

# **Numerische** Bestimmung der **Bogenlänge** einer **glatten Kurve** unter Verwendung von **NumPy**

*Dr. Frank Peuker, Probenvortrag*

**Numerische** Bestimmung der **Bogenlänge**  
einer **glatten Kurve** unter Verwendung von

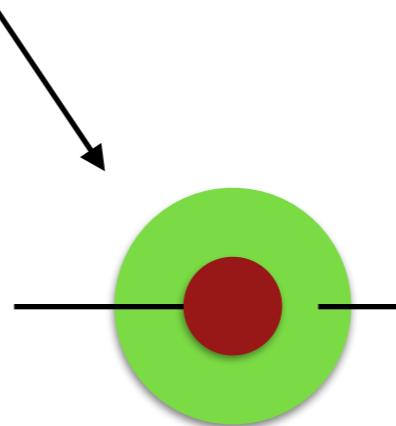
**NumPy**

??

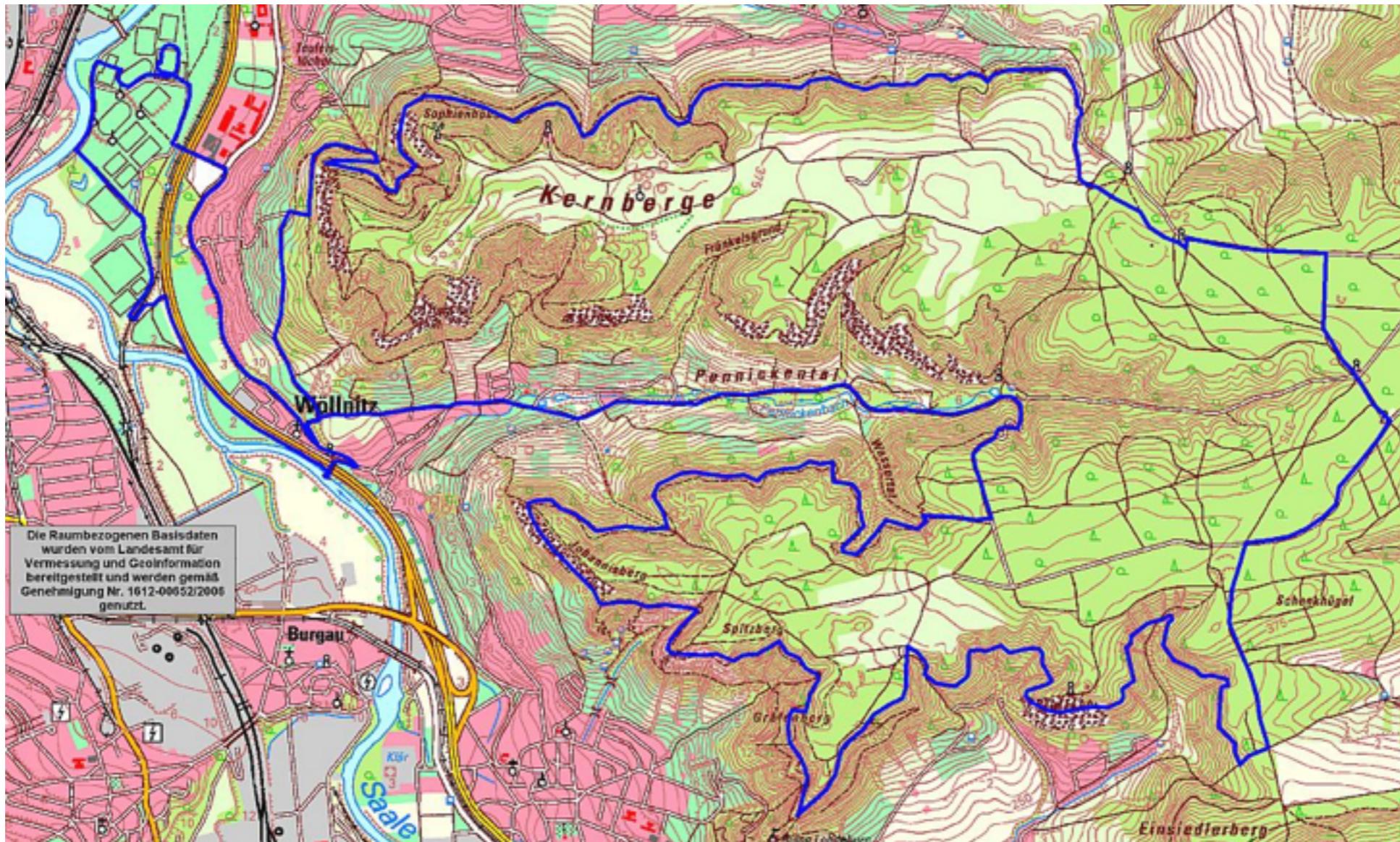
Python  
(Kern)

NumPy  
(Paket)

?

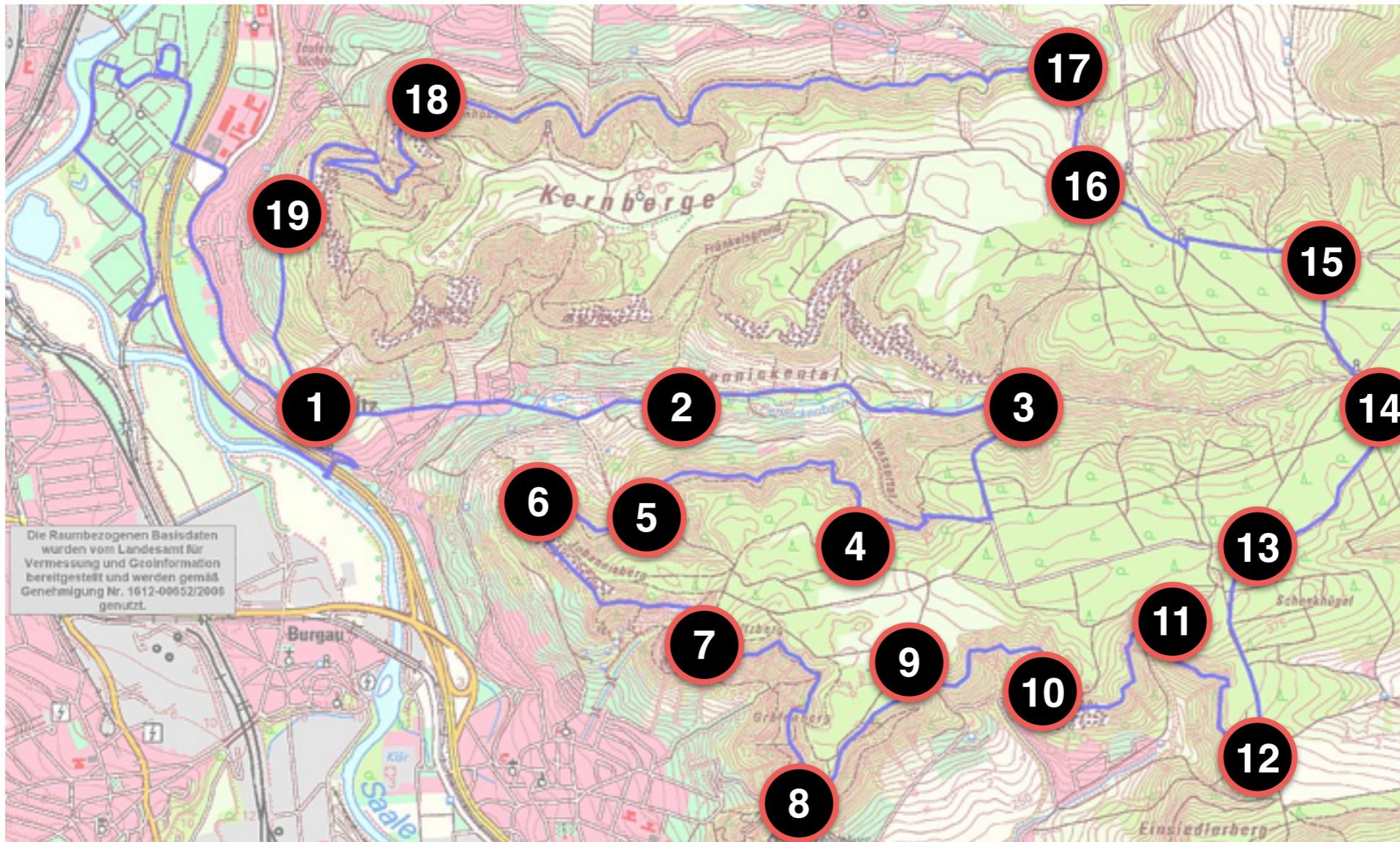


# Laufstrecke

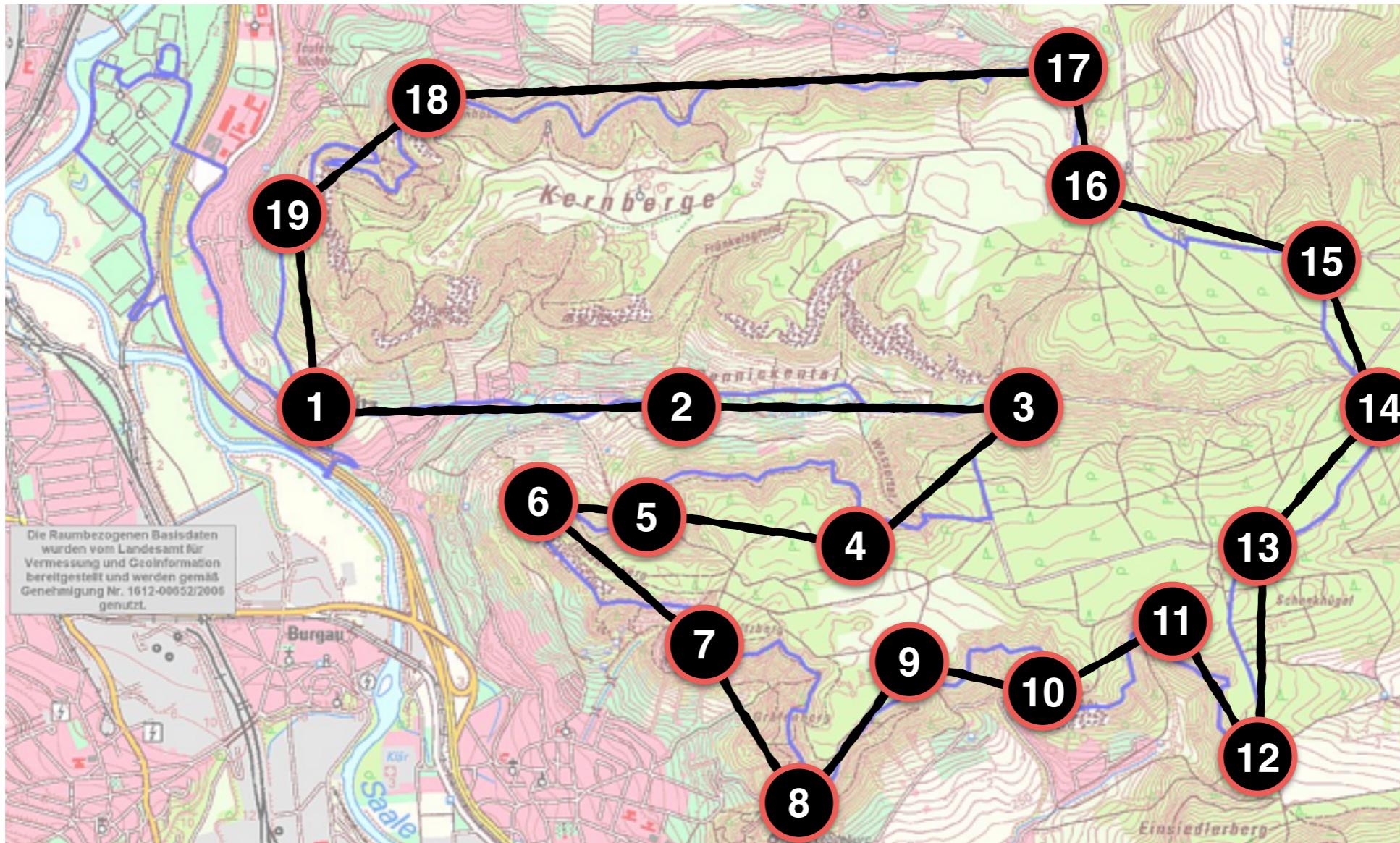


**Kernberglauf - 27 km**

# Wie berechnet man die Länge der Laufstrecke?



# Wie berechnet man die Länge der Laufstrecke?



Approximierte Strecke: Fehler wird mit mehr Punkten kleiner

# Was ist die Bogenlänge einer glatten Kurve?

$$y = f(x) = \sin(x)$$

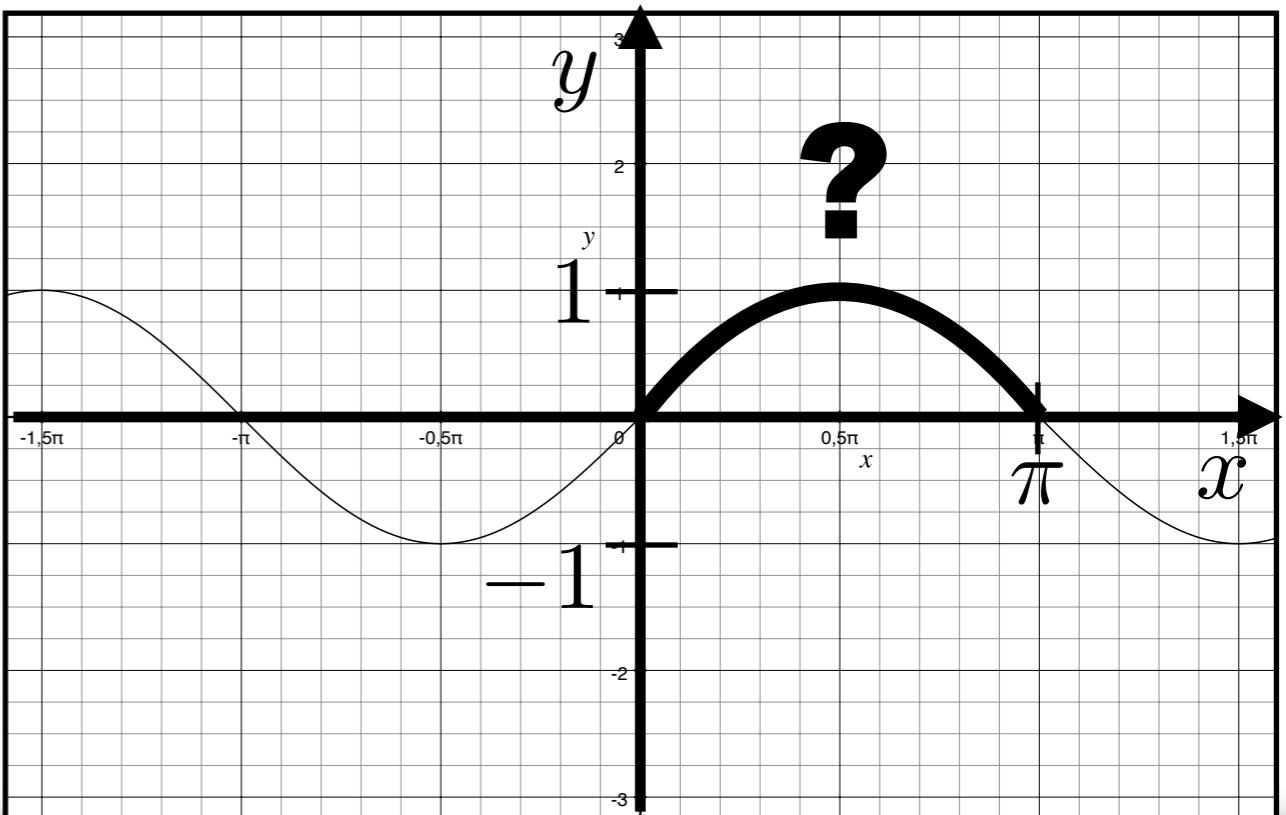
**Abbildung f:**

$$f : [a, b] \rightarrow \mathbb{R}$$

$$f \in C^1([a, b])$$

**Glatte Kurve:**

$$\Gamma_f = \{(x, y) \in \mathbb{R}^2 \mid y = f(x)\}$$

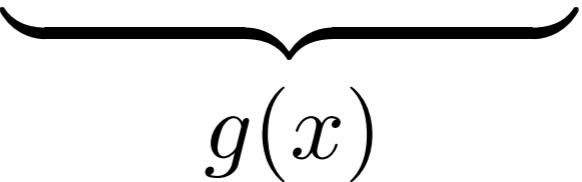


Berechnen Sie die Bogenlänge von  
 $\sin(x)$  auf dem Intervall  $[0, \pi]$

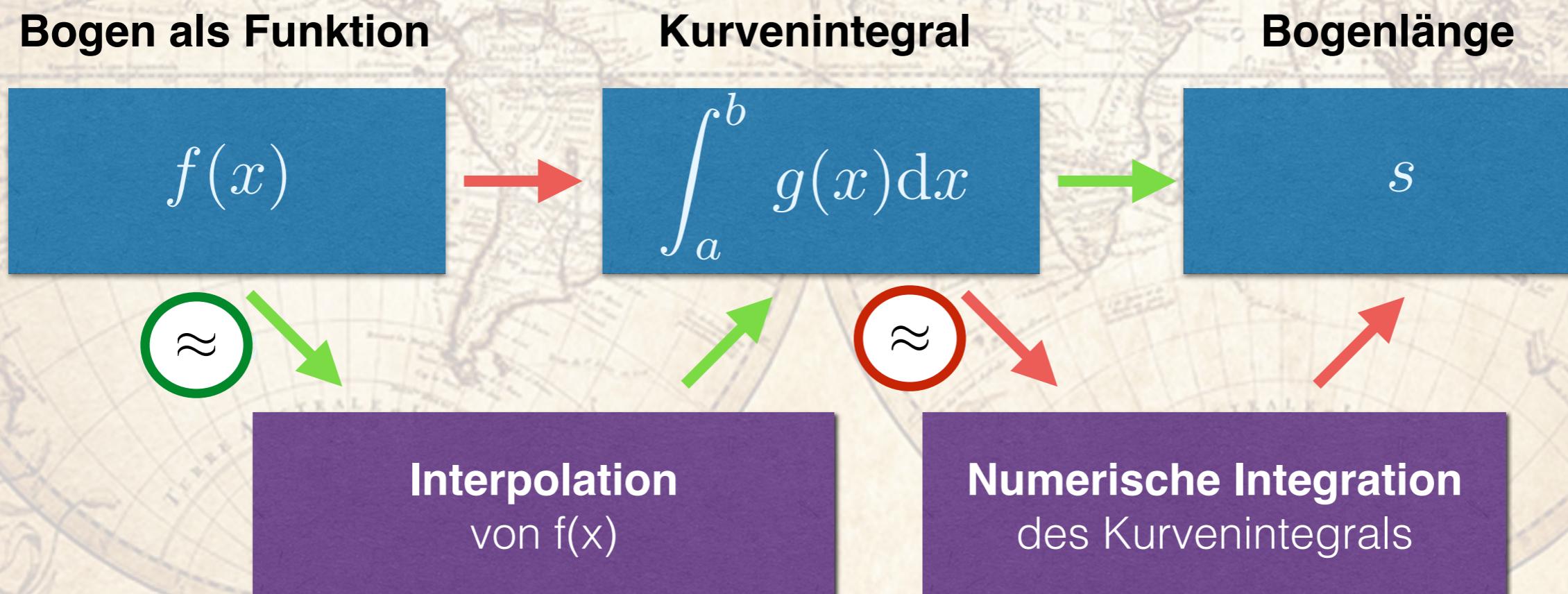
# Wie bestimmt man die Bogenlänge analytisch?

Tafel 1: Herleitung Kurvenintegral

**Kurvenintegral** bei explizit gegebener Funktion

$$\int_a^b \sqrt{1 + [f'(x)]^2} dx$$

$$g(x)$$

# Wie bestimmt man die Bogenlänge numerisch?



Analytik



Numerik



Weg 1



Weg 2

# Numerische Bestimmung durch Interpolation

Bogen als Funktion

$$f(x)$$



Kurvenintegral

$$\int_a^b g(x) dx$$

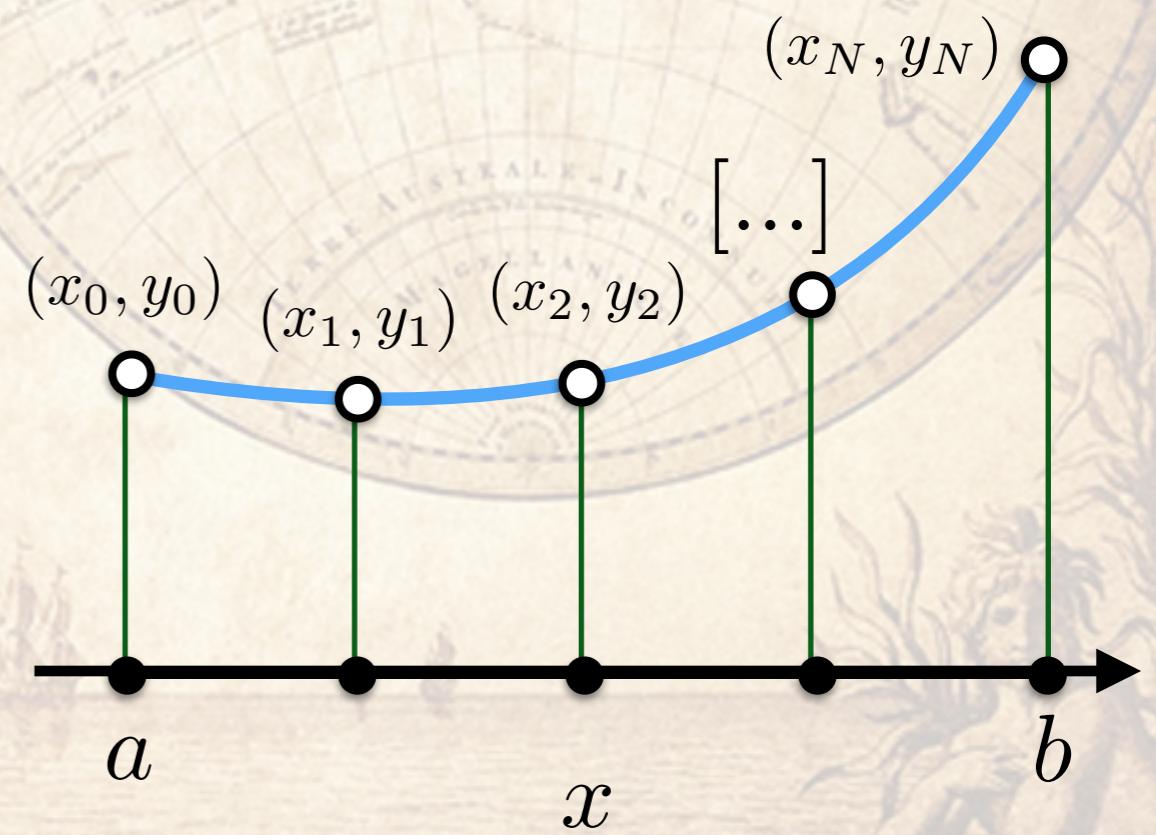
Bogenlänge

$$s$$

$$f(x) \approx \sum_{i=1}^{N-1} f_i(x)$$

Interpolation

**Idee:** Interpoliere so,  
dass Kurven-Integral exakt lösbar



# Numerische Bestimmung durch Lineare Interpolation

Kurvenintegral =  
Summe aus Abständen (Pythagoras)

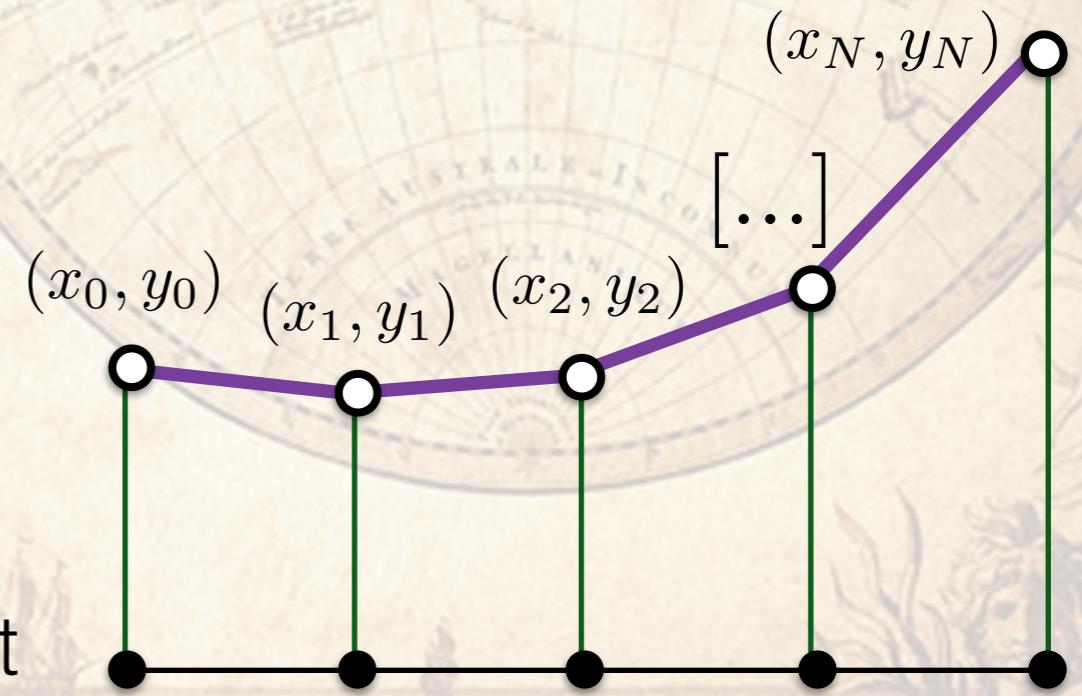
$$f(x)$$

$$s = \sum_{i=1}^{N-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$$

$$f(x) \approx \sum_{i=1}^{N-1} f_i(x)$$

## Lineare Interpolation:

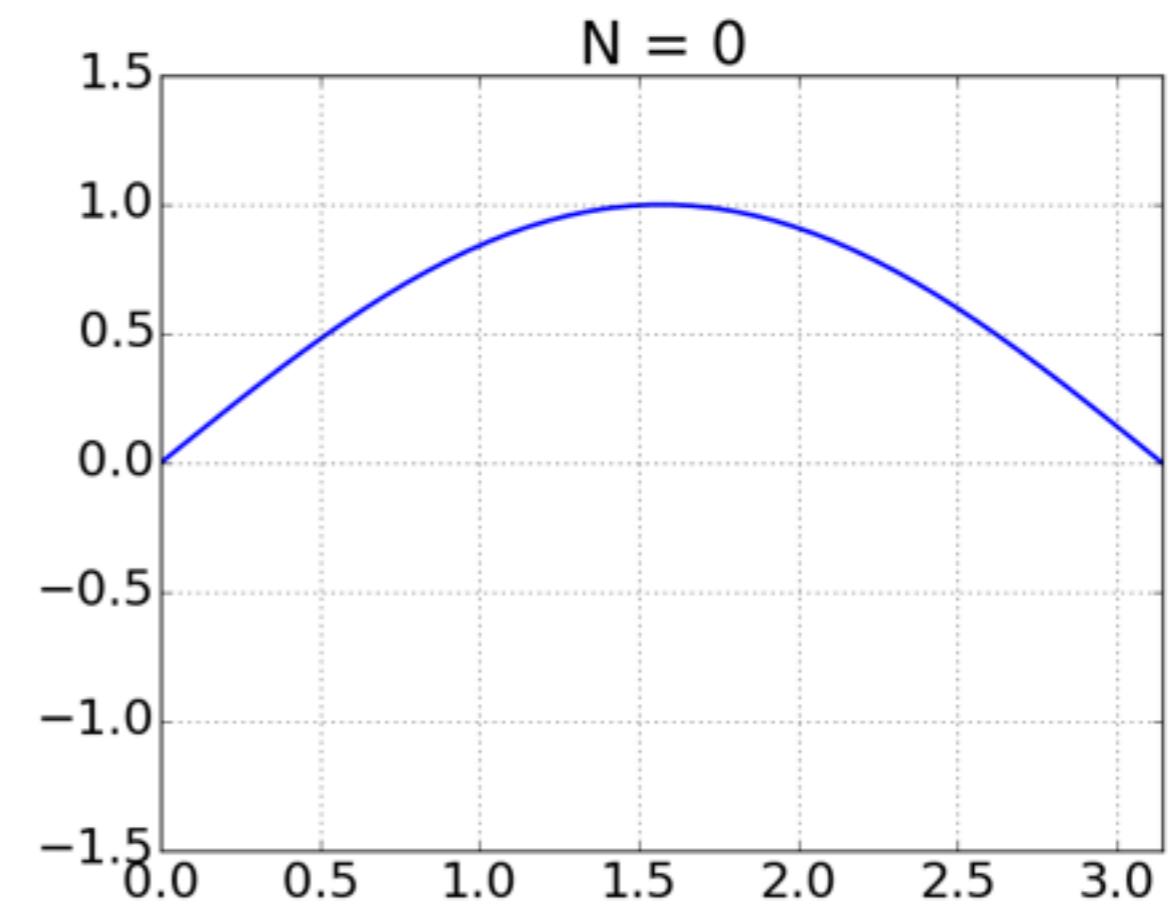
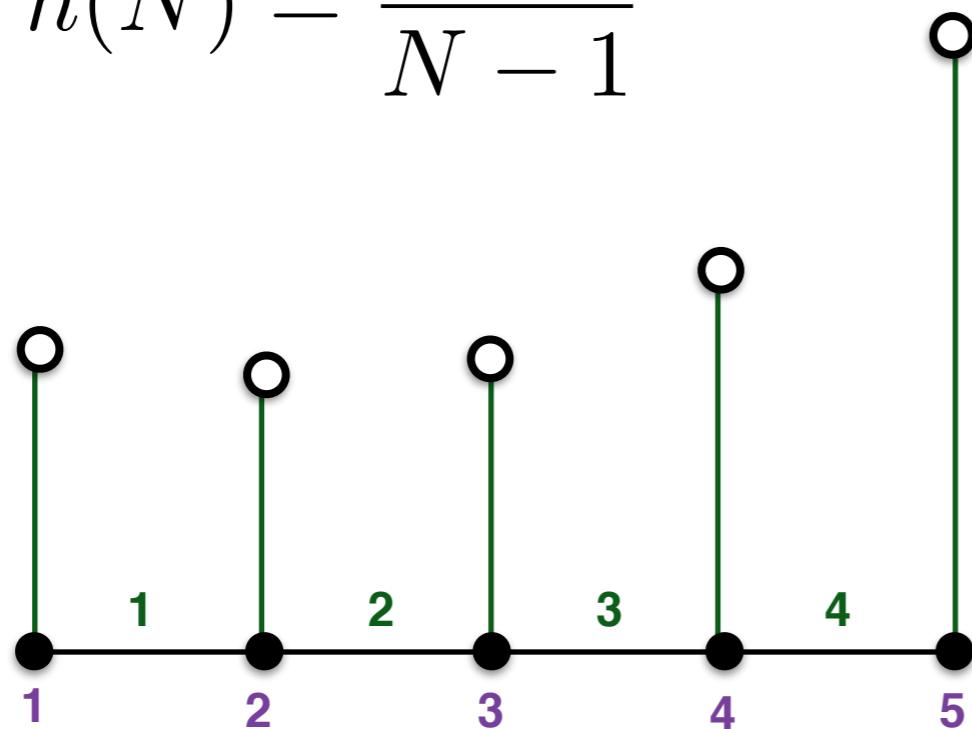
„Kurvenintegral schrumpft  
zum Pythagoras“



# Wahl des Gitters

- Gleichabständiges Gitter
- Gitter festgelegt wenn  $N$  (Zahl der Stützstellen) bzw.  $h$  (Gitterweite) vorgegeben

$$h(N) = \frac{b - a}{N - 1}$$



# interp1.py

- Satz des Pythagoras →
- Gitter und →  
Funktionswerte
- Aufsummierte →  
Abstände  
zwischen den Punkten
- Ausgabe in Konsole →  
print
- Das vorgegebene →  
Problem**
- Aufruf der →  
Hauptroutine

```
1 import numpy as np
2
3 def abstand(x1,y1,x2,y2):
4     return np.sqrt((x2-x1)**2+(y2-y1)**2)
5
6 def punktmenge(N):
7     xi = np.linspace(a,b,N)
8     yi = [f(x) for x in xi]
9     return [xi, yi]
10
11 def bogenlaenge(N):
12     [xi,yi] = punktmenge(N)
13     s = 0.
14     for i in range(N-1):
15         s = s + abstand(xi[i],yi[i],xi[i+1],yi[i+1])
16     return s
17
18 def dump(N,s):
19     #...
20
21 def f(x):
22     return np.sin(x)
23
24 a = np.double(0)
25 b = np.double(np.pi)
26 N = 16
27 s = bogenlaenge(N)
28 dump(N,s)
```



# Wie groß ist denn jetzt die Bogenlänge $s$ ?

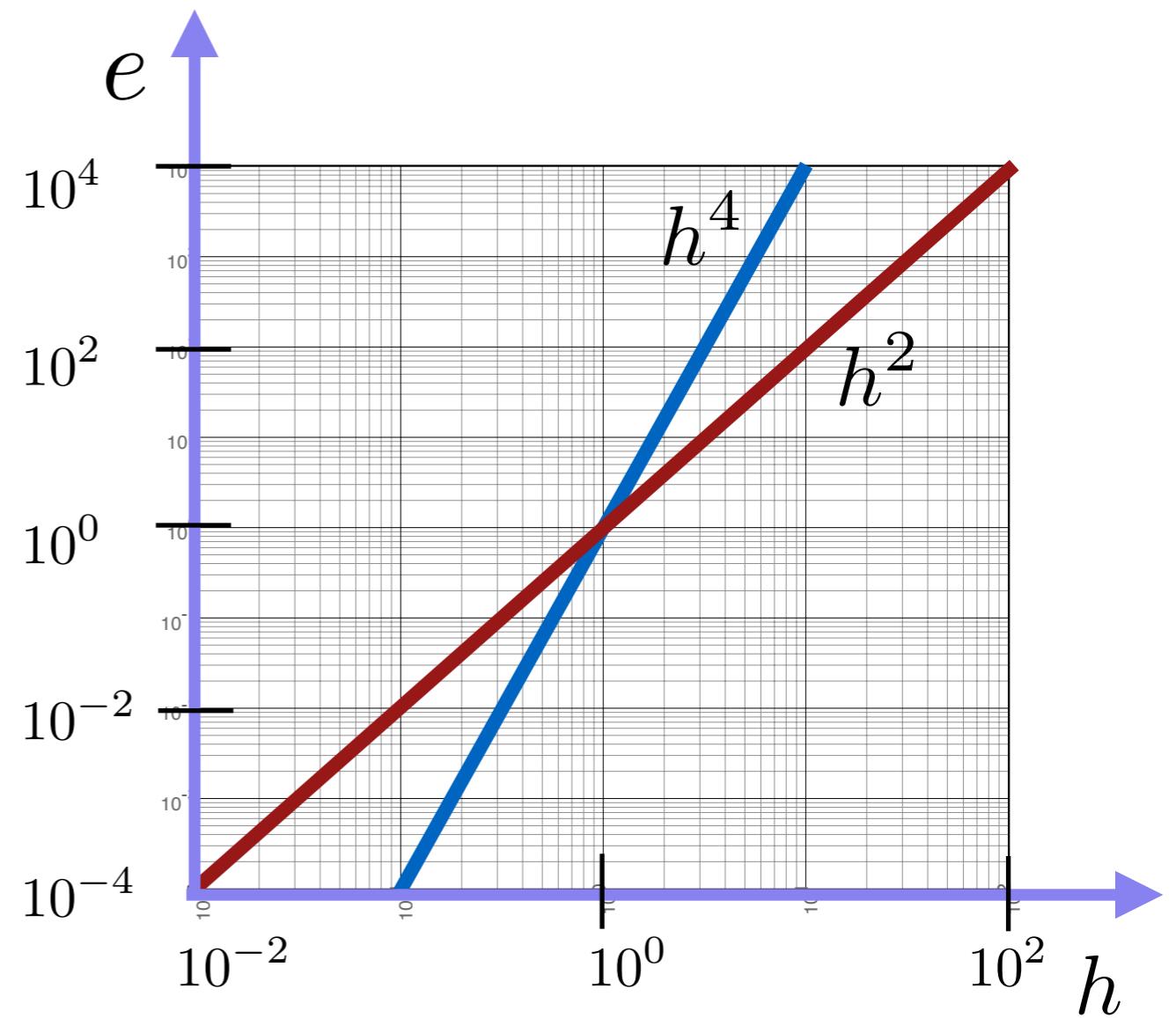
- $s$  ist abhängig von  $N$  — es gibt viele Bogenlängen!
- Welches  $N$  soll ich nehmen ?
- **Konvergenz** wichtig ! => Dann kann man Genauigkeit vorgeben und solange  $N$  erhöhen bis diese erreicht !

# Konvergenz-Analyse

- Gegenüberstellung von Fehler und Gitterweite
- **Doppelt-logarithmische Darstellung =>**  
Polynome sind Geraden!
- Anstieg = Konvergenzordnung

$$h_i = (b - a)/(N_i - 1)$$

$$e_i = \|s_{\text{ref}} - s_N\|$$



# konvergenz.py

Plot-Modul laden →

```
1 import numpy as np
2 from matplotlib import pyplot
3
4 # ...
5
6 def dump(N,s,e):
7     # ...
8
9 def dumpHeadline():
10    # ...
11
12 def drawPlot(hi,ei):
13    # ...
14
15 a = np.double(0)
16 b = np.double(np.pi)
17 N = 4096
18 sref = bogenlaenge(N)
19
20 Ni = [2,4,8,16,32,64,128,256,512,1024]
21 hi = []
22 ei = []
23 dumpHeadline()
24 for N in Ni:
25     s = bogenlaenge(N)
26     h = (b-a)/(N-1)
27     e = np.fabs(s-sref)
28     hi.append(h)
29     ei.append(e)
30     dump(N,s,e)
31
32 drawPlot(hi,ei)
```

Einfache Plot-Routine  
für doppelt-logarithmische  
Darstellung →

Da keine exakte Lsg.:  
Nehme numerische Lsg. mit  
hohem N →

Speichern Gitterabstand h  
und Fehlern e zu  
allen N in Ni →

Ausgabe des Plots →



Berechnen Sie die Bogenlänge von  
 $\sin(x)$  auf dem Intervall  $[0, \pi]$



Fünf signifikante Stellen reichen mir  
erstmal.



Moment ...



**3,8202**

# Zur Übung

> **git clone https://github.com/eah-informatik/probevorlesung.git**

> **git pull**

- **Ü1:** Leiten Sie den Satz des Pythagoras aus dem Kurvenintegral her. (Kurvenintegral einer linearen Funktion)
- **Ü2:** Berechnen Sie folgende Bogenlängen und interpretieren sie die Konvergenz-Plots.

$$f(x) = \sqrt{1 - x^2} \quad \text{auf } [-1, 1]$$

$$f(x) = \sin(20x) \cdot \sqrt{1 - x^2} \quad \text{auf } [-1, 1]$$

$$f(x) = \sin(x) \quad \text{auf } [0, 50\pi]$$

# Quiz 1

Mit welcher Formel lässt sich die Gitterweite aus der Anzahl der Stützstellen N bestimmen?

D

$$h = (b+a)/(N-1)$$

E

$$h = (b-a)/(N-1)$$

N

$$h = (b-a)/N$$

© Universität Regensburg

# Quiz 2

**Aus welchem Satz lässt sich der Abstand zwischen zwei Punkten berechnen?**

D

Satz des Archimedes

E

Satz des Thales

N

Satz des Pythagoras

# Quiz 3

**In welchem Diagramm-Typ werden Polynome gerade gebogen ?**

D

Beide Achsen logarithmisch

E

Normale Achsen

N

Eine Achse normal, die andere logarithmisch

# Quiz 4

Mit welcher NumPy-Funktion erzeugt man in einer einzigen Programmierzeile ein gleichabständiges Gitter?

D

`numpy.sum([a,b,N])`

E

`numpy.linspace(a,b,N)`

N

`numpy.array([a,b,N])`

E

N

D

E