Python

Motivation

Ähnliche Vorteile wie Julia:

- frei (nicht kommerziell)
- offen
- leicht

Getting started

VSC (Visual Studio Code) installieren und Python-Paket laden ...

Es gibt viele Alternative Arbeitsweisen ...

Unter Ubuntu ist Python meist schon vorhanden, kann im Terminal direkt aufgerufen werden ...

Python – Example

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# Parameterdefinitionen
a = 0.6
b = 0.13533

# Liste der Parameter fuer die Funktion
p = [a, b]

# Weitere Parameter
c = 8.8
d = 8.2

# Liste der weiteren Parameter
q = [c, d]

def time_dependency(x, p, q):
# Extrahiere die Werte aus den Parametern p und q
a = p[1]
b = p[2]
c, d = q

# Berechnungen fuer die zeitabhaengigen Parameter
z = 2 * np.pi * x / 360.0

# Temperatur
t = ct + at * np.sin(b*z - t)

return t
```

Python – Example continued

```
# Definition der Differentialgleichungen
def model_rhs(y, t, params, q):
# Zeitabhaengige Parameter berechnen
t = time_dependency(t, params, q)
# Extrahieren der festen Parameter
a, b, c, d = params

# Differentialgleichungen
yp = np.zeros(2)
yp[0] = a*y[0]-b*y[0]*y[1]
yp[1] = c*y[0]*y[1]-d*y[1]
return yp
# Messwerte als Liste (aus den bereitgestellten Daten)
measurements = [
[0.0, 11.7, 0.12],
[2.92024, 11.34845, 0.11908],
[10.69874, 10.6349, 0.11883],
[22.93707, 9.92608, 0.11863],
]
y0 = measurements[1:]  # Anfangswerte y[0] bis y[1]
# Loesung der Differentialgleichungen
solution = odeint(model_rhs, y0, t, args=(p, q))

# Plotten der Ergebnisse
plt.figure(figsize=(12, 8))
plt.plot(t, solution[:, 0], label='y0: concentrations')
plt.xlabel('Zeit (Tage)')
plt.ylabel('Konzentration')
plt.legend()
plt.title('Mein Modell')
plt.grid(True)
plt.show()
```