Assignment 8

Angewandte Modellierung 25

Carl Colmant

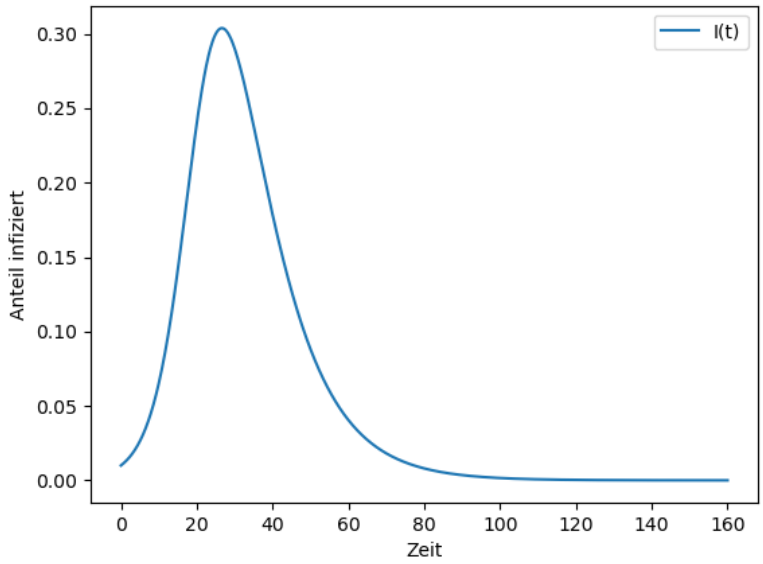
June 27, 2025

Exercise 1.

In Aufgabe eins soll das SIR Modell in python und R simuliert werden. Dazu nutze ich in python diesen code:

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

# Parameter
a, beta = 0.3, 0.1
S0, I0, R0 = 0.99, 0.01, 0.0

def sir(y, t, a, beta):
    S, I, R = y
    dS = -a * S * I
    dI = a * S * I - beta * I
    dR = beta * I
    return [dS, dI, dR]

t = np.linspace(0, 160, 1600)
sol = odeint(sir, [S0, I0, R0], t, args=(a, beta))

plt.plot(t, sol[:, 1], label='I(t)')
plt.xlabel('Zeit')
plt.ylabel('Anteil infiziert')
plt.legend();
plt.show()
```

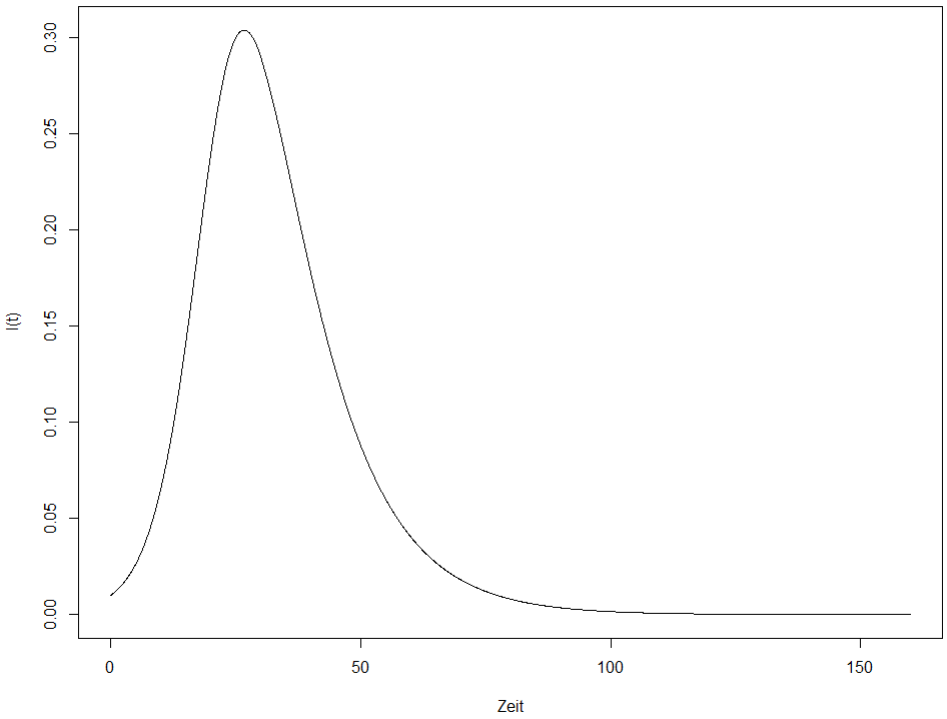
mit dem entstehenden plot:



In R sieht das ganze dann so aus:

```r
if (!require('deSolve')) {
  install.packages('deSolve')
}
library(deSolve)
a <- 0.3; beta <- 0.1
init <- c(S=0.99, I=0.01, R=0)

sir <- function(t, state, parms) {
  with(as.list(c(state, parms)), {
    dS <- -a * S * I
    dI <- a * S * I - beta * I
    dR <- beta * I
    list(c(dS, dI, dR))
  })
}

times <- seq(0, 160, by=0.1)
out <- ode(y=init, times=times, func=sir, parms=c(a=a, beta=beta))
plot(out[, 'time'], out[, 'I'], type='l', xlab='Zeit', ylab='I(t)')
```

mit dem plot:



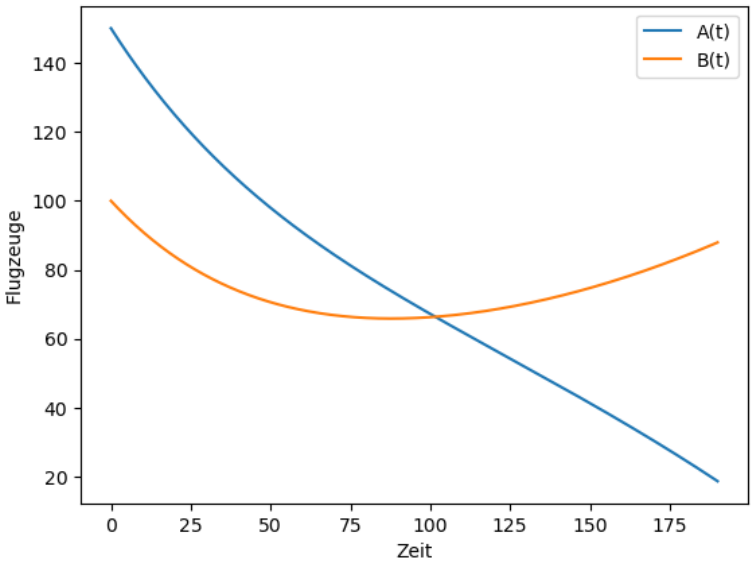
Ich persöhnlich finde die python plots meistens etwas schöner aber in diesem beispiel liefern sie das exact gleiche.

Exercise 2.

In dieser Aufgabe soll das Lanchester Combat Modell simuliert werden, welches Flugzeugkämpfe im ersten Weltkrieg simulieren sollte. Ich habe hier zwei mögliche Kämpfe simuliert deshalb sind die ausgaben aus den beiden scripts natürlich nicht gleich. Python code:

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

# Parameter
kA, kB = 0.01, 0.015
r1, r2 = 0.8, 1.2
b1, b2 = 0.005, 0.007
A0, B0 = 150, 100

def lanchester(y, t, kA, kB, r1, r2, b1, b2):
    A, B = y
    dA = -kB * B + r1 - b1 * A
    dB = -kA * A + r2 - b2 * B
    return [dA, dB]

t = np.linspace(0, 50, 500)
sol = odeint(lanchester, [A0, B0], t, args=(kA, kB, r1, r2, b1, b2))

plt.plot(t, sol[:,0], label='A(t)')
plt.plot(t, sol[:,1], label='B(t)')
plt.xlabel('Zeit')
plt.ylabel('Flugzeuge')
plt.legend();
plt.show()
```

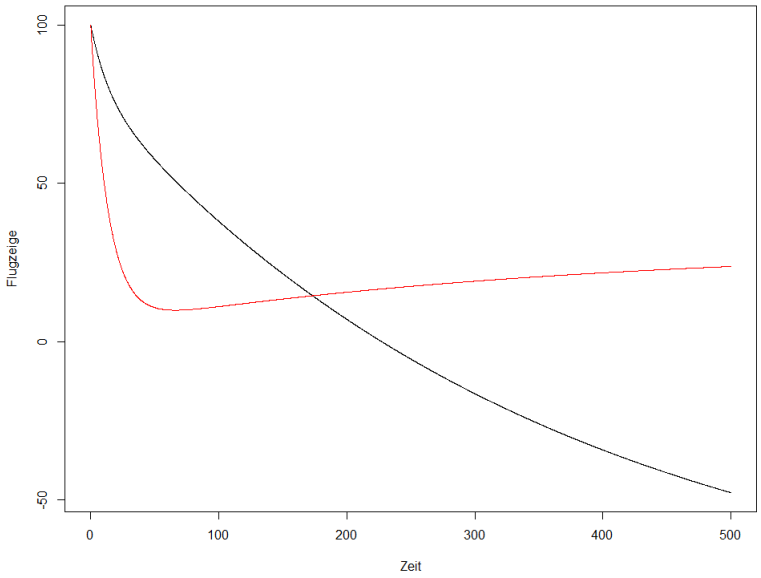
daraus entsteht dann dieser Plot:



In diesem Kampfbeispiel habe ich eine Armee A und eine Armee B gegeneinander kämpfen lassen. A startet mit einer größeren Armee aber B hat etwas bessere Flugzeuge und kann diese etwas schneller produzieren. wenn der Kampf lang genug geht wird B gewinnen aber wenn der kampf nur kurz anhält z.B. weil eine wichitge Stadt eingenommen wird oder weil aus einem anderen Grund A den Krieg gewinnt haben sie sogar immer noch mehr Flugzeuge.
R code:

```r
if (!require('deSolve')) {
  install.packages('deSolve')
}
library(deSolve)
kA <- 0.01; kB <- 0.015
r1 <- 0.0; r2 <- 1.2
b1 <- 0.005; b2 <- 0.07
init <- c(A=100, B=100)

lanchester <- function(t, state, parms) {
  with(as.list(c(state, parms)), {
    dA <- -kB * B + r1 - b1 * A
    dB <- -kA * A + r2 - b2 * B
    list(c(dA, dB))
  })
}

times <- seq(0, 500, by=0.1)
out <- ode(y=init, times=times, func=lanchester,
           parms=c(kA=kA, kB=kB, r1=r1, r2=r2, b1=b1, b2=b2))
plot(out[, 'time'], out[, 'A'], type='l', xlab='Zeit', ylab='Flugzeige')
lines(out[, 'time'], out[, 'B'], col='red')
```

Plot:



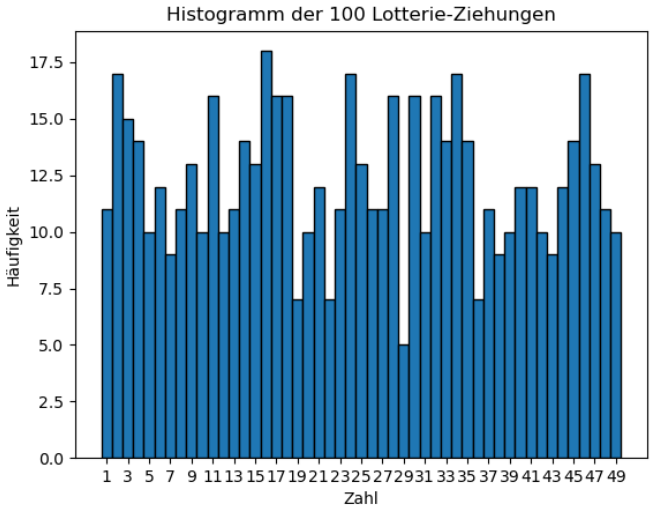
In diesem Krieg habe ich den beiden Armeen gleich viele start Flugzeuge gegeben aber A hat sehr viel bessere Piloten und deshalb eine sehr viel kleinere verschleißrate dafür hat B eine schnellere Producktion als A. Das führt dazu das B zwar früh viele Fleugzeuge in der simulation verliert aber ab einer bestimmten rate an Flugzeugen self replacing ist und so einfach durch brute force den Krieg gewinnt.

Exercise 3.

In dieser Aufgabe soll eine Lotto zeihung simuliert werden dafür soll numpy benutzt werden:

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

# Einzelne Ziehung von 6 Zahlen
zahlen = np.random.choice(np.arange(1,50), size=6, replace=False)
print("Gezogene Zahlen:", np.sort(zahlen))

# 100 Ziehungen und Histogramm
n_draws = 100
# Für jede Ziehung einzeln ohne replacement
draws = np.array([np.random.choice(np.arange(1,50), size=6, replace=False) for _ in range(n_draws)])
numbers = draws.flatten()

plt.hist(numbers, bins=np.arange(1,51)-0.5, edgecolor='black')
plt.xlabel('Zahl')
plt.ylabel('Häufigkeit')
plt.title(f'Histogramm der {n_draws} Lotterie-Ziehungen')
plt.xticks(np.arange(1,50,2))
plt.show()
```

Ich habe das ganze für 100 verschidene Ziehungen auf geplottet:



Github

Wie immer sind alle meine benutzten Dateien auf meinem Github zu finden.