Analyse Numérique
Dérivée Numérique

Simone Deparis

May 17, 2023

Contents

1 Dérivée numérique

Soit $f : [a, b] \to \mathbf{R}$, de classe C^1 et $x_0 \in [a, b]$. La dérivée $f'(x_o)$ est donnée par

$$\begin{aligned} f'(x_0) &= \lim_{h\to 0^+} \frac{f(x_0+h)-f(x_0)}{h}, \\ &= \lim_{h\to 0^+} \frac{f(x_0)-f(x_0-h)}{h}, \\ &= \lim_{h\to 0} \frac{f(x_0+h)-f(x_0-h)}{2h}. \end{aligned}$$

Soient $x_0 \in [a, b]$, (Dy) une approximation de $f'(x_0)$ et (D^2y) une approximation de $f''(x_0)$.

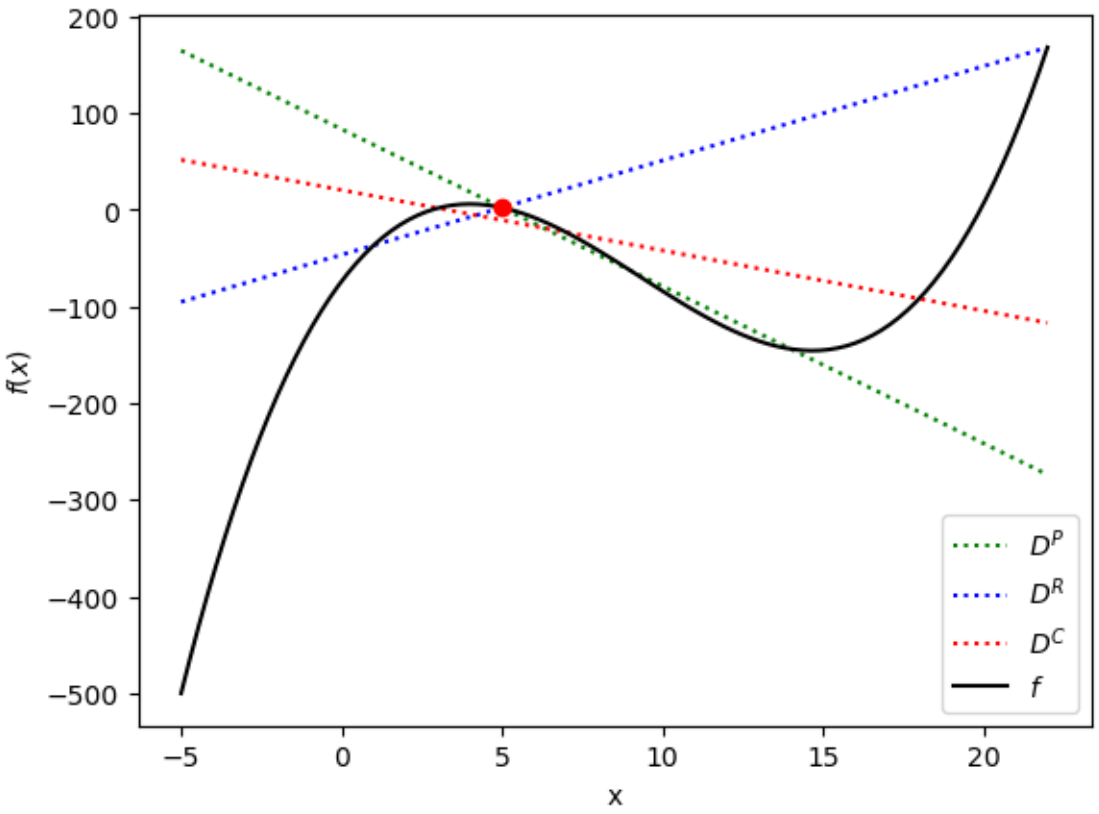
On appelle

- **différence finie progressive** l'approximation
$$(Dy)^P = \frac{f(x_0+h)-f(x_0)}{h}$$
- **différence finie rétrograde** l'approximation
$$(Dy)^R = \frac{f(x_0)-f(x_0-h)}{h}$$
- **différence finie centrée** l'approximation
$$(Dy)^C = \frac{f(x_0+\frac{1}{2}h)-f(x_0-\frac{1}{2}h)}{h}$$
- **différence finie centrée d'ordre 2** l'approximation
$$(D^2y)^C = \frac{f(x_0+h)-2f(x_0)+f(x_0-h)}{h^2}$$

[1]:
```
# importing libraries used in this book
import numpy as np
import matplotlib.pyplot as plt
```

Les différences finies progressive, rétrograde et centrée approchent la dérivée de la fonction au le point x_0.

[2]:
```
# Define a function and a point x0
f = lambda x : 0.25*(x-4)**3 - 4*(x-4)**2 +6
x0 = 5

# Choosing finite difference size
h = 4



# Defininig first order finite differences
DP = ( f(x0 + h) - f(x0) ) / h
DR = ( f(x0) - f(x0-h) ) / h
DC = ( f(x0 + h/2) - f(x0-h/2) ) / h

# defining the straight lines with slope equal to the FDs
dfp = lambda x : f(x0) + DP*(x-x0)
dfr = lambda x : f(x0) + DR*(x-x0)
dfc = lambda x : f(x0-h/2) + DC*(x-x0+h/2)

# Drowing points
z = np.linspace(-5, 22, 100)

plt.plot(z, dfp(z), 'g:', z, dfr(z), 'b:', z, dfc(z), 'r:' )
plt.plot(z, f(z), 'k', x0,f(x0),'ro')

plt.xlabel('x'); plt.ylabel('$f(x)$');
plt.legend(['$D^P$', '$D^R$', '$D^C$', '$f$'])
plt.show()
```



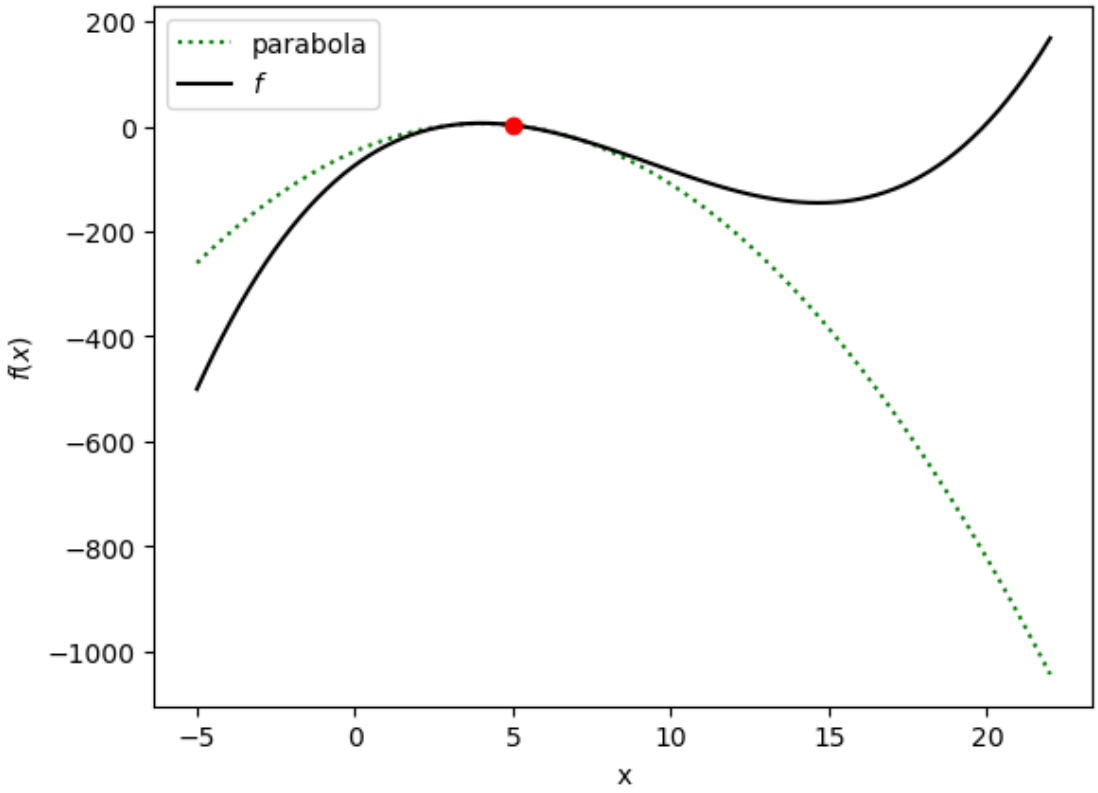
La différence finie d'ordre 2 approche la deuxième dérivée de la fonction au le point x_0.

Pour le voir graphiquement, on peut dessiner une fonction quadratique qui a la forme

$$p_2(x) = f(x_0) + (Dy)^C\,(x - x_0) + \frac{1}{2}(D^2 y)\,(x - x_0)^2$$

[3]:
```
# Define a function and a point x0
f = lambda x : 0.25*(x-4)**3 - 4*(x-4)**2 +6
x0 = 5

# Choosing finite difference size
h = 4


# Defininig first order centered finite differences
DC = ( f(x0 + h/2) - f(x0-h/2) ) / h

# Defininig second order centered finite differences
D2 = ( f(x0 + h) - 2* f(x0) + f(x0-h) ) / (h**2)
```

```
# defining the parabola going through (x_0, y_0)
parabola = lambda x : f(x0) + DC*(x-x0) + 0.5 * D2 * (x-x0)**2

# Drowing points
z = np.linspace(-5, 22, 100)

plt.plot(z, parabola(z), 'g:' )
plt.plot(z, f(z), 'k', x0,f(x0),'ro')

plt.xlabel('x'); plt.ylabel('$f(x)$');
plt.legend(['parabola', '$f$'])
plt.show()
```



1.1 Exercice

Il s'agit de vérifier numériquement que

$$\left|f'(x_0) - (Dy)^P\right| = O(h^1).$$

On pose $x_0 = 1$, $f(x) = \sin(x)$ $\forall x \in \mathbb{R}$

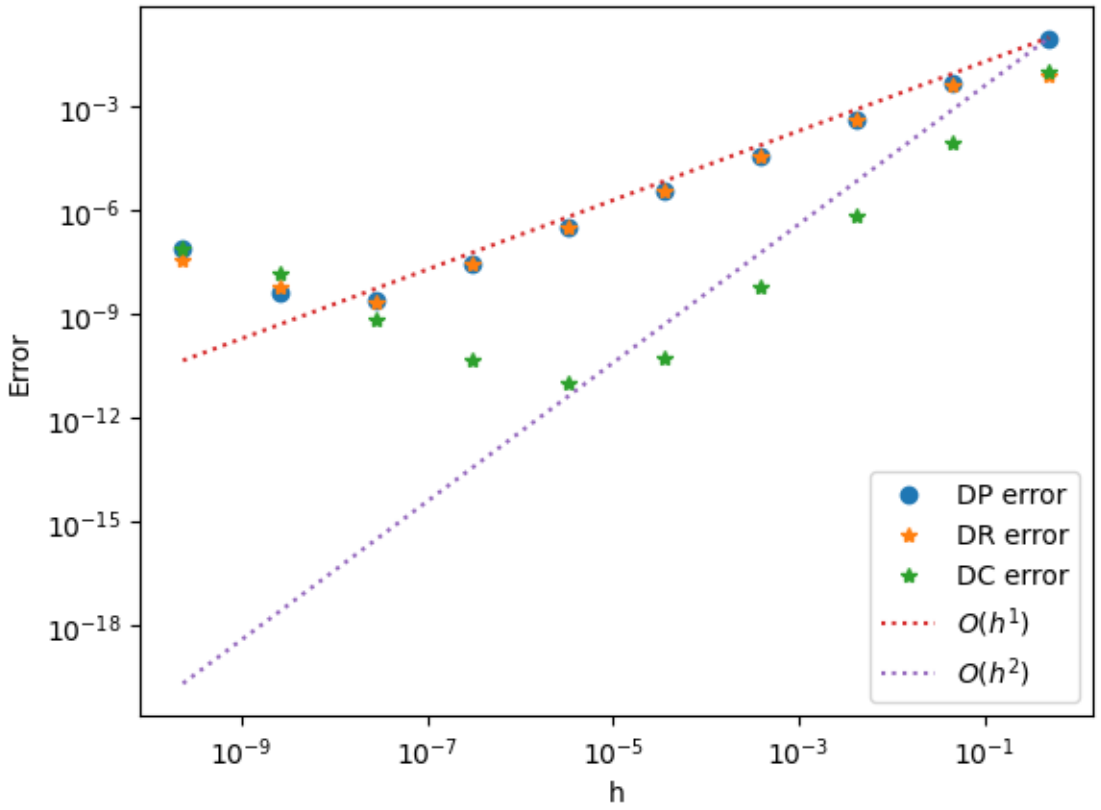
Calculez l'erreur commise en utilisant la différence finie progressive.

Quelle est la valeur de l'erreur pour h=0.1?

Ensuite vérifiez numériquement l'ordre pour $(Dy)^R$ et $(Dy)^C$.

[4]:
```
def DPerror(f,df, x0,h) :
    # formule de differences finies
    DP = ( f(x0 + h) - f(x0) ) / h
    err = abs(DP-df(x0));
    return err

def DRerror(f,df, x0,h) :
    # formule de differences finies
    DR = ( f(x0) - f(x0-h) ) / h
    err = abs(DR-df(x0));
    return err

def DCerror(f,df, x0,h) :
    # formule de differences finies
    DC = ( f(x0 + h/2) - f(x0-h/2) ) / h
    # Similarly, it is possible to define
    # DC = ( f(x0 + h) - f(x0-h) ) / (2*h)
    err = abs(DC-df(x0));
    return err


# This function just provides a line to compare the convergence in a log-log␣
 ↪plot.
from FiniteDifferenceLib import sampleConvergence
# sampleConvergence(h,order,ref) :
#   computes a pseudo order of convergence on h
#   ref is a reference maximum value


x0 = 0.2

h = 2**np.linspace(-32,-1,10)
plt.plot(h, DPerror(np.sin, np.cos, x0, h) , 'o' )
plt.plot(h, DRerror(np.sin, np.cos, x0, h) , '*' )
plt.plot(h, DCerror(np.sin, np.cos, x0, h) , '*' )


plt.plot(h, sampleConvergence(h, 1, 1e-1) , ':',
        h, sampleConvergence(h, 2, 1e-1) , ':')

plt.xlabel('h'); plt.ylabel('Error');
plt.legend(['DP error','DR error', 'DC error', '$O(h^1)$', '$O(h^2)$'])
```

```
plt.xscale('log')
plt.yscale('log')
plt.show()
```



1.2 Exercice

Il s'agit de vérifier numériquement que

$$\left|f''(x_0) - (D^2 y)\right| = O(h^2).$$

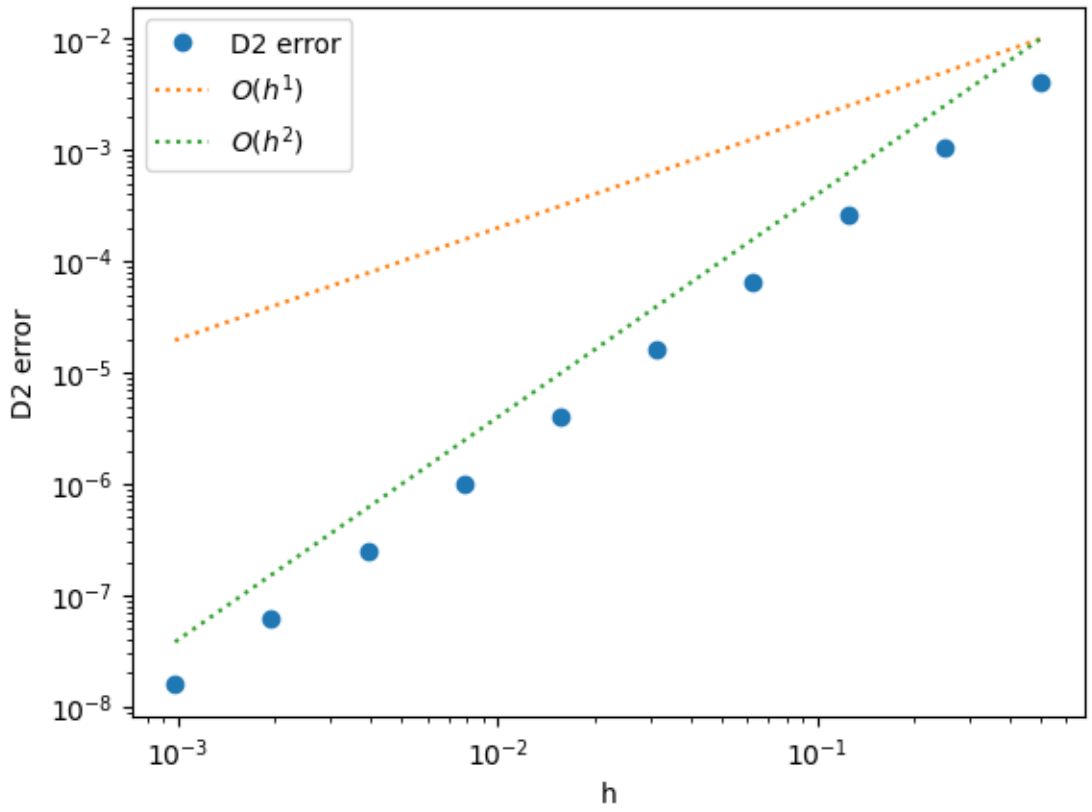
On pose $x_0 = 1$, $f(x) = \sin(x)\ \forall x \in \mathbb{R}$

Calculez l'erreur commise par ce schéma.

Que vaut l'erreur pour h=0.1?

[5]:
```
def D2error(f,df, x0,h) :
    # formule de differences finies
    D2 = ( f(x0 + h) - 2* f(x0) + f(x0-h) ) / (h**2)
    err = abs(D2-df(x0));
    # print(' x0 %e h %e erreur %e \n',x0,h,err)
```

```
    return err

x0 = 0.2

df = lambda x : -np.sin(x)

h = 2**np.linspace(-10,-1,10)
plt.plot(h, D2error(np.sin, df, x0, h) , 'o' )
plt.plot(h, sampleConvergence(h, 1, 1e-2) , ':',
         h, sampleConvergence(h, 2, 1e-2) , ':')

plt.xlabel('h'); plt.ylabel('D2 error');
plt.legend(['D2 error', '$O(h^1)$', '$O(h^2)$'])
plt.xscale('log')
plt.yscale('log')
plt.show()
```



1.3 Exercice

Il s'agit de vérifier numériquement que

$$\left| f'(x_0) - \frac{3f(x_0) - 4f(x_0 - h) + f(x_0 - 2h)}{2h} \right| = O(h^2).$$

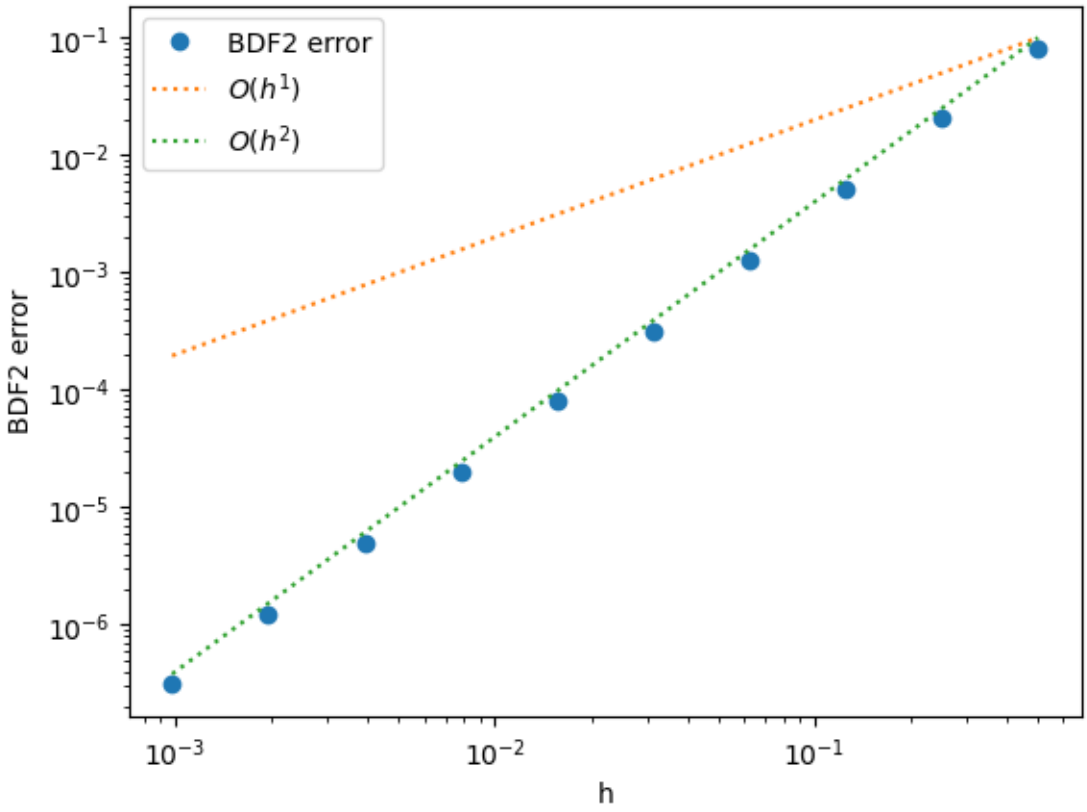
Cette formule de différences finies est à l'origine du schéma "BDF2'' pour résoudre numériquement des équations différentielles (Chapitre 9 du livre).

On pose $x_0 = 1$, $f(x) = \sin(x)$ $\forall x \in \mathbb{R}$

Calculez l'erreur commise par ce schéma.

Que vaut l'erreur pour h=0.1?

[6]:
```
def bdf2error(f,df, x0,h) :
    # formule de differences finies BDF2
    diff = (3*f(x0)-4*f(x0-h)+f(x0-2*h))/(2*h);
    err = abs(diff-df(x0));
    # print(' x0 %e h %e erreur %e \n',x0,h,err)

    return err

x0 = 0.2

h = 2**np.linspace(-10,-1,10)
plt.plot(h, bdf2error(np.sin, np.cos, x0, h) , 'o' )
plt.plot(h, sampleConvergence(h, 1, 1e-1) , ':',
        h, sampleConvergence(h, 2, 1e-1) , ':')

plt.xlabel('h'); plt.ylabel('BDF2 error');
plt.legend(['BDF2 error', '$O(h^1)$', '$O(h^2)$'])
plt.xscale('log')
plt.yscale('log')
plt.show()
```


BDF2 error
$O(h^1)$
$O(h^2)$
BDF2 error
h
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-3}
10^{-2}
10^{-1}