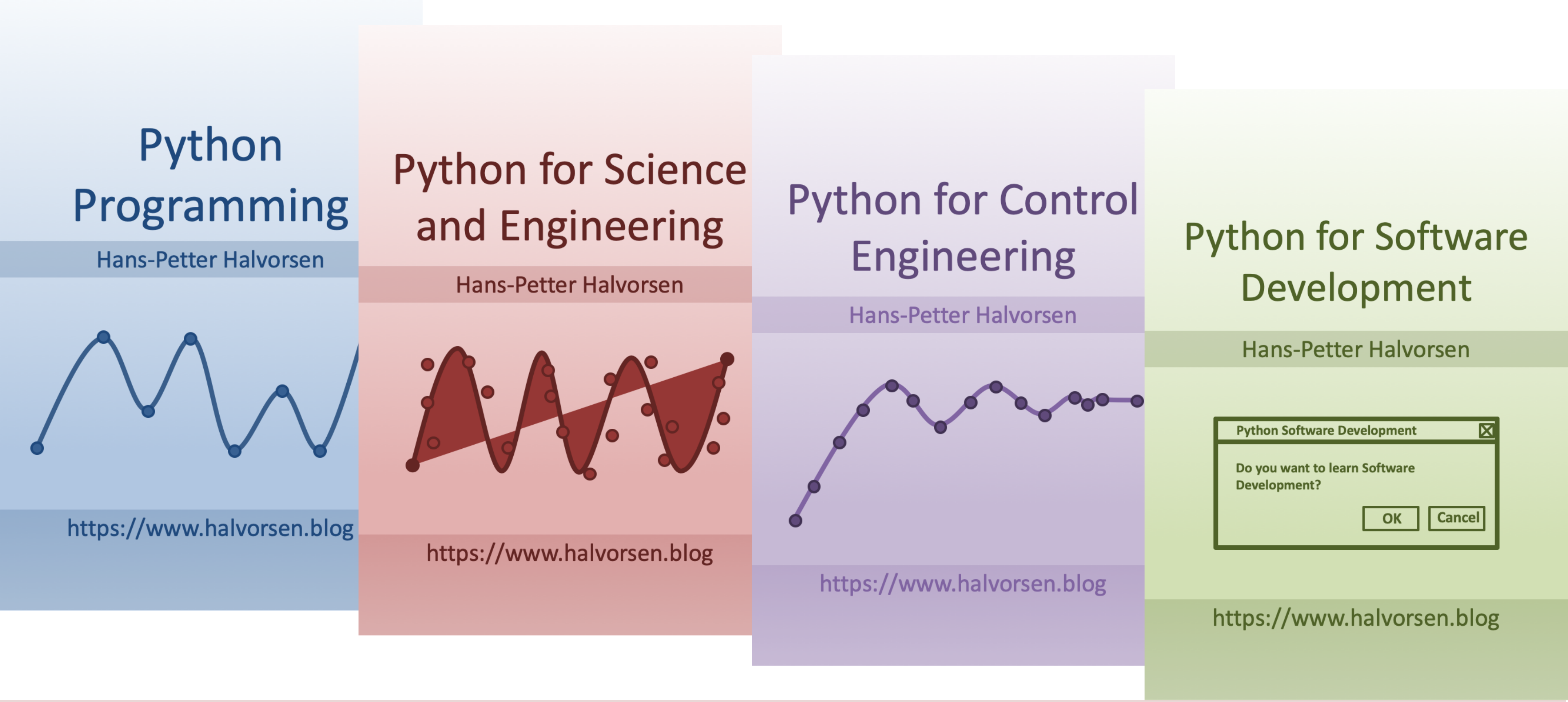
Numerical Differentiation in Python

Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

https://www.halvorsen.blog/documents/programming/python/

Additional Python Resources



https://www.halvorsen.blog/documents/programming/python/

Contents

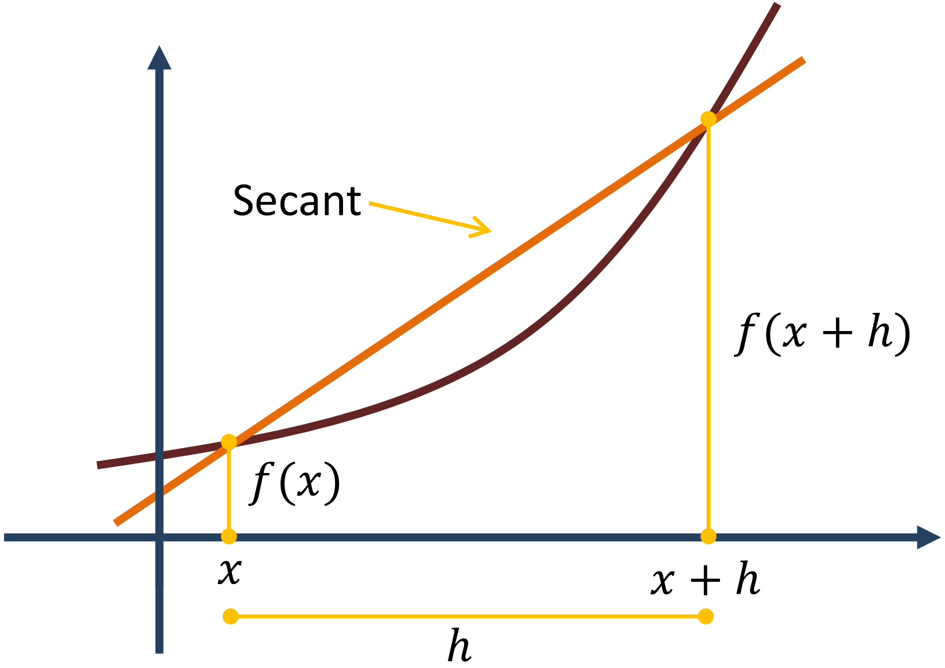
- The Derivative
- Numerical Differentiation
- Python Examples

It is assumed that already know about the derivative from mathematics courses and that you want to use Python to find numerical solutions

The Derivative

The derivative of a function $y = f(x)$ is a measure of how y changes with x

The derivative of a function $f(x)$ is denoted $\frac{df(x)}{dx}$



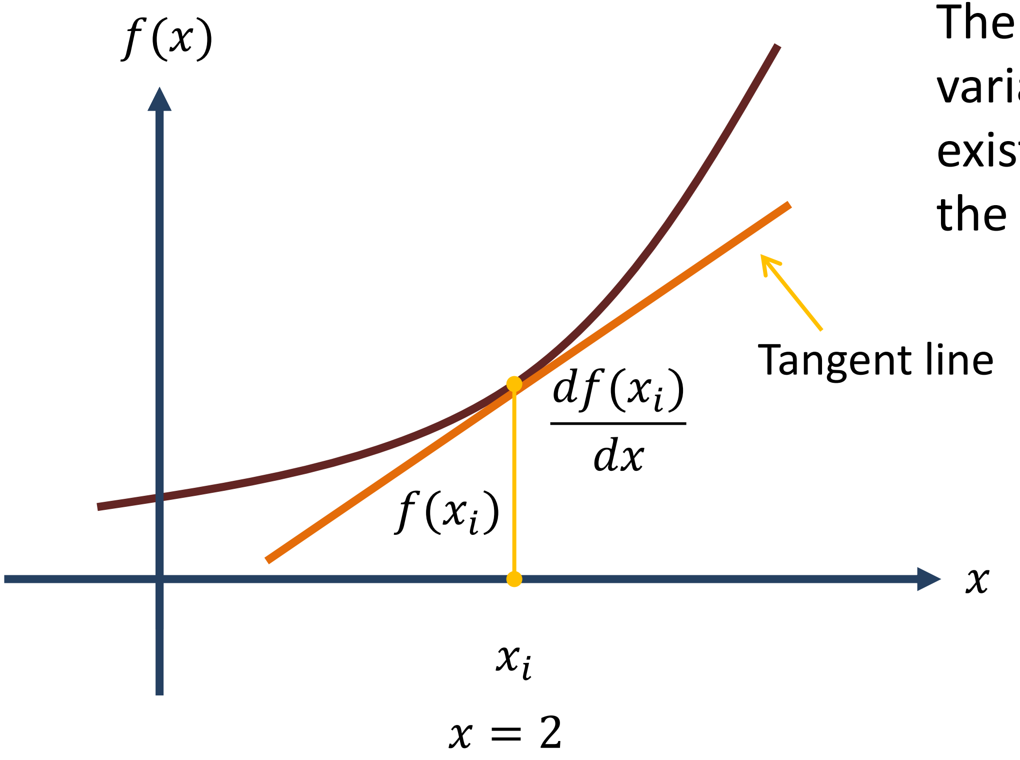
We have the following definition:

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

Different notation is used:

$$\frac{df(x)}{dx} = y'(x) = \dot{y}(x)$$

The Derivative



The derivative of a function of a single variable at a chosen input value, when it exists, is the slope of the tangent line to the graph of the function at that point.

Example:

$$f(x) = x^2$$

$$\frac{df(x)}{dx} = 2x$$

$$\frac{df(2)}{dx} = 2 \times 2 = 4$$

https://en.wikipedia.org/wiki/Derivative

Derivative Rules

There are many derivative rules (as you probably know from mathematics courses)

We will focus on the the basic rule:

$$f(x) = kx^n \quad \Rightarrow \quad \frac{df(x)}{dx} = k \cdot n \cdot x^{n-1}$$

Example:

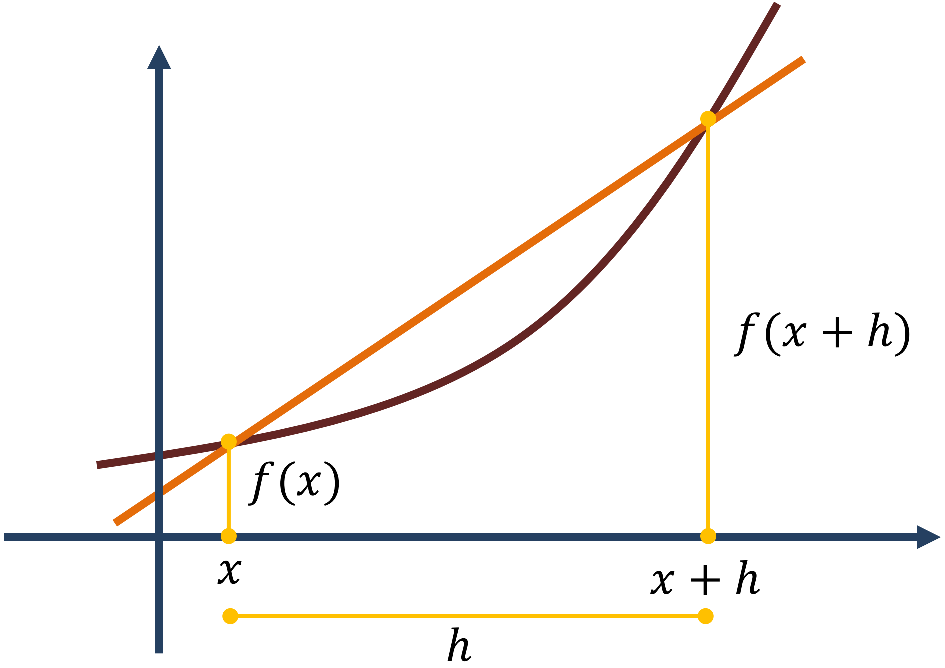
$$f(x) = 4x^3$$

$$\frac{df(x)}{dx} = 4 \times 3x^2 = 12x^2$$

$$\frac{df(3)}{dx} = 12 \times 3^2 = 12 \times 9 = 108$$

Basic Numerical Approach

A numerical approach to the derivative of a function $y = f(x)$ is:



$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

Note! We will use Python in order to find the <u>numeric</u> solution – not the analytic solution

Example

$y(x) = x^2 \Rightarrow \frac{dy}{dx} = ?$

We know for this simple example that the exact analytical solution is: $\frac{dy}{dx} = 2x$

Given the following values:

x	y
−2	4
−1	1
0	0
1	1
2	4

$$\frac{dy}{dx}(x = -2) = -4$$
$$\frac{dy}{dx}(x = -1) = -2$$
$$\frac{dy}{dx}(x = 0) = 0$$
$$\frac{dy}{dx}(x = 1) = 2$$
$$\frac{dy}{dx}(x = 2) = 4$$

Let's use Python to see if we get the same values?

Python Code

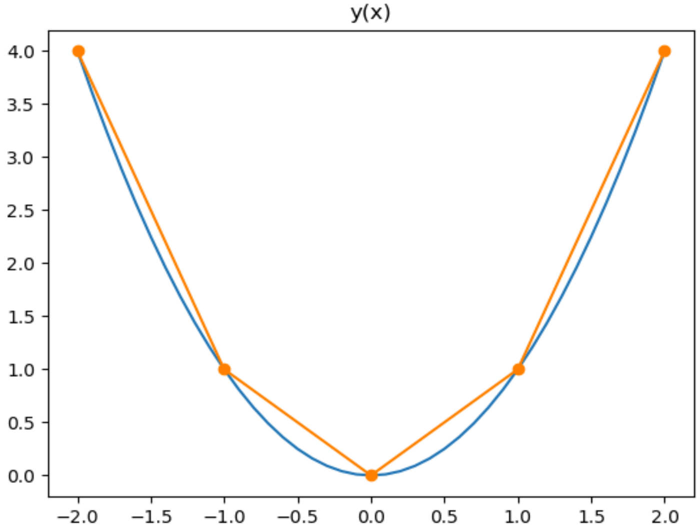
We start to plot the function:

$$y(x) = x^2$$

We use the following data points:

x	y
-2	4
-1	1
0	0
1	1
2	4

Resulting plot:




```
import numpy as np
import matplotlib.pyplot as plt


xstart = -2
xstop = 2.1
increment = 0.1
x = np.arange(xstart,xstop,increment)

y = x**2

plt.plot(x,y)


xstart = -2
xstop = 3
increment = 1
x = np.arange(xstart,xstop,increment)

y = x**2;

plt.plot(x,y, '-o')
plt.title("y(x)")
```

Python Code

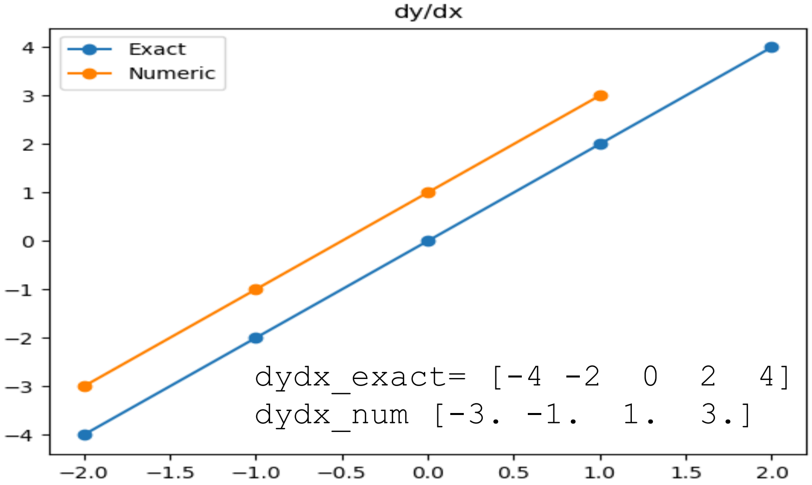
We will use numerical differentiation to find $\frac{dy}{dx}$ for the following function:

$$y(x) = x^2$$

We use the following data points:

x	dy/dx
-2	
-1	
0	
1	
2	

Results:




```
import numpy as np
import matplotlib.pyplot as plt

xstart = -2
xstop = 3
increment = 1
x = np.arange(xstart,xstop,increment)

y = x**2;

# Exact/Analytical Solution
dydx_exact = 2*x

print("dydx_exact=", dydx_exact)

plt.plot(x, dydx_exact, 'o-')

# Numerical Solution
dydx_num = np.diff(y) / np.diff(x);

print("dydx_num", dydx_num)

xstart = -2
xstop = 2

x = np.arange(xstart,xstop,increment)

plt.plot(x, dydx_num, 'o-')
plt.title("dy/dx")
plt.legend(["Exact", "Numeric"])
```

Comments to the Results

$$y(x) = x^2 \Rightarrow \frac{dy}{dx} = ?$$

Exact Solution vs. Python Solution:

x	dy/dx (Exact solution)	dy/dx (Numeric solution)
-2	-4	-3
-1	-2	-1
0	0	1
1	2	3
2	4	-

Results from Python Script:

```
dydx_exact= [-4 -2  0  2  4]
dydx_num [-3. -1.  1.  3.]
```

$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

- The accuracy of the results are not so good.
- Can we expect better results when we increase number of data points?
- Let's Try!

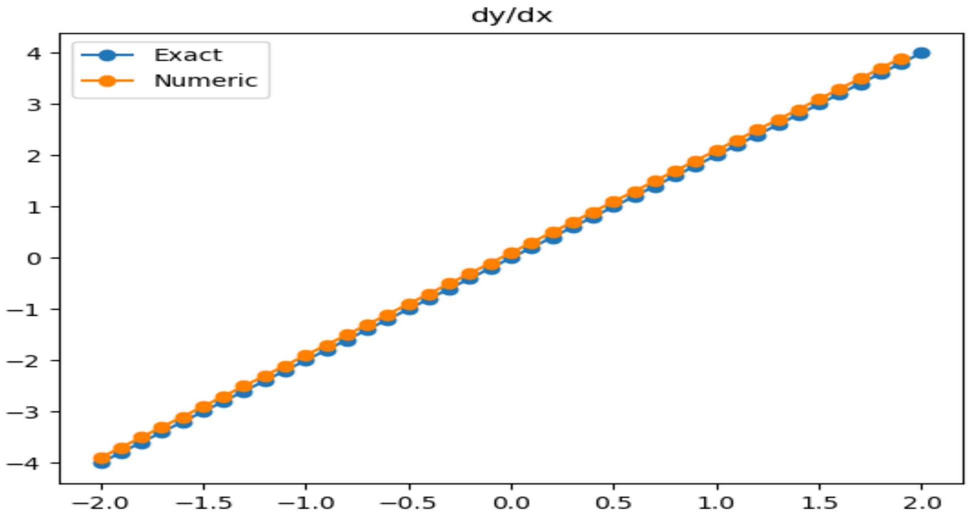
Python Code

We increase number of data points to see if we get better results

Previously: `x = -2,-1,0,1,2`

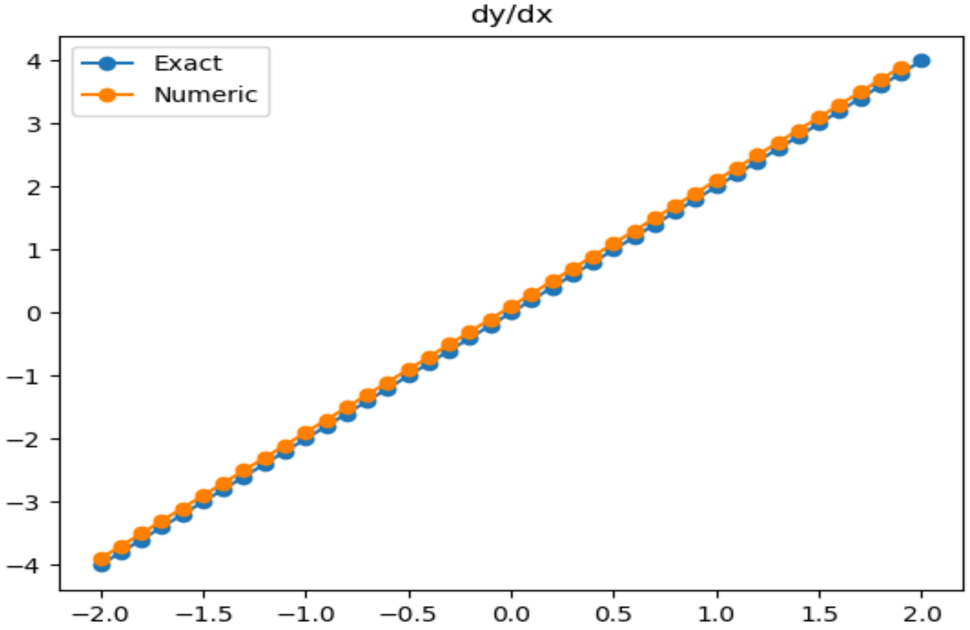
Now: `x = -2.0,-1.9, -1.8, … 0, 0.1, …, 1.9, 2.0`

Yes!




```
import numpy as np
import matplotlib.pyplot as plt

xstart = -2
xstop = 2.1
increment = 0.1

x = np.arange(xstart,xstop,increment)

y = x**2;

# Exact/Analytical Solution
dydx_exact = 2*x

print("dydx_exact=", dydx_exact)
plt.plot(x, dydx_exact, 'o-')

# Numerical Solution
dydx_num = np.diff(y) / np.diff(x);

print("dydx_num", dydx_num)

xstart = -2
xstop = 2

x2 = np.arange(xstart,xstop,increment)

plt.plot(x2, dydx_num, 'o-')
plt.title("dy/dx")
plt.legend(["Exact", "Numeric"])
```

Comments to the Results



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

$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

- We see that the numeric solutions becomes very close to the exact solutions.
- When $h \to 0$ we should expect that the numerical solutions should exactly match the exact solutions.

Differentiation on Polynomials

Hans-Petter Halvorsen

Polynomials

A polynomial is expressed as:

$$p(x) = p_1 x^n + p_2 x^{n-1} + \cdots + p_n x + \; p_{n+1}$$

where $p_1, p_2, p_3, \dots$ are the coefficients of the polynomial.

In Python we can use the `polyder()` function to perform differentiation on polynomials.
This function works the same way as the `polyint()` function which performs integration on polynomials.

Derivative Rules

There are many derivative rules (as you probably know from mathematics courses)

This basic rule is valid for Polynomials:

$$f(x) = kx^n \quad \Rightarrow \quad \frac{df(x)}{dx} = k \cdot n \cdot x^{n-1}$$

Example:

$$f(x) = 4x^3$$

$$\frac{df(x)}{dx} = 4 \times 3x^2 = 12x^2$$

$$\frac{df(3)}{dx} = 12 \times 3^2 = 12 \times 9 = 108$$

Example

Given the following polynomial:

$$p(x) = x^3 + 2x^2 - x + 3$$

Note!!! In order to use it in Python, we reformulate:

$$p(x) = 3 - x + 2x^2 + x^3$$

We find:

$$\frac{dp(x)}{dx} = 0 - 1 + 4x + 3x^2 = -1 + 4x + 3x^2$$

Python

$$p(x) = 3 - x + 2x^2 + x^3$$

```
import numpy.polynomial.polynomial as poly

p = [3, -1, 2, 1]

dpdx = poly.polyder(p)

print("dpdx =", dpdx)
```

$$\frac{dp(x)}{dx} = -1 + 4x + 3x^2 \longrightarrow$$

The Python solution:

```
dpdx = [-1.  4.  3.]
```

We see that the Python script gives the correct answer!

Another Python Example

```
import numpy.polynomial.polynomial as poly

p = [2, 0, 0, 1]

dpdx = np.polyder(p)

print("dpdx =", dpdx)
```

Given the Polynomial:

$$p(x) = 2 + x^3$$

We need to reformulate to make it work with Python:

$$p(x) = \mathbf{2} + \mathbf{0} \cdot x + \mathbf{0} \cdot x^2 + \mathbf{1} \cdot x^3$$

We find the derivative:

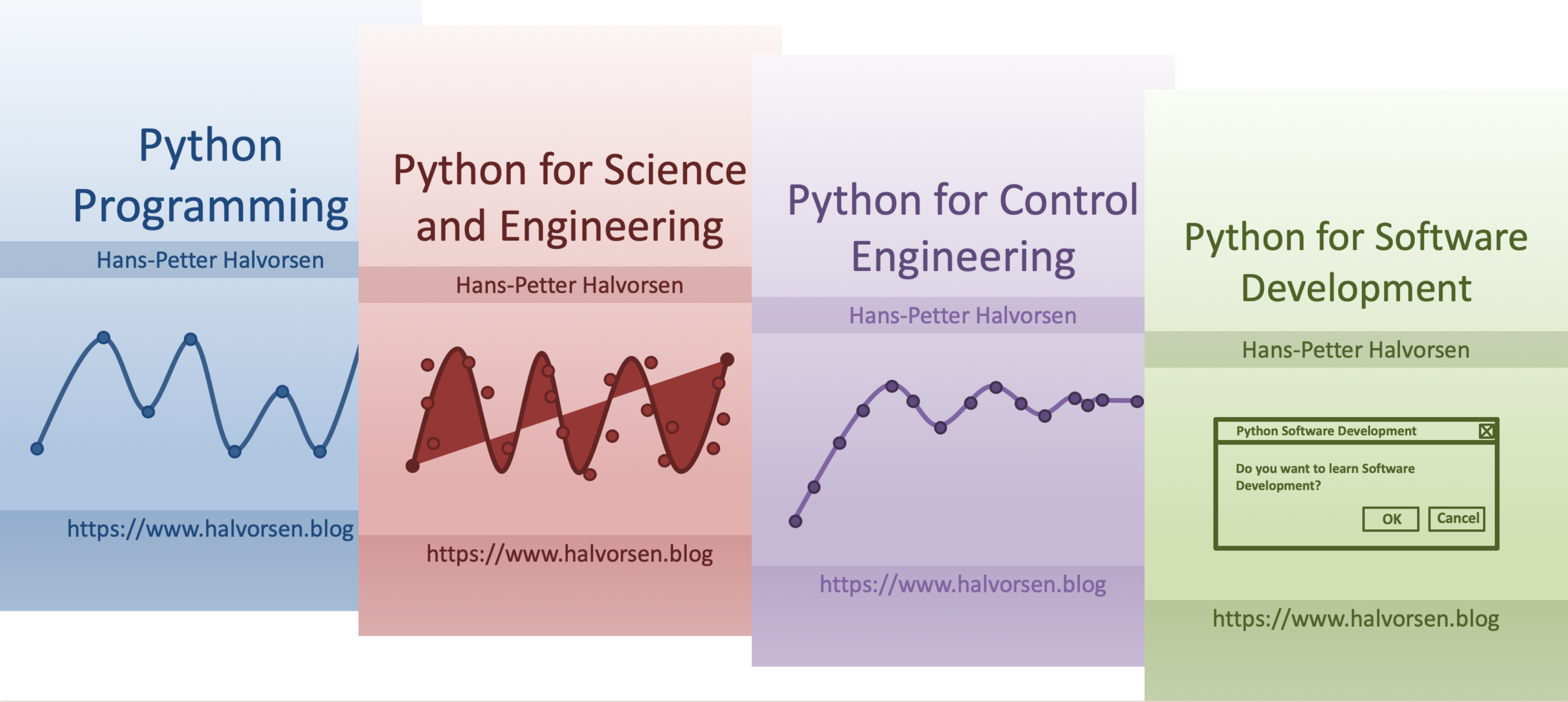
$$\frac{dp(x)}{dx} = \mathbf{0} \cdot x + \mathbf{0} \cdot 2x + \mathbf{3}x^2 = 3x^2 \longrightarrow$$

The Python solution:

```
dpdx = [0. 0. 3.]
```

We see that the Python script gives the correct answer!

Additional Python Resources



https://www.halvorsen.blog/documents/programming/python/

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog