https://www.halvorsen.blog

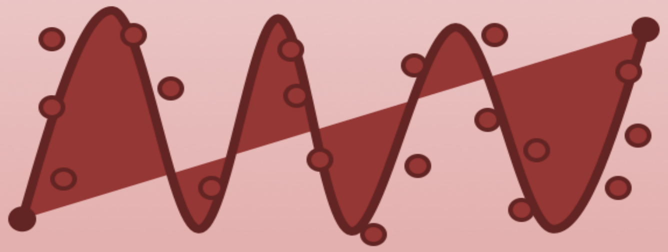
Optimization in Python

Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

Python for Science and Engineering

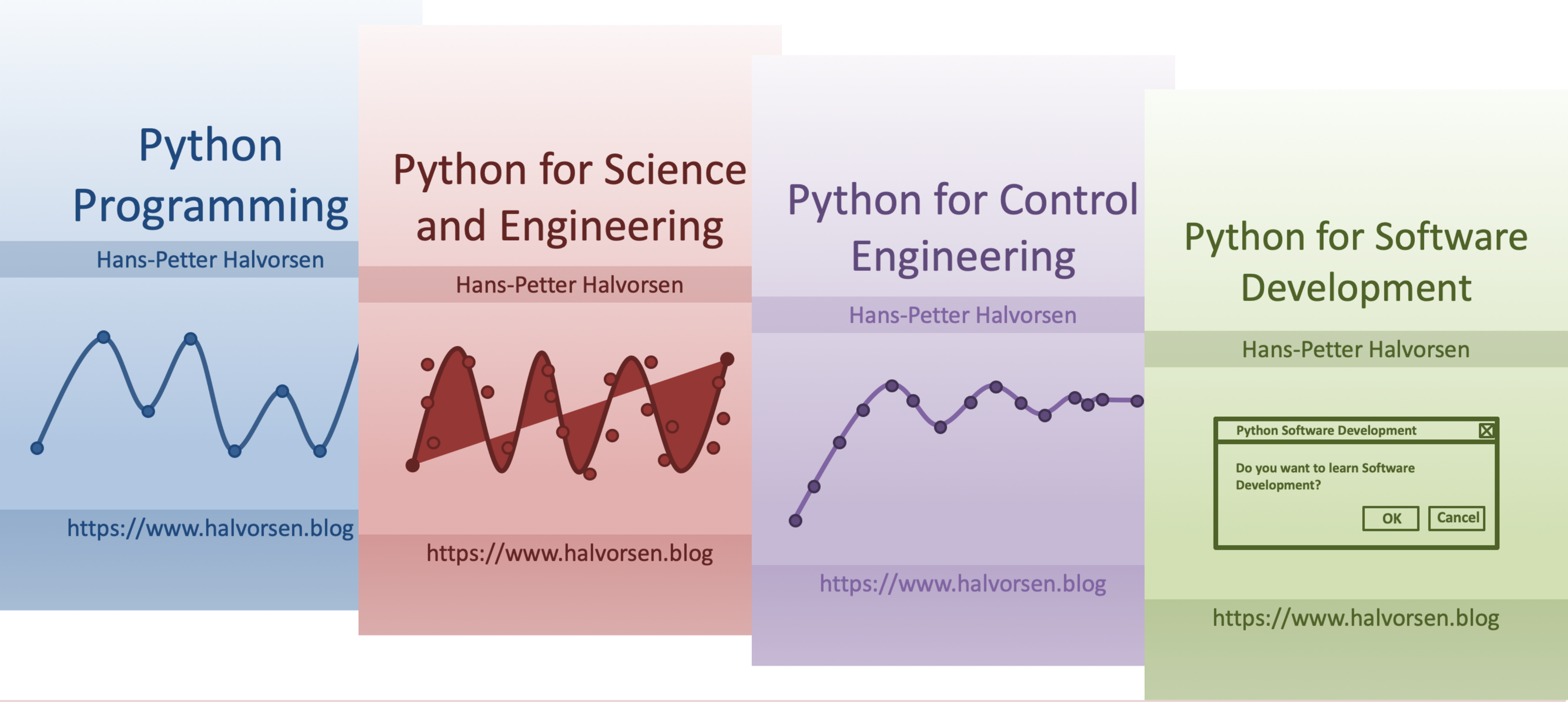
Hans-Petter Halvorsen



https://www.halvorsen.blog

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

Additional Python Resources



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

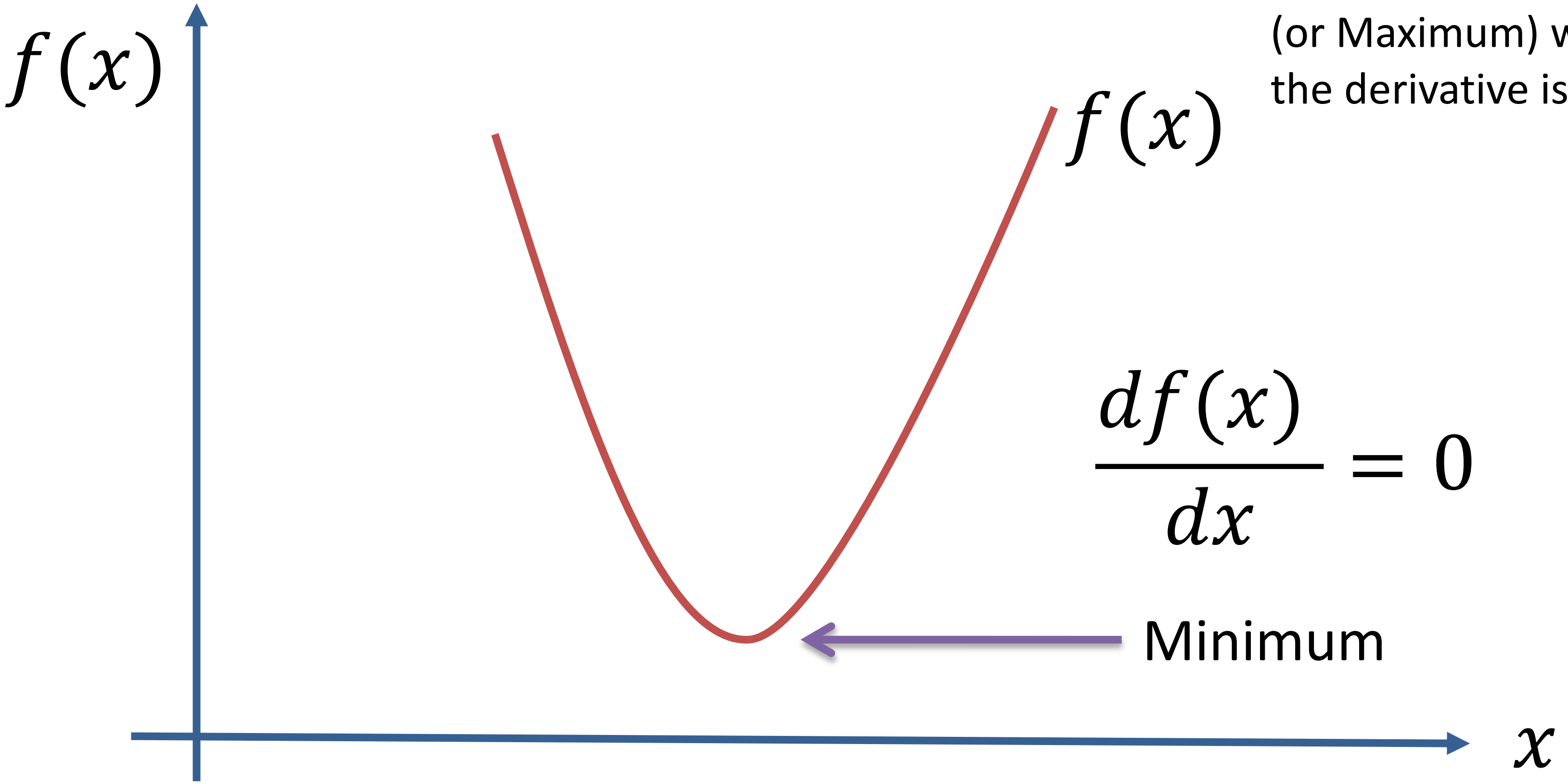
Contents

- Optimization
  - Find minimum (or maximum) of a given function
  - Curve Fitting, where you find an “optimal“ Model based on a given Data Set, i.e., You find the model parameters for a selected model that best fits the data set
- The **SciPy** Library
- Lots of Python Examples

Optimization

Optimization is based on finding the minimum of a given function

We find the Minimum (or Maximum) where the derivative is zero



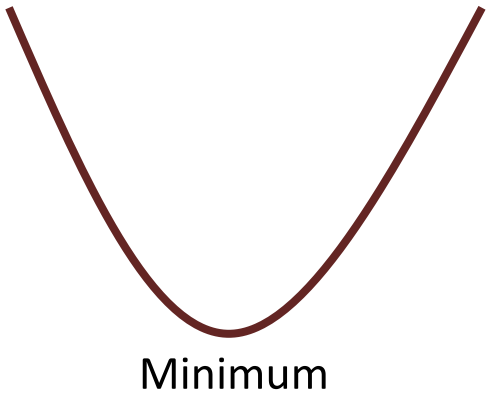
$$\frac{df(x)}{dx} = 0$$

Optimization

- Optimization is important in mathematics, control and simulation applications
- Basically it is all about finding minimum (or maximum) of a given function
- E.g., in Model Predictive Control (MPC) you use optimization to find the optimal control signal based on some criteria and constraints

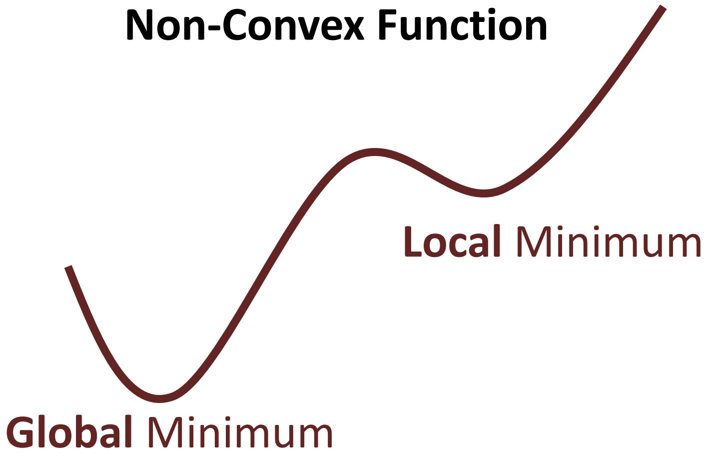
Optimization Challenges

Convex Function



Optimizing convex functions is easy

Non-Convex Function



Optimizing non-convex functions can be much more complicated

When you have more than one variable (Multiple variables) it also become more complex

https://scipy-lectures.org/advanced/mathematical_optimization/

Optimization - Example

The cost function often used in MPC is like this:

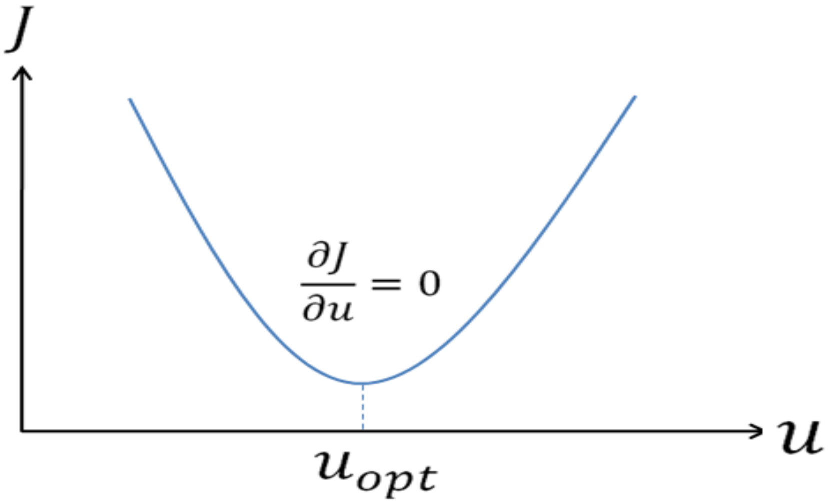
$$J = \sum_{k=0}^{N_p} (\hat{y} - r)^T Q \, (\hat{y} - r) + \sum_{k=0}^{N_c} \Delta u^T R \, \Delta u$$

Where u is the Control Signal

So the basic challenge is to solve:

$$\frac{\partial J}{\partial u} = 0$$

By solving this we get the future optimal control (u_{opt})



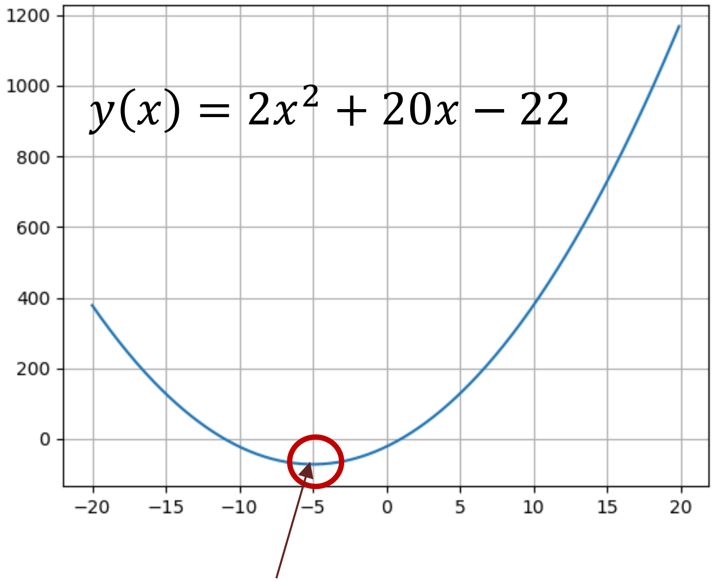
The optimal control signal used by the MPC controller

In this Tutorial/Video we will only go through some general Optimization problems and not focus on MPC or other specific applications

Optimization

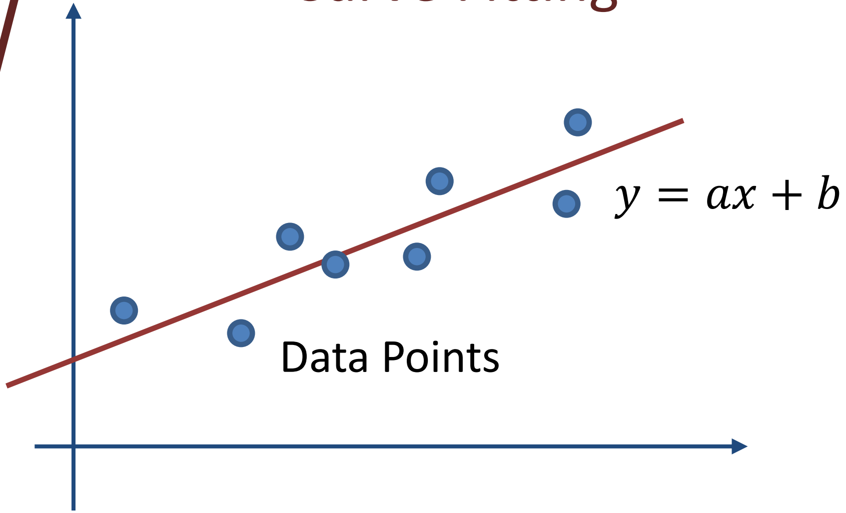
In this video we will go through 2 types of Optimization problems

Find Minimum of a given Function



Minimum

Curve Fitting



Find an "Optimal" Model based on a given Data Set

Example – Find Minimum

Example: We want to find for what value of x the function has its minimum value

$$y(x) = 2x^2 + 20x - 22$$

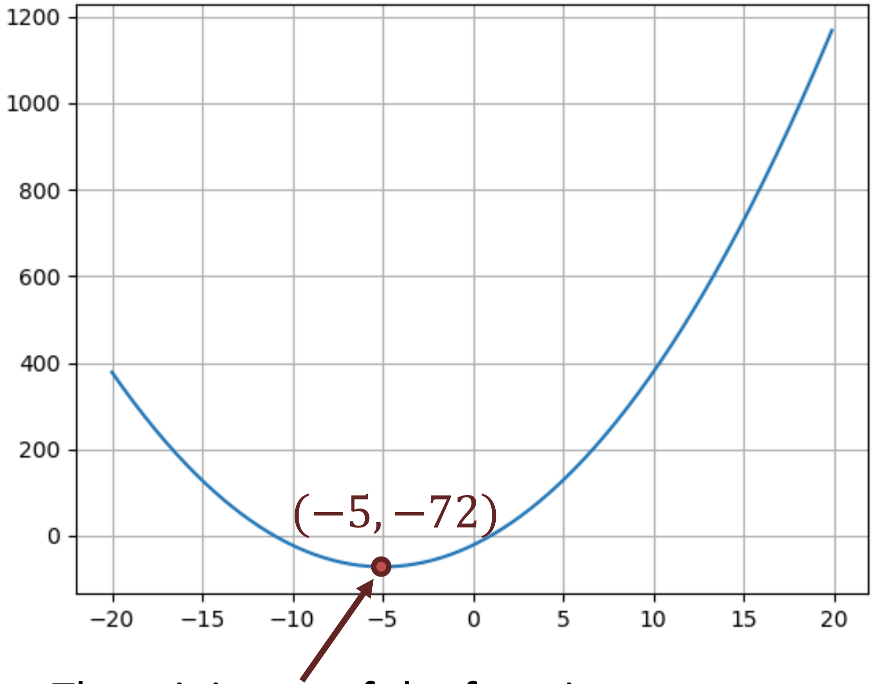
We can of course find the derivative of the function and find where the derivative is equal to zero:

$$\frac{dy}{dx} = 4x + 20 = 0$$

This gives:

$$x_{min} = -5$$

$$y(-5) = 50 - 100 - 22 = -72$$



The minimum of the function

“Simple“ Solution

Example: We want to find for what value of x the function has its minimum value

$$y(x) = 2x^2 + 20x - 22$$

We use Python to iterate through all values of $y(x)$ using a While Loop. Inside the While Loop we compare $y(i)$ and $y(i+1)$. If $y(i+1)$ is larger than $y(i)$ we have found the minimum.

The Python results becomes the same as the analytical solution:

$(-5, -72)$ ←

Python Solution:

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

xstart = -20
xstop = 20
increment = 0.1
x = np.arange(xstart,xstop,increment)
y = 2 * x*x + 20 * x - 22

plt.plot(x,y)
plt.grid()

i = 0

while y[i] > y[i+1]:
      i = i+1

print(x[i])
print(y[i])
```

Optimization with SciPy

Hans-Petter Halvorsen

SciPy

- SciPy is a free and open-source Python library used for scientific computing and engineering
- SciPy contains modules for optimization, linear algebra, interpolation, image processing, ODE solvers, etc.

SciPy

- The optimize Module in the SciPy Library provides functions for minimizing (or maximizing) objective functions
- Functions:
  - **`fminbound()`**, `fmin()`, `minimize_scalar()`, `minimize()`

https://docs.scipy.org/doc/scipy/reference/optimize.html

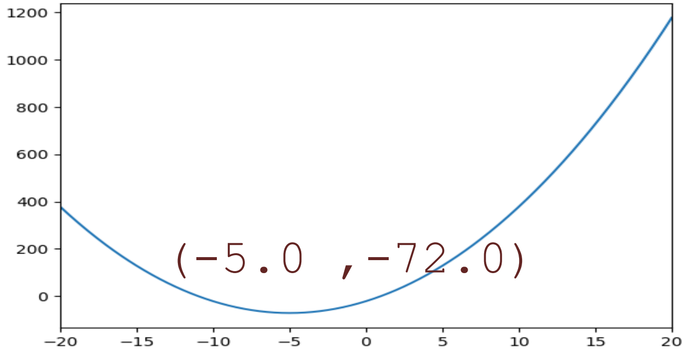
Scalar Function - Example

Given the following function:

$$y(x) = 2x^2 + 20x - 22$$

(same as in previous example)

We use the **optimize.fminbound()** function in the SciPy Library



We get the same results as previous example

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize

def func(x):
    y = 2 * x**2 + 20*x - 22
    return y

xmin = -20
xmax = 20
dx = 0.1
N = int((xmax - xmin)/dx)
x = np.linspace(xmin, xmax, N+1)

y = func(x)

plt.plot(x,y)
plt.xlim([xmin,xmax])

x_min = optimize.fminbound(func, xmin, xmax)
y_min = func(x_min)

print(x_min)
print(y_min)
```

https://www.halvorsen.blog

Multiple Variables in SciPy

Hans-Petter Halvorsen

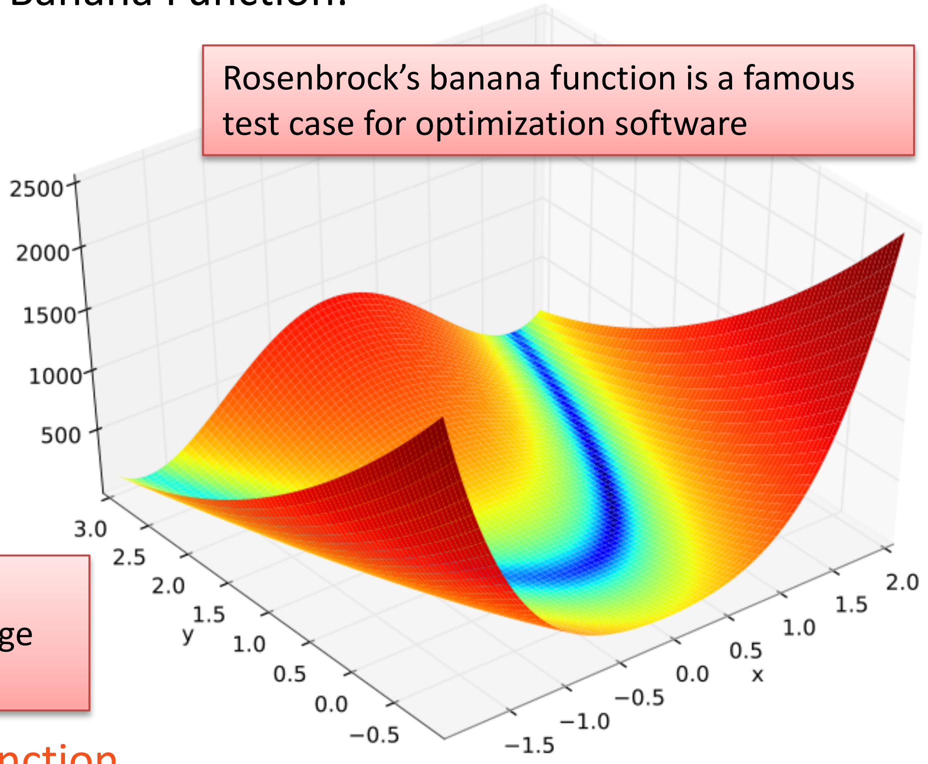
Rosenbrock's Banana Function

The function below is known as Rosenbrock's Banana Function:

$$f(x,y) = (a - x)^2 + b(y - x^2)^2$$

We will find the Minimum of this function

Rosenbrock’s banana function is a famous test case for optimization software



This function is used to verify performance and robustness of optimization algorithms since it is demanding to find the minimum for this function.

The global minimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial. To converge to the **global minimum**, however, is difficult.

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

Rosenbrock's Banana Function

$f(x, y) = (a - x)^2 + b(y - x^2)^2$

It has a global minimum at $(x, y) = (a, a^2)$, where $f(x, y) = 0$
Usually these these parameters are set such that $a = 1$ and $b = 100$. Only in the trivial case where $a = 0$ the function is symmetric, and the minimum is at the origin.

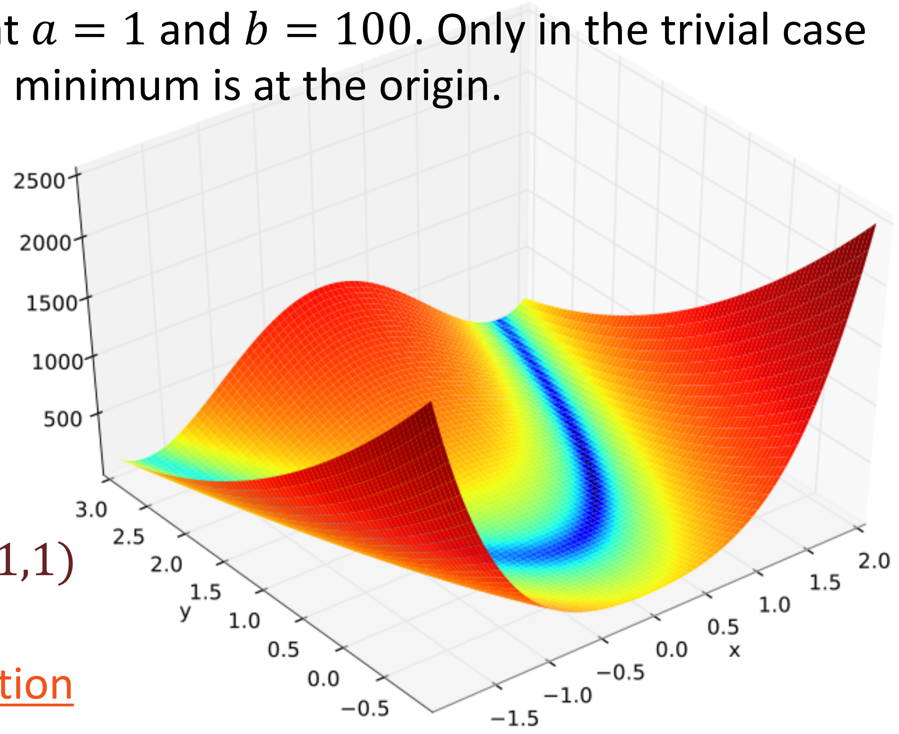
We set $a = 1$ and $b = 100$

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

We will find the Minimum of this function

It has a global minimum at $(x, y) = (a, a^2) = (1,1)$

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



Rosenbrock's Banana Function

$$f(x, y) = (a - x)^2 + b(y - x^2)^2$$

Global minimum at $(x, y) = (a, a^2)$

Setting $a = 1$ gives global minimum at $(x, y) = (1,1)$

The Python code gives the following results:

```
Optimization terminated successfully.
         Current function value: 
0.000000
         Iterations: 85
         Function evaluations: 159
[1.00002202 1.00004222]
```

```python
import scipy.optimize as opt


def banana(x):
    a = 1
    b = 100
    y = (a-x[0])**2 + b*(x[1]-x[0]**2)**2
    return y


xopt = opt.fmin(func=banana, x0=[-1.2,1])

print(xopt)
```

Note! `x[0]=x` and `x[1]=y`

Python – Alternative Code

```
import scipy.optimize as opt

def banana(var):
    a = 1
    b = 100
    x, y = var
    y = (a-x)**2 + b*(y-x**2)**2
    return y


xopt = opt.fmin(func=banana, x0=[-1.2,1])

print(xopt)
```

In previous code example we used **x[0]=x** and **x[1]=y**

The code alternative illustrated here is probably more readable

var is a NumPy array consisting 2 elements, namely x and y values in this case

You should also try with other values for a and b (especially for a, since a affects the minimum)

Using other Optimization Functions

Hans-Petter Halvorsen

SciPy – Other Functions

Banana Function Examples

Previous Example using **`fmin()`**

```
import scipy.optimize as opt

def banana(var):
    a = 1
    b = 100
    x, y = var
    y = (a-x)**2 + b*(y-x**2)**2
    return y

xopt = opt.fmin(func=banana, x0=[-1.2,1])

print(xopt)
```

New Example using **`minimize()`**

```
import scipy.optimize as opt

def banana(var):
    a = 1
    b = 100
    x, y = var
    y = (a-x)**2 + b*(y-x**2)**2
    return y


xopt = opt.minimize(banana, x0=[-1.2,1])

print(xopt)
```

Scalar Function Examples

SciPy – Other Functions

Previous Example using **fminbound()**

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize

def func(x):
    y = 2 * x**2 + 20*x - 22
    return y

xmin = -20
xmax = 20
dx = 0.1
N = int((xmax - xmin)/dx)
x = np.linspace(xmin, xmax, N+1)

y = func(x)

plt.plot(x,y)
plt.xlim([xmin,xmax])

x_min = optimize.fminbound(func, xmin, xmax)
y_min = func(x_min)

print(x_min)
print(y_min)
```

New Example using **minimize_scalar()**

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize

def func(x):
    y = 2 * x**2 + 20*x - 22
    return y

xmin = -20
xmax = 20
dx = 0.1
N = int((xmax - xmin)/dx)
x = np.linspace(xmin, xmax, N+1)

y = func(x)

plt.plot(x,y)
plt.xlim([xmin,xmax])

res = optimize.minimize_scalar(func)

print(res)
```

SciPy – Other Functions

- The **scipy.optimize** contains many different optimization functions that use different optimization methods
- You need to find and use the functions and methods that is best for your Optimization problem
- This Tutorial/Video only scratches the surface of the Optimization Topic
- For more information about Optimization in SciPy, read the documentation:

https://docs.scipy.org/doc/scipy/reference/optimize.html

https://www.halvorsen.blog

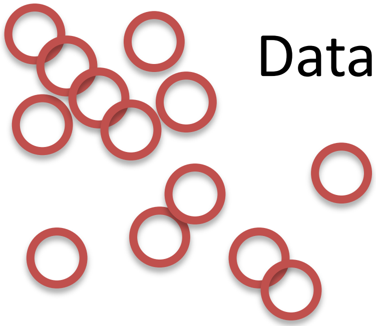
Curve Fitting

Hans-Petter Halvorsen

Curve Fitting

Curve Fitting is all about fitting data to a Mathematical Model

Mathematical Model



Data

$$y = f(x)$$

- Curve Fitting is also an Optimization problem
- You find an “optimal“ Model based on a given Data Set.
- You find the model parameters for a selected model that best fits the data set

Curve Fitting

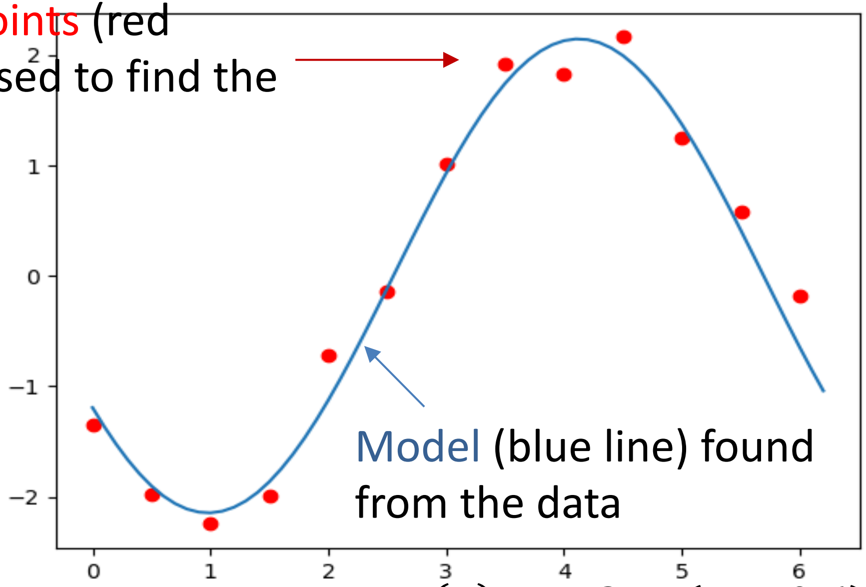
- Python has curve fitting functions that allows us to create empiric data model.
- We will show a basic example
- More about Curve Fitting in another Video/Another part of the Textbook

Example

Assume we want to fit some given data to the following model:

$$y(x) = \mathrm{a} \cdot \sin(x + b)$$

Data Points (red dots) used to find the Model



Model (blue line) found from the data

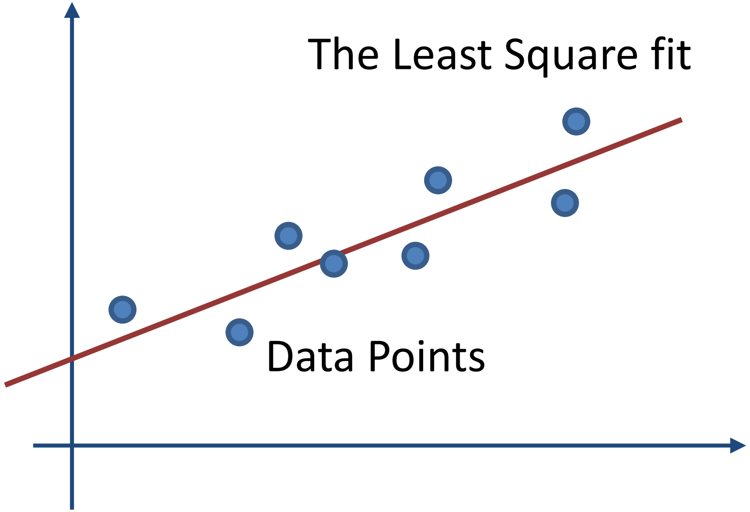
[-2.03108093   0.629067   ] $\Rightarrow$ $y(x) \approx -2\sin(x + 0.6)$

```python
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

start = 0
stop = 2*np.pi
increment = 0.5
x = np.arange(start,stop,increment)

a = 2
b = 10
np.random.seed()
y_noise = 0.2 * np.random.normal(size=x.size)
y = a * np.sin(x + b)
y = y + y_noise

plt.plot(x,y, 'or')


def model(x, a, b):
    y = a * np.sin(x + b)
    return y

popt, pcov = curve_fit(model, x, y)
print(popt)

increment = 0.1
xmodeldata = np.arange(start,stop,increment)

ymodel = model(xmodeldata, *popt)

plt.plot(xmodeldata,ymodel)
```

Least Square Method (LSM)

The least squares method requires the model to be set up in the following form based on input-output data :

$$Y = \Phi\theta$$



The Least Square Method is given by:

$$\theta_{LS} = (\Phi^T\Phi)^{-1}\Phi^T Y$$

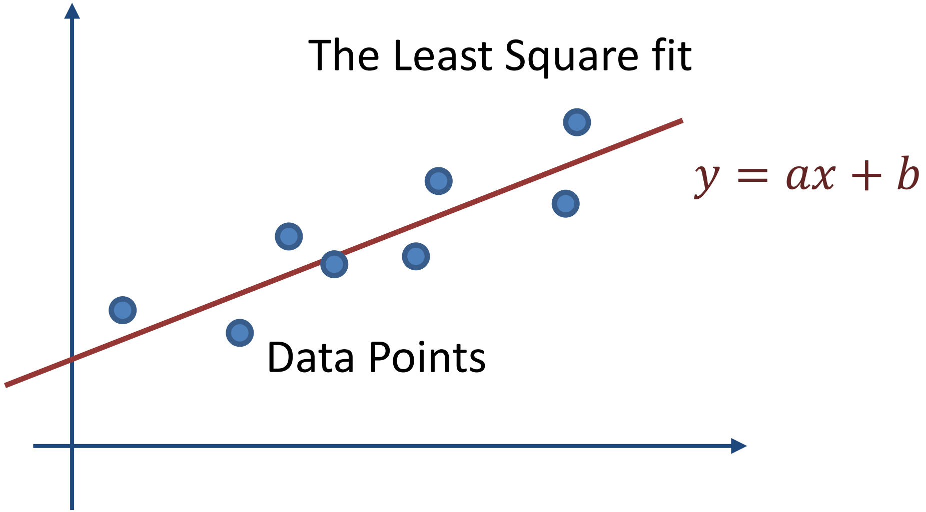
LSM Example

Given the following Data:

x	y
0	15
1	10
2	9
3	6
4	2
5	0

$$y = ax + b$$

We need to find a and b



$$15 = a \cdot 0 + b$$
$$10 = a \cdot 1 + b$$
$$9 = a \cdot 2 + b$$
$$6 = a \cdot 3 + b$$
$$2 = a \cdot 4 + b$$
$$0 = a \cdot 5 + b$$

$$Y = \Phi\theta$$

$$\begin{bmatrix} 15 \\ 10 \\ 9 \\ 6 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\theta_{LS} = (\Phi^T\Phi)^{-1}\Phi^T Y$$

Python Code

```
import numpy as np

Phi = np.array([[0, 1], [1, 1], [2, 1], [3, 1], [4, 1], [5, 1]])

Y = np.array([[15],[10],[9],[6],[2],[0]])

theta_ls = np.linalg.lstsq(Phi, Y, rcond=None)[0]
print(theta_ls)

theta_ls = np.linalg.inv(Phi.transpose() * np.mat(Phi)) * Phi.transpose() * Y
print(theta_ls)
```

Compare built-in LSM and LMS from scratch

From the Python code we get the following results:

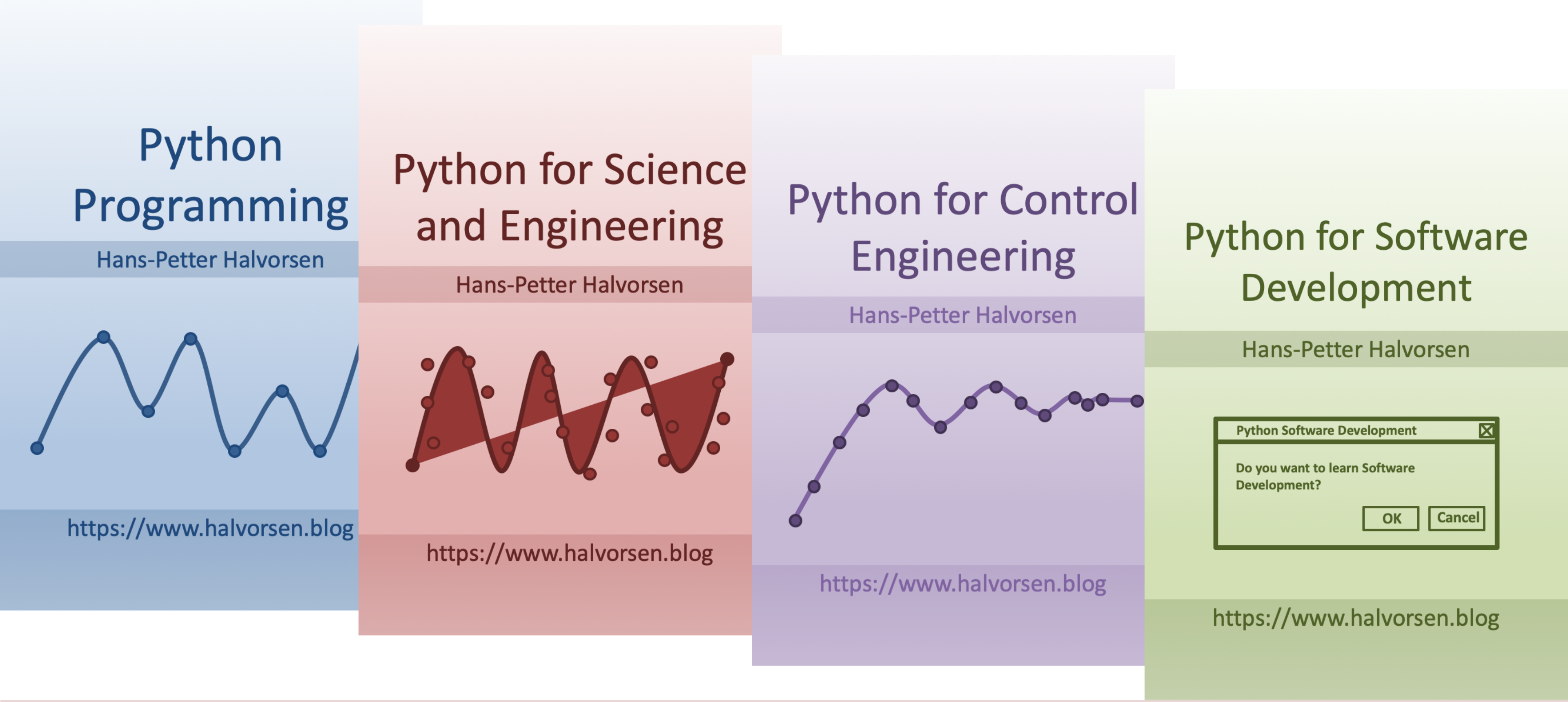
`[-2.91428571 14.28571429]`

This means $a = -2.91$ and $b = 14.29$

Or:

$$y = -2.91x + 14.29$$

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