (f(x0+h) - f(x0-h))/(2*h)
    e_ext = round(abs(d_1f(x0)-estimativa),16)
    print(f"h = {h}, f'(x0) = {estimativa},e_ext = {e_ext}")
    h = h/10
```