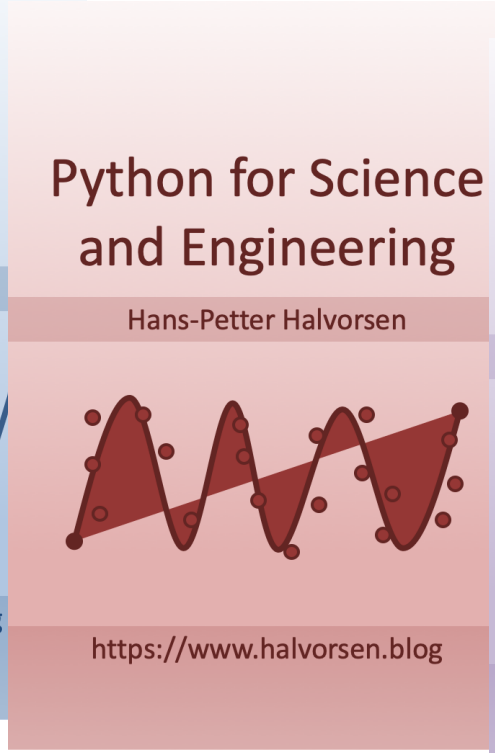
Interpolation in Python

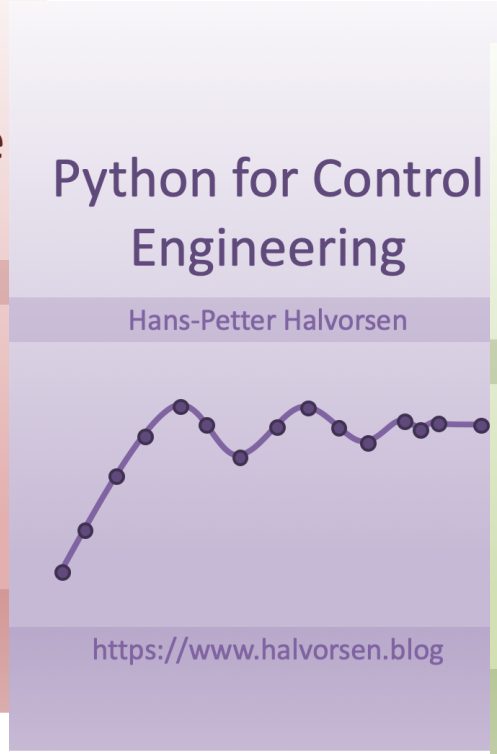
Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

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

Additional Python Resources





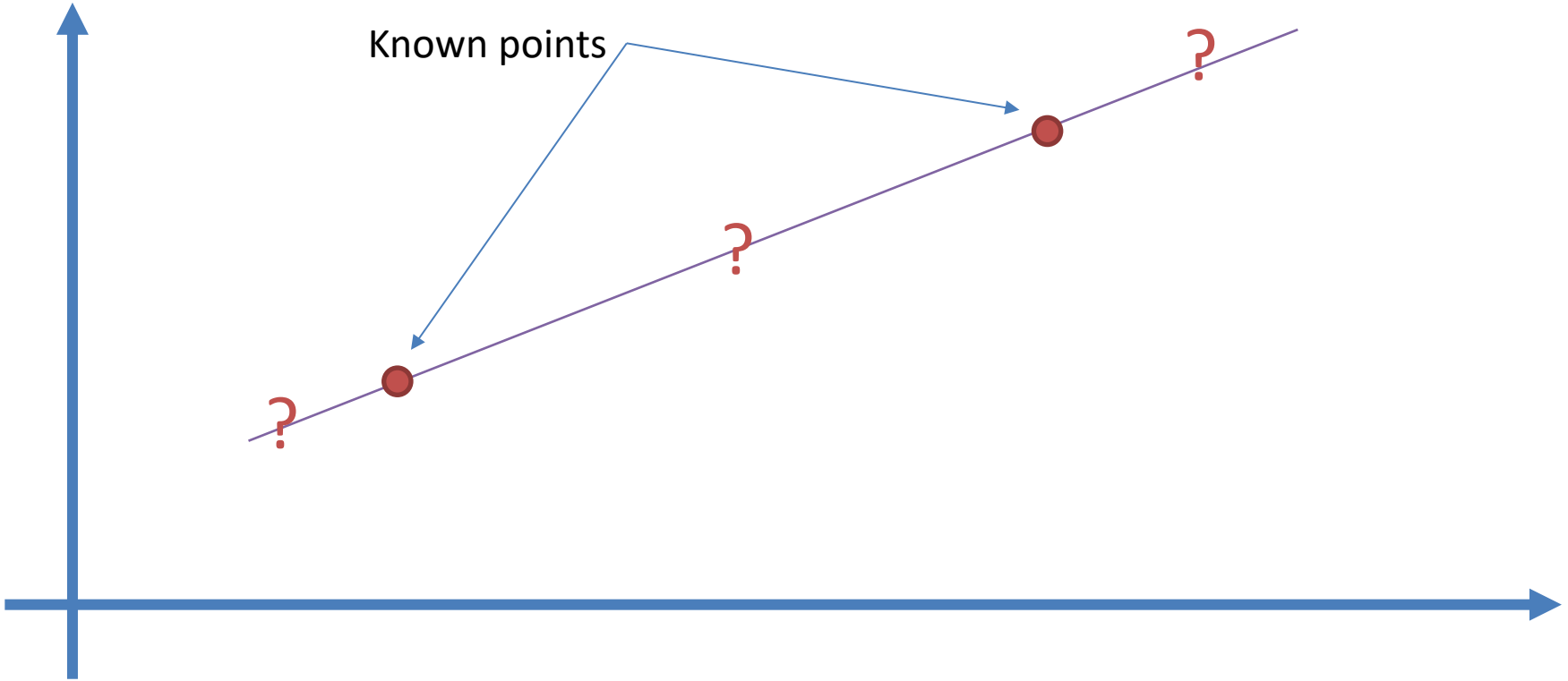
https://www.halvorsen.blog/documents/programming/python/

Interpolation

- Interpolation is used to estimate data points between two known points.
- The most common interpolation technique is **Linear Interpolation**.
- Others are **Quadratic**, **Cubic**, … (Splines)

Interpolation

Interpolation is used to estimate data points between two known points



Interpolation

We can use the following packages:

- **numpy**.interp
  https://docs.scipy.org/doc/numpy/reference/generated/numpy.interp.html
- **scipy**.interpolate
  https://docs.scipy.org/doc/scipy/reference/tutorial/interpolate.html

```
y_new = np.interp(x_new, x, y)
```

```
f = interpolate.interp1d(x, y)
```

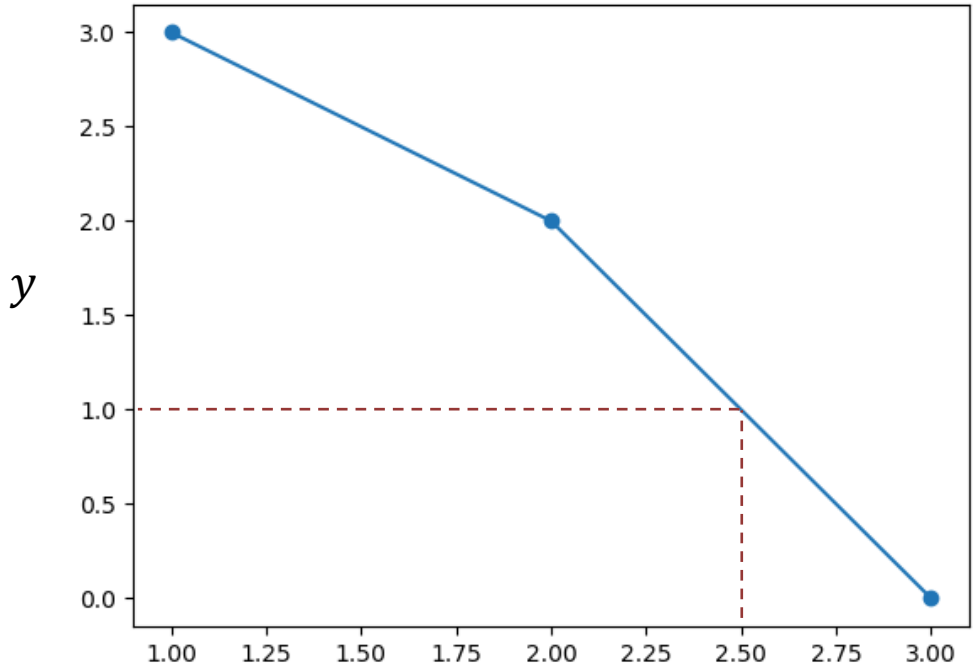
Interpolation - Example

Assume the following Data:

x	y
1	3
2	2
3	0

Assume we want to find the value for y when $x = 2.5$

We can plot the data points like this:

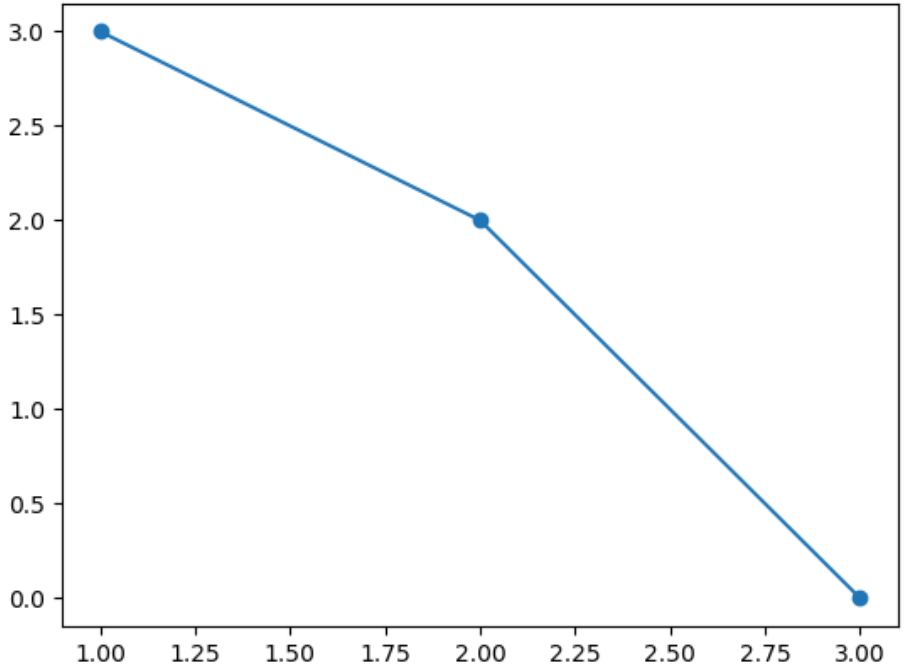


From the plot we see that $y = 1$ is a good guess

Interpolation - Example

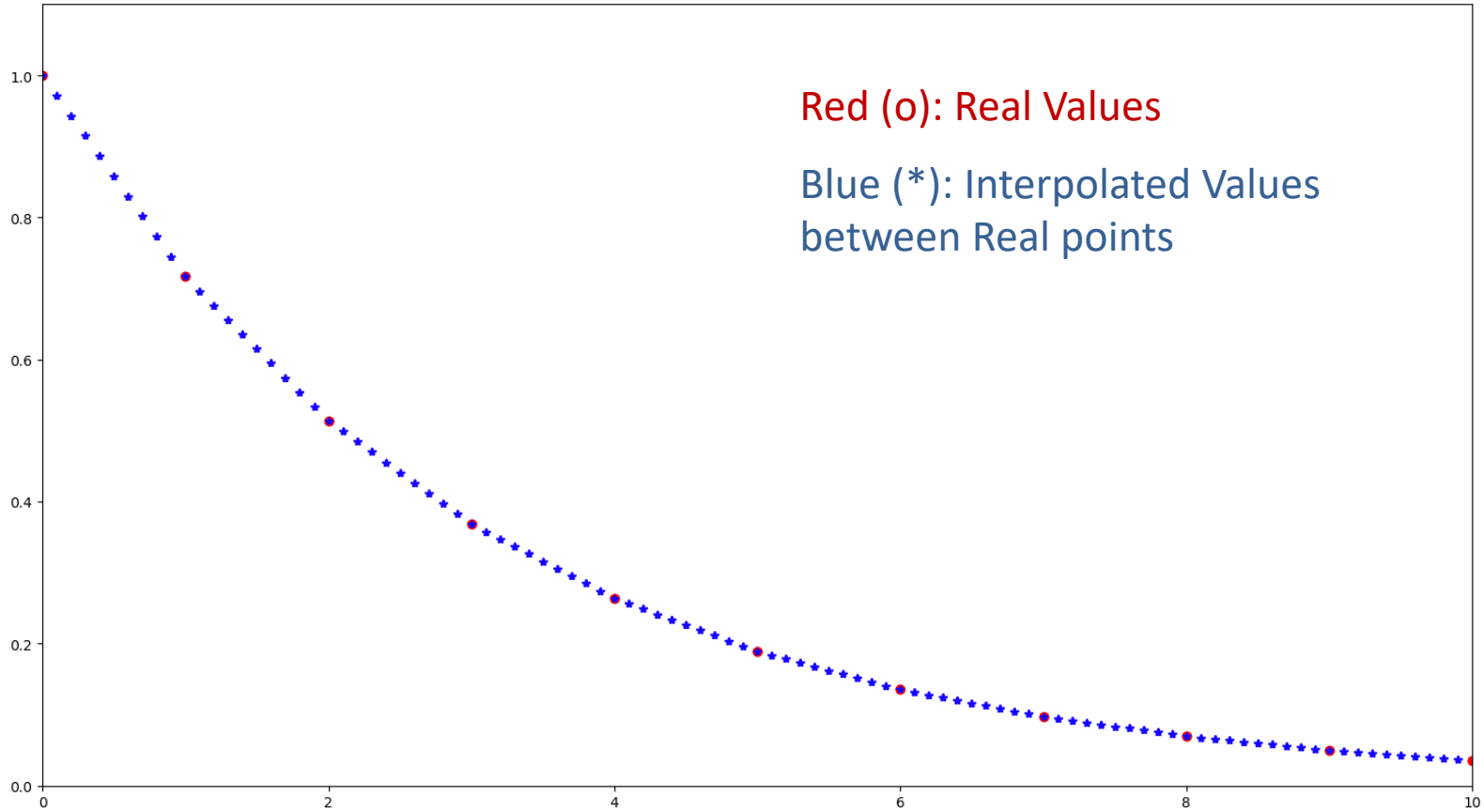
Python Code:

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

x = [1, 2, 3]
y = [3, 2, 0]

x_new = 2.5

y_new = np.interp(x_new, x, y)

print("New Interpolated Value:")
print(y_new)

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




```
New Interpolated Value:
1.0
```

SciPy cont.


1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
Red (o): Real Values
Blue (*): Interpolated Values between Real points

SciPy

Python Code:

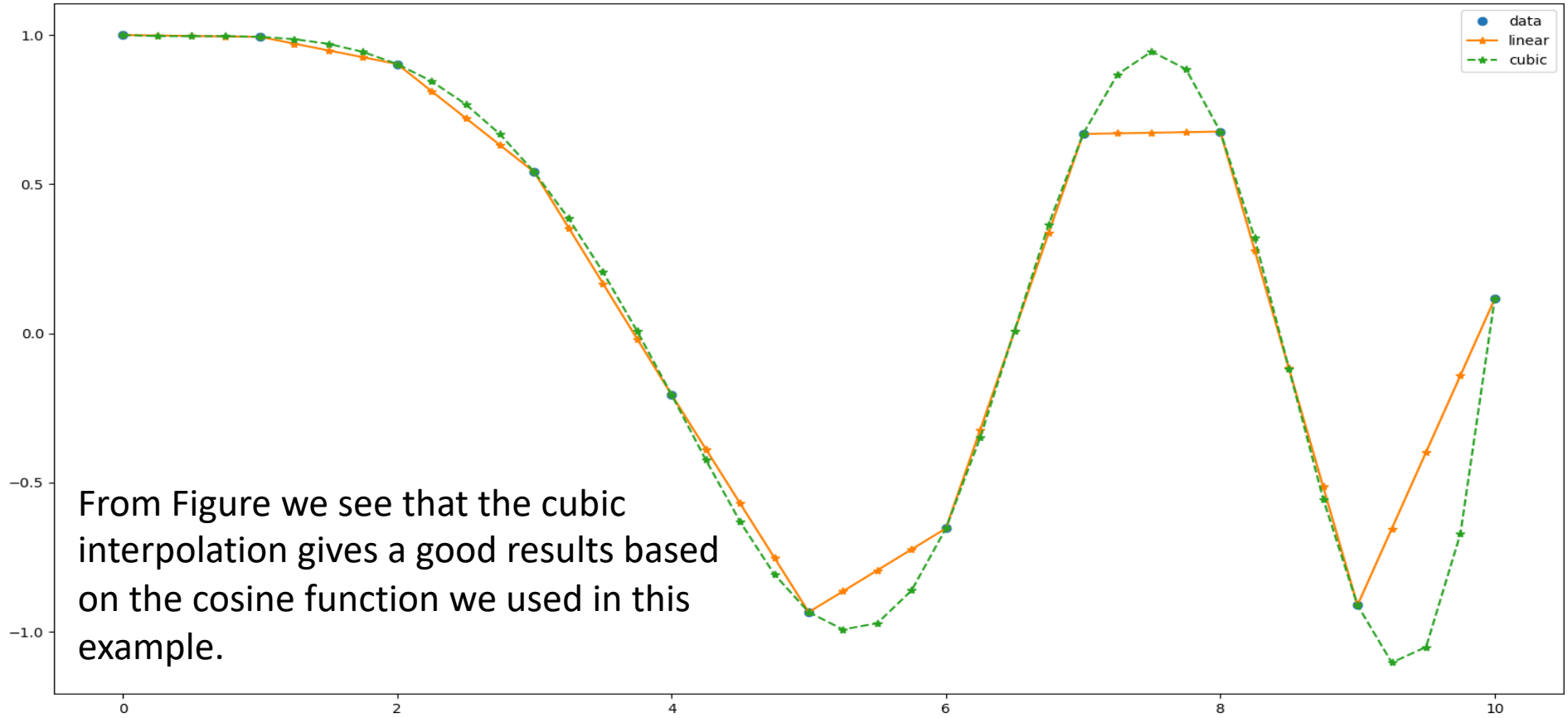
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import interpolate

x = np.arange(0, 11)
y = np.exp(-x/3.0)
f = interpolate.interp1d(x, y)

xstart=0
xstop=10
xstep=0.1
xnew = np.arange(xstart, xstop, xstep)
ynew = f(xnew)
plt.plot(x, y, 'or', xnew, ynew, '*b')
plt.axis([0,10,0,1.1])
plt.show()
```

Linear vs Cube Interpolation

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d

x = np.linspace(0, 10, num=11, endpoint=True)
y = np.cos(-x**2/9.0)

f = interp1d(x, y) #linear is default
f2 = interp1d(x, y, kind='cubic')


xnew = np.linspace(0, 10, num=41, endpoint=True)

plt.plot(x, y, 'o', xnew, f(xnew), '-*', xnew, f2(xnew), '--*')
plt.legend(['data', 'linear', 'cubic'], loc='best')
plt.show()
```

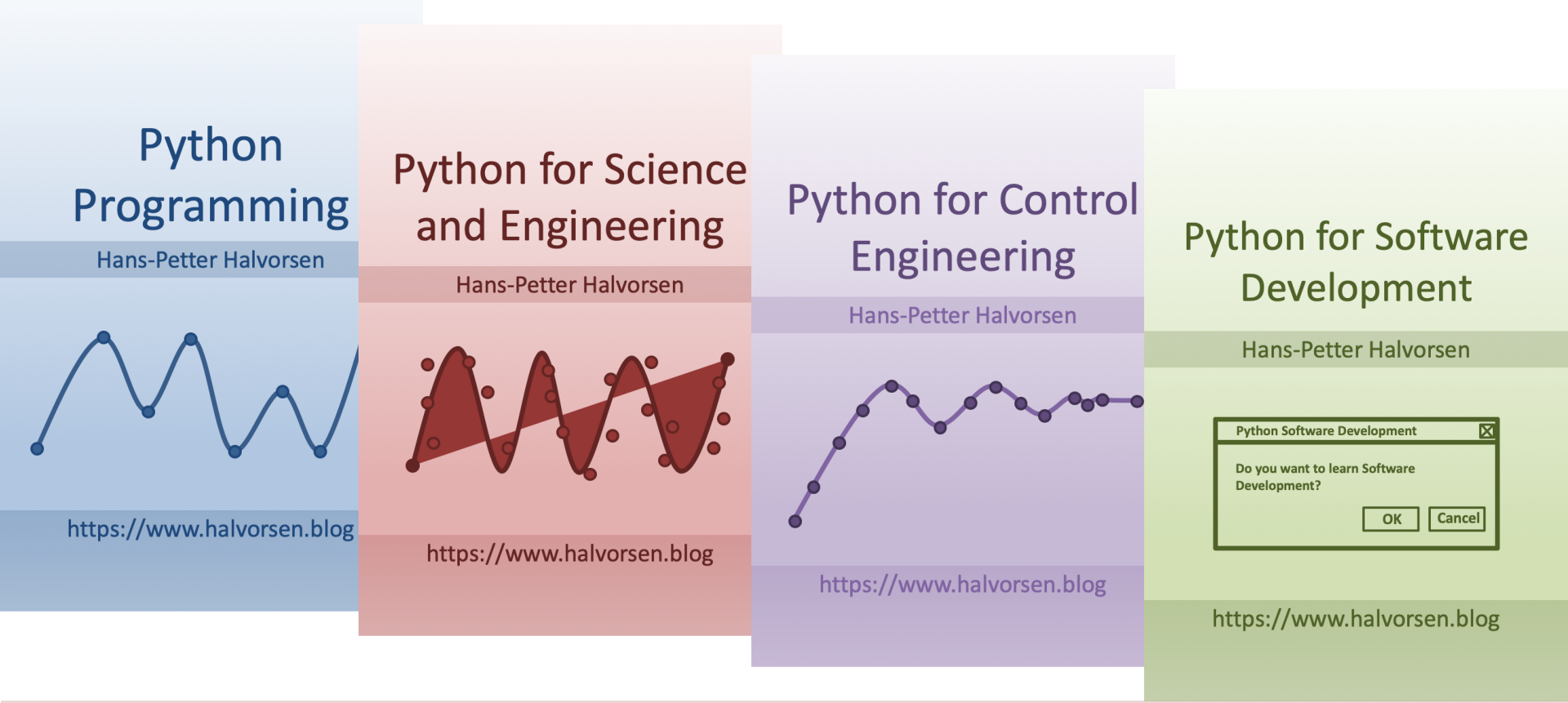
We start with a cos function and compare Linear and Cubic interpolation

Linear vs Cube cont.



From Figure we see that the cubic interpolation gives a good results based on the cosine function we used in this example.

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