ODExamples

Julian Bauer

October 12, 2018

Contents

1 Problems **2**
1.1 Reaction kinetics 2
1.1.1 Problem statement 2
1.1.2 Interpretation 2

2 Python **3**
2.1 Code 3
2.2 Pictures 5

1 Problems

1.1 Reaction kinetics

Source: [Fritzen(2013)]

1.1.1 Problem statement

$$\underline{\dot{u}} = \begin{bmatrix} \dot{u}_0 \\ \dot{u}_1 \\ \dot{u}_2 \end{bmatrix} = \underbrace{k_0 u_u \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}}_{term0} + \underbrace{k_1 u_1 u_2 \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}}_{term1} + \underbrace{k_2 u_1 \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}}_{term2} \quad (1)$$

Symbol	Interpretation with $i \in [0, 1, 2]$
u_i	Contentration of particles of type i
k_i	Constant of proportionallity of term i
$\dot{u}_i$	Rate of change of u_i

Table 1: Symbols in Equation 1

1.1.2 Interpretation

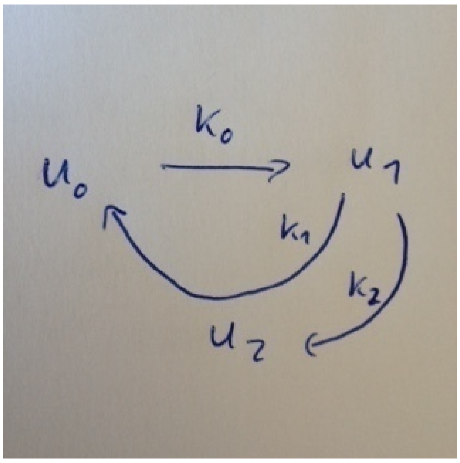


Figure 1: Visualization of Equation 1

Term0 Particles of type 0 turn into particles of type 1, decreasing the concentration u_0 and increasing u_1. The rate of change is proportional to the concentration u_0 with the constant of proportionallity k_0.

Term1 Particles of type 1 turn into particles of type 0, decreasing the concentration u_1 and increasing u_0. The rate of change is proportional to the product of the concentrations u_0 and u_1 with the constant of proportionallity k_1. This means, the presence of type 2 particles catalyze the transition of type 1 particels into type 0 particles. Term0 and Term1 counteract each other.

Term2 Particles of type 1 turn into particles of type 2, decreasing the concentration u_1 and increasing u_2. The rate of change is proportional to the concentration u_1 with the constant of proportionallity k_2.

2 Python

2.1 Code

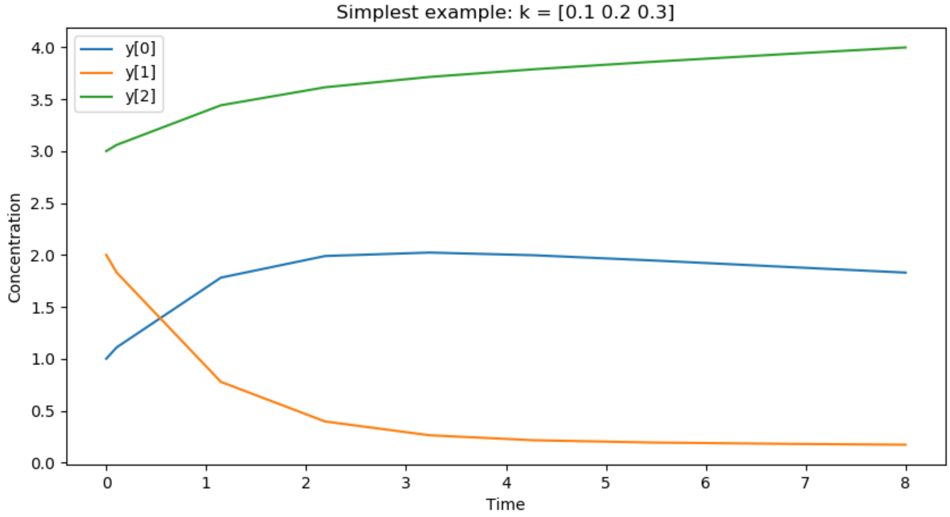
Note: Symbols of concentrations u_i are replaced by y_i.

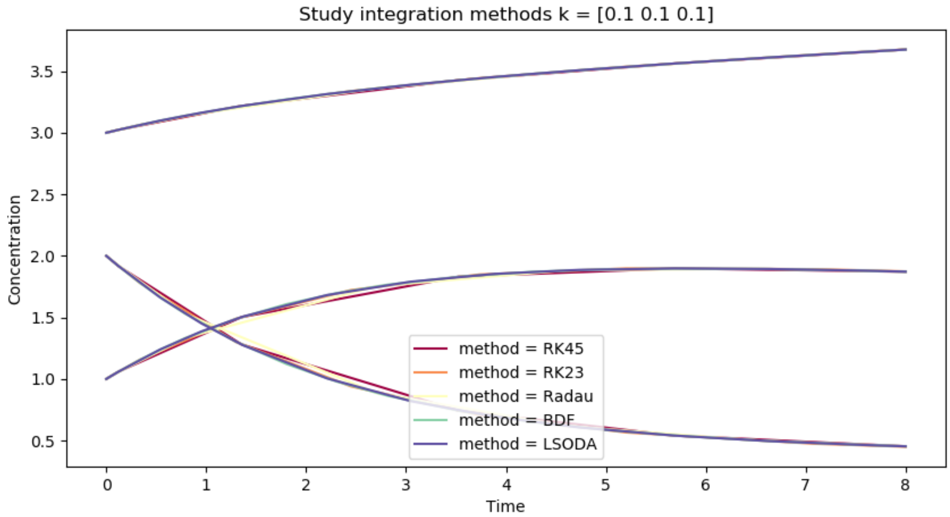
```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt          # Plotting
import matplotlib

#####################################################################
# Simplest example

# Define parameter
k = [0.1, 0.2, 0.3]       # Mol / second

# Define rate as function
def kineticsSimple(t, y):
    '''Pass k into function as global variable'''
    return ( k[0] * y[0]           * np.array([-1.0, 1.0, 0.0]) \
           + k[1] * y[1] * y[2]    * np.array([ 1.0,-1.0, 0.0]) \
           + k[2] * y[1]           * np.array([ 0.0,-1.0, 1.0]) )

# Prepare plotting
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111)
ax.set_title('Simplest example: '+'k = ['+' '.join([str(ki) for ki in k])+']')

# Solve
sol = solve_ivp(
            fun=kineticsSimple,
            t_span=[0, 8],       # Seconds
            y0=[1,2,3],          # Mol
            )
# Plot
for indexComp, comp in enumerate(sol.y):
    ax.plot(sol.t, comp, label='y[{}]'.format(str(indexComp)))

ax.set_ylabel('Concentration')
ax.set_xlabel('Time')
ax.legend()
plt.savefig('simple.png')


#####################################################################
# Define rate
def kinetics(t, y, k):
    '''Pass k into funtcion as additional parameter'''
    return ( k[0] * y[0]           * np.array([-1.0, 1.0, 0.0]) \
           + k[1] * y[1] * y[2]    * np.array([ 1.0,-1.0, 0.0]) \
           + k[2] * y[1]           * np.array([ 0.0,-1.0, 1.0]) )

#####################################################################
# Study integration method

# Define parameter
k = [0.1, 0.1, 0.1] # Mol / second

# Prepare plotting
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111)
ax.set_title('Study integration methods '+'k = ['+' '.join([str(ki) for ki in k])+'
    ]')

cmap = matplotlib.cm.get_cmap('Spectral')
```

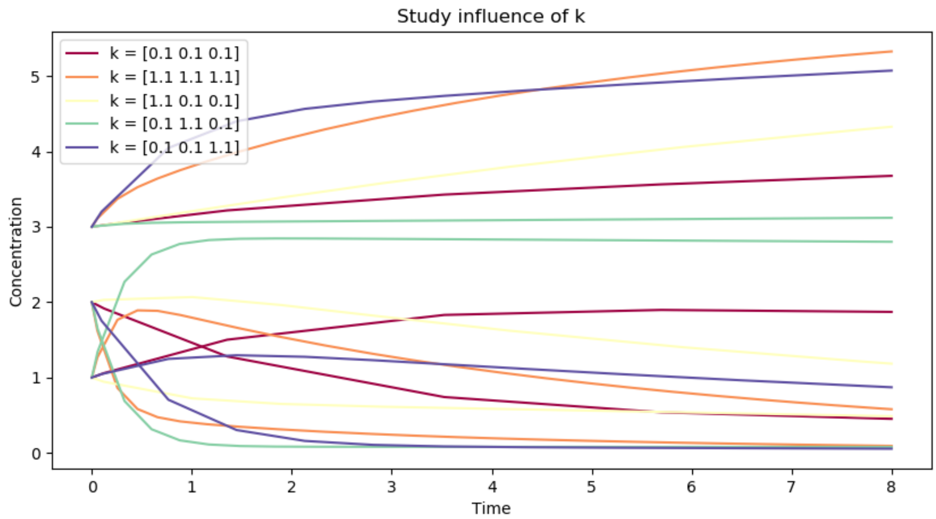
```
methods = ['RK45', 'RK23', 'Radau', 'BDF', 'LSODA']
for index, method in enumerate(methods):
    colorIndex = index / (len(methods)-1)

    # Solve
    sol = solve_ivp(
                fun=lambda t, y: kinetics(t, y, k),
                t_span=[0, 8],      # Seconds
                y0=[1,2,3],         # Mol
                method=method,
                )
    # Plot
    for indexComp, comp in enumerate(sol.y):
        ax.plot(
            sol.t,
            comp,
            label='method = '+method if indexComp == 0 else None,
            color=cmap(colorIndex),
            )

ax.set_ylabel('Concentration')
ax.set_xlabel('Time')
ax.legend()
plt.savefig('influence_method.png')

####################################################################
# Study kinetics parameter

# Prepare plotting
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111)
ax.set_title('Study influence of k')

cmap = matplotlib.cm.get_cmap('Spectral')

# Define parameter
studyK = [
    [0.1, 0.1, 0.1],
    [1.1, 1.1, 1.1],
    [1.1, 0.1, 0.1],
    [0.1, 1.1, 0.1],
    [0.1, 0.1, 1.1],
    ]

for index, k in enumerate(studyK):
    colorIndex = index / (len(studyK)-1)

    # Solve
    sol = solve_ivp(
                fun=lambda t, y: kinetics(t, y, k),
                t_span=[0, 8],      # Seconds
                y0=[1,2,3],         # Mol
                method='RK45',
                )
    # Plot
    for indexComp, comp in enumerate(sol.y):
        ax.plot(
                sol.t,
                comp,
                label='k = ['+' '.join([str(ki) for ki in k])+']' if indexComp == 0 else None,
                color=cmap(colorIndex),
                )

ax.set_ylabel('Concentration')
ax.set_xlabel('Time')
ax.legend()
plt.savefig('influence_k.png')
```

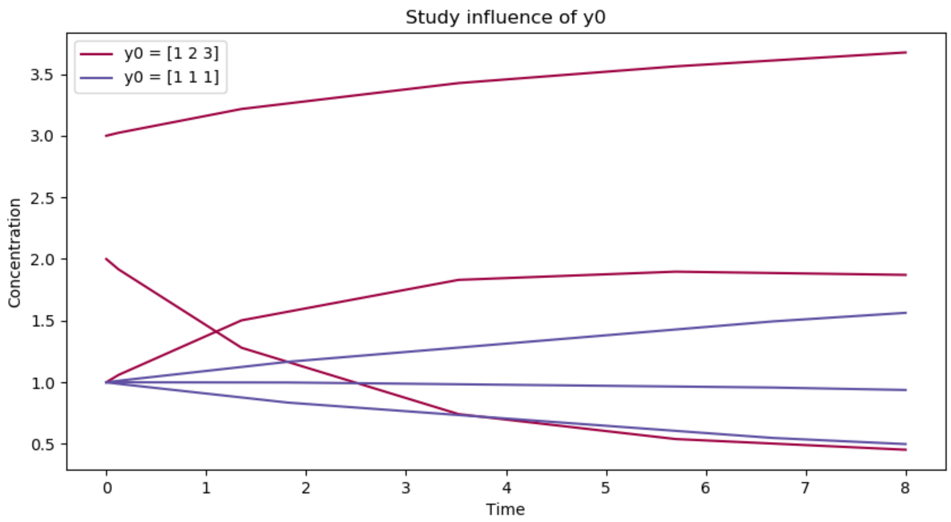
```python


##########################################################################
# Study initial values

# Prepare plotting
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111)
ax.set_title('Study influence of y0')

cmap = matplotlib.cm.get_cmap('Spectral')

# Define parameter

k = [0.1, 0.1, 0.1] # Mol / second

studyY0 = [
    [1, 2, 3],
    [1, 1, 1],
    ]

for index, y0 in enumerate(studyY0):
    colorIndex = index / (len(studyY0)-1)

    # Solve
    sol = solve_ivp(
                fun=lambda t, y: kinetics(t, y, k),
                t_span=[0, 8],      # Seconds
                y0=y0,              # Mol
                method='RK45',
                )
    # Plot
    for indexComp, comp in enumerate(sol.y):
        ax.plot(
                sol.t,
                comp,
                label='y0 = ['+' '.join([str(i) for i in y0])+']' if indexComp == 0 else None,
                color=cmap(colorIndex),
                )

ax.set_ylabel('Concentration')
ax.set_xlabel('Time')
ax.legend()
plt.savefig('influence_y0.png')

plt.show()
```

2.2 Pictures


Simplest example: k = [0.1 0.2 0.3]
y[0]
y[1]
y[2]
Concentration
Time
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
1
2
3
4
5
6
7
8


Study integration methods k = [0.1 0.1 0.1]
method = RK45
method = RK23
method = Radau
method = BDF
method = LSODA
Concentration
Time
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1
2
3
4
5
6
7
8


Study influence of k
k = [0.1 0.1 0.1]
k = [1.1 1.1 1.1]
k = [1.1 0.1 0.1]
k = [0.1 1.1 0.1]
k = [0.1 0.1 1.1]
Concentration
Time
0
1
2
3
4
5
0
1
2
3
4
5
6
7
8


Study influence of y0
y0 = [1 2 3]
y0 = [1 1 1]
Concentration
Time
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
1
2
3
4
5
6
7
8

List of Figures

1 Visualization of Equation 1 2

List of Tables

1 Symbols in Equation 1 2

Listings

../01_reactionKinetics/reactionKinetics.py 3

References

[Fritzen(2013)] Fritzen, F., 2013. Skriptum Rechnerunterstützte Mechanik II. IFM KIT.