Laboratorium 09

Równania różniczkowe zwyczajne

Iga Antonik, Helena Szczepanowska

Zadanie 1.

Przedstaw każde z poniższych równań różniczkowych zwyczajnych jako równoważny układ równań pierwszego rzędu (ang. first-order system of ODEs):

- a) równanie Van der Pol'a: $y'' = y'(1 - y^2) - y$
- b) równanie Blasiusa: $y''' = -yy''$
- c) II zasada dynamiki Newtona dla problemu dwóch ciał:

$$y_1'' = -GMy_1/(y_1^2 + y_2^2)^{3/2}$$

$$y_2'' = -GMy_2/(y_1^2 + y_2^2)^{3/2}$$

Rozwiązanie

a) równanie Van der Pol'a:

Podstawienia:

$$y_1 = y$$

$$y_2 = y'$$

Układ równań:

$$y_1' = y_2$$

$$y_2' = y_2(1 - y_1^2) - y_1$$

b) równanie Blasius'a:

Podstawienia:

$$y_1 = y$$

$$y_2 = y'$$

$$y_3 = y''$$

Układ równań:

$$y_1' = y_2$$

$$y_2' = y_3$$

$$y_3' = -y_1y_3$$

c) II zasada dynamiki Newton'a dla problemu dwóch ciała:

Podstawienie:

$$y_3 = y_1$$

$$y_4 = y_2$$

$$y_5 = y_1'$$

$$y_6 = y_2'$$

Układ równań:

$$y_3{}' = y_5$$

$$y_4{}' = y_6$$

$$y_5{}' = \frac{-GM \cdot y_3}{(y_3^2 + y_4^2)^{3/2}}$$

$$y_6{}' = \frac{-GM \cdot y_4}{(y_3^2 + y_4^2)^{3/2}}$$

Zadanie 2.

Dane jest równanie różniczkowe zwyczajne $y' = -5y$ z warunkiem początkowym $y(0) = 1$. Równanie rozwiązujemy numerycznie z krokiem $h = 0.5$.

- a) Analityczna stabilność. Wyjaśnij, czy rozwiązania powyższego równania są stabilne?
- b) Numeryczna stabilność. Wyjaśnij, czy metoda Euler'a jest stabilna dla tego równania z użytym krokiem h?
- c) Oblicz numerycznie wartości przybliżonego rozwiązania dla $t = 0.5$ metodą Euler'a.
- d) Wyjaśnij, czy niejawna metoda Euler'a jest stabilna dla tego równania z użytym krokiem h?
- e) Oblicz numerycznie wartości przybliżonego rozwiązania dla $t = 0.5$ niejawną metodą Euler'a.

Rozwiązanie

a)

Rozwiązania równania $y' = -5y$ są analitycznie stabilne. Dla warunku początkowego $y(0) = 1$ rozwiązanie to $y = e^{-5t}$

$$\lim_{t->\infty} e^{-5t} = 0$$

Poniewaz rozwiazanie zbiega do zera, gdy czas t rosnie, mozemy stwierdzic, ze rozwiazania tego rownania sa stabilne w sensie analitycznym.

Biblioteki

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Obliczanie numerycznej stabilności

Warunek dla jawnej metody Eulera

$$|1 + h\lambda| < 1$$

Warunek dla niejawnej metody Eulera

$$\left|\frac{1}{1 - h\lambda}\right| < 1$$

In [ ]:
```
lambd = -5
h = 0.5

explicit_euler_amp_factor = lambda lambd,h: 1 + lambd*h
```

```
implicit_euler_amp_factor = lambda lambd,h: 1/(1-lambd*h)

print("Amplification factors:")
print("explicit euler method: ",round(explicit_euler_amp_factor(lambd,h),2),
       "\nimplicit euler method: ",round(implicit_euler_amp_factor(lambd,h),2))
```

```
Amplification factors:
explicit euler method:  -1.5
implicit euler method:  0.29
```

b)

Jawna metoda Eulera nie jest numerycznie stabilna, poniewaz wartość bezwzględna z czynnika wzmocnienia nie jest mniejsa od 1.

$$|-1.5| > 1$$

d)

Niejawna metoda Eulera jest numerycznie stablina, poniewaz wartość bezwzględna z czynnika wzmocnienia jest mniejsa od 1.

$$|0.29| < 1$$

Obliczanie jawną metodą Eulera

In [ ]:
```
n=5

t_values = np.arange(0, n + h, h)
y_values = np.zeros(len(t_values))
y_values[0] = 1
for i in range(1, len(t_values)):
        y_values[i] = y_values[i-1]*(1+h*lambd)


print(f"t={t_values[1]} y={y_values[1]}")
```

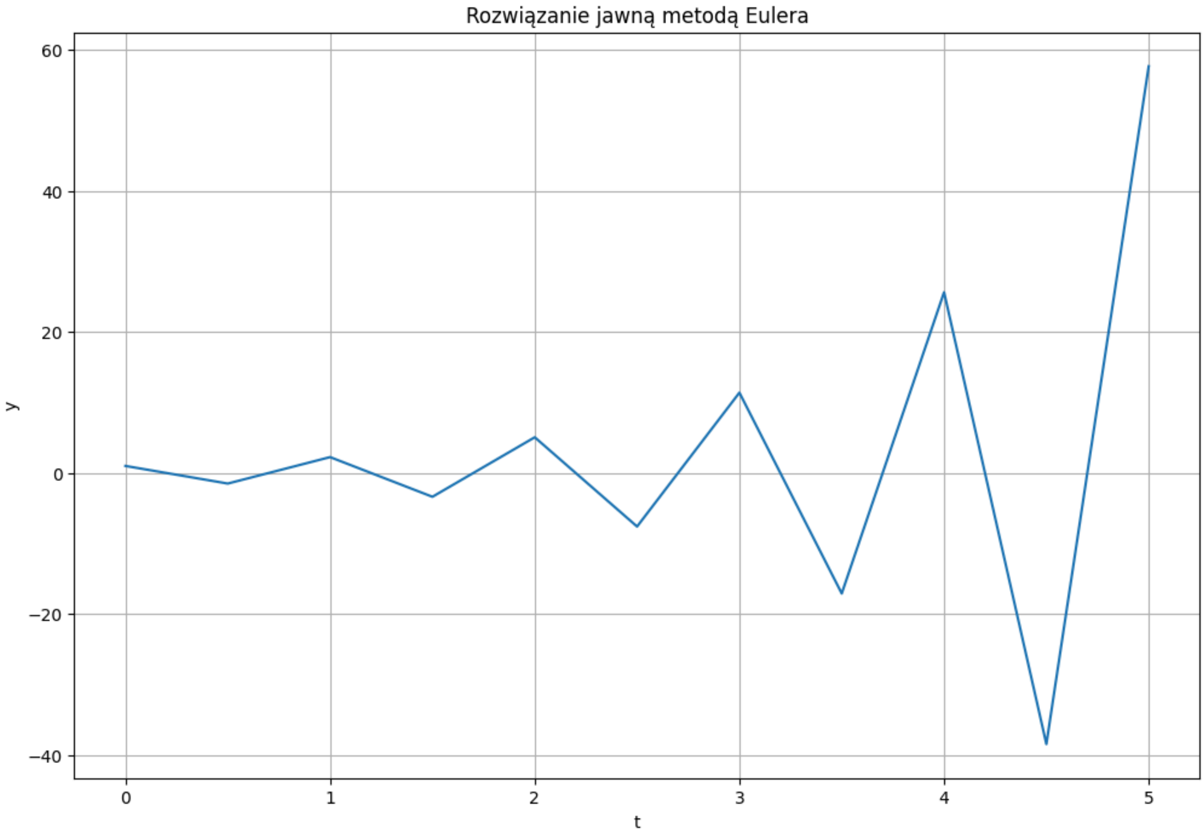
```
t=0.5 y=-1.5
```

c)

Wartość przyblizonego rozwiazania dla $t = 0.5$ wynosi $y = -1.5$

Wykres rozwiązanie uzyskanego tą metodą

In [ ]:
```
plt.figure(figsize=(12, 8))
plt.title("Rozwiązanie jawną metodą Eulera")
plt.plot(t_values, y_values)
plt.xlabel('t')
plt.ylabel('y')
plt.grid()
plt.show()
```



Na powyszym wykresie dobrze widać, ze jawna metoda eulera jest niestabilna.

Obliczanie niejawną metodą Eulera

In [ ]:
```
t_values_imp = np.arange(0, n + h, h)
y_values_imp  = np.zeros(len(t_values_imp))
y_values_imp[0] = 1
for i in range(1, len(t_values_imp)):
        y_values_imp[i] = y_values_imp[i-1]/(1-lambd*h)

print(f" t={t_values_imp[1]} y={round(y_values_imp[1],2)}")
```

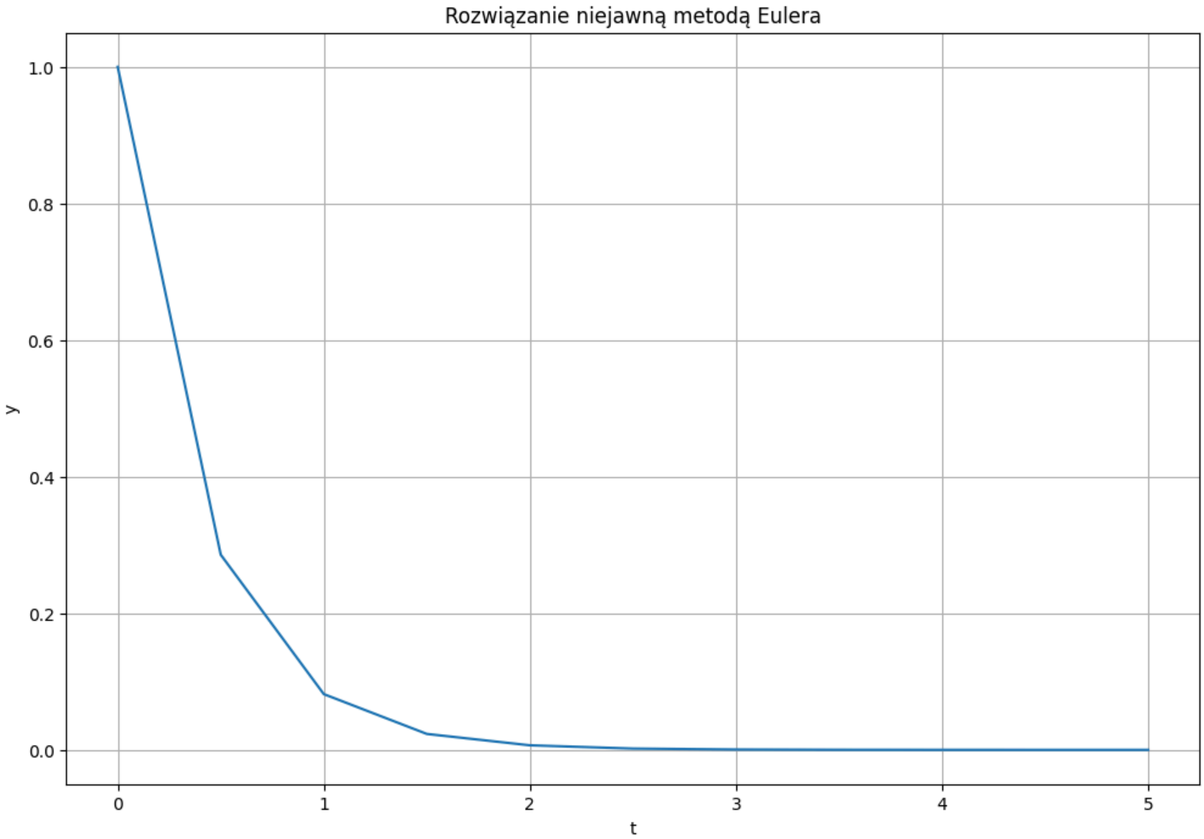
```
t=0.5 y=0.29
```

c)

Wartość przyblizonego rozwiazania dla $t = 0.5$ wynosi $y = 0.29$

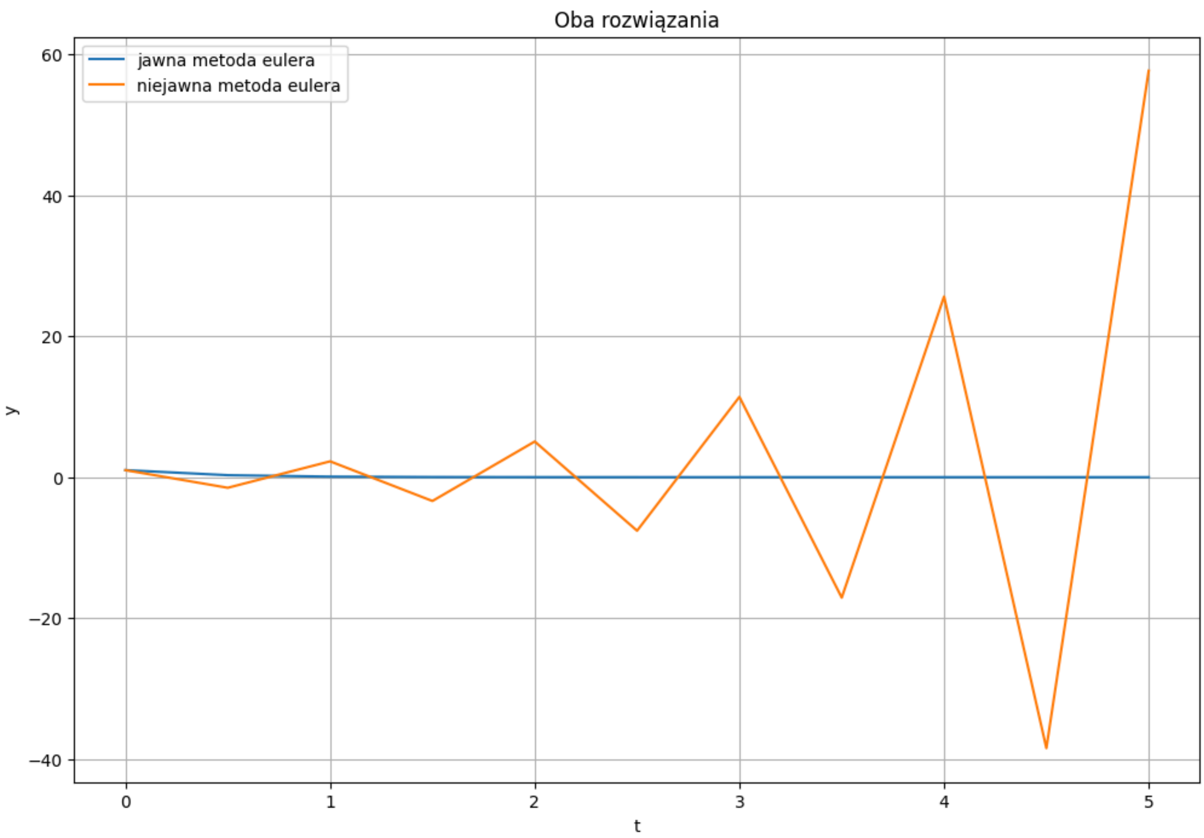
Wykres rozwiązanie uzyskanego tą metodą

In [ ]:
```
plt.figure(figsize=(12, 8))
plt.title("Rozwiązanie niejawną metodą Eulera")
plt.plot(t_values_imp, y_values_imp)
plt.xlabel('t')
plt.ylabel('y')
plt.grid()
plt.show()
```



Wykres obu rozwiązań razem

In [ ]:
```
plt.figure(figsize=(12, 8))
plt.title("Oba rozwiązania")
plt.plot(t_values_imp, y_values_imp, label='jawna metoda eulera')
plt.plot(t_values, y_values, label='niejawna metoda eulera')
plt.xlabel('t')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.show()
```



Zadanie 3.

Model Kermack'a-McKendrick'a przebiegu epidemii w populacji opisany jest układem równań różniczkowych:

$$S' = -\frac{\beta}{N} IS$$

$$I' = \frac{\beta}{N} IS - \gamma I$$

$$R' = \gamma I$$

gdzie S reprezentuje liczbę osób zdrowych, podatnych na zainfekowanie,

I reprezentuje liczbę osób zainfekowanych i roznoszących infekcję,

R reprezentuje liczbę osób ozdrowiałych,

N reprezentuje liczbę osób w populacji.

Parametr β reprezentuje współczynnik zakaźności (ang. transmission rate). Parametr γ reprezentuje współczynnik wyzdrowień (ang. recovery rate). Wartość $1/\gamma$ reprezentuje średni czas choroby.

Założenia modelu:

- Przyrost liczby osób zakażonych jest proporcjonalny do liczby osób zakażonych oraz do liczby osób podatnych.
- Przyrost liczby osób odppornych lub zmarłych jest wprost proporcjonalny do liczby aktualnie chorych.
- Okres inkubacji choroby jest zaniedbywalnie krótki.
- Populacja jest wymieszana.

Jako wartości początkowe ustal:

$$S(0) = 762, I(0) = 1, R(0) = 0$$

Przyjmij też $N = S(0) + I(0) + R(0) = 763$ oraz $\beta = 1$. Zakładając, że średni czas trwania grypy wynosi $1/\gamma = 7$ dni, przyjmij $\gamma = 1/7$.

Całkując od $t = 0$ do $t = 14$ z krokiem 0.2, rozwiąż powyższy układ równań:

- jawną metodą Eulera

$$y_{k+1} = y_k + h_k f(t_k, y_k)$$

- niejawną metodą Eulera

$$y_{k+1} = y_k + h_k f(t_{k+1}, y_{k+1})$$

- metodą Rungego-Kutty czwartego rzędu (RK4)

$$y_{k+1} = y_k + \frac{h_k}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

  , gdzie:

$$k_1 = f(t_k, y_k)$$
$$k_2 = f(t_k + h_k/2, y_k + h_k k_1/2)$$
$$k_3 = f(t_k + h_k/2, y_k + h_k k_2/2)$$
$$k_4 = f(t_k + h_k, y_k + h_k k_3)$$

  Wykonaj nastepujące wykresy:

- Dla każdej metody przedstaw na wspólnym rysunku wykresy komponentów rozwiązania (S, I, R) jako funkcje t (3 wykresy).

- Na wspólnym rysunku przedstaw wykresy funkcji $S(t) + I(t) + R(t)$ znalezione przez każdą metodę (1 wykres). Czy niezmiennik $S(t) + I(t) + R(t) \equiv N$ jest zachowany?

Wiemy, że liczba osób zakażonych w pewnej szkole kształtowała się następująco:

Dzień	t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Zakażeni	I	1	3	6	25	73	222	294	258	237	191	125	69	27	11	4

Wybierz jedną z powyższych metod numerycznych i oszacuj prawdziwe wartości współczynników $\theta = [\beta, \gamma]$. W tym celu wykonaj minimalizację funkcji kosztu. Jako funkcję kosztu wykorzystaj sumę kwadratów reszt (ang. residual sum of squares):

$$L(\theta) = (I_i - \hat{} I_i)_2$$

gdzie I_i oznacza prawdziwą liczbę zakażonych, a $\hat{} I_i$ oznacza liczbę zakażonych wyznaczonych metodą numeryczną. Ponieważ nie znamy gradientu $\nabla\theta L(\theta)$, do minimalizacji wykorzystaj metodę Neldera-Meada, która nie wymaga informacji o gradiencie. Powtórz obliczenia, tym razem jako funkcję kosztu wykorzystując:

$$L(\theta) = -\sum_{i=0}^{T} I_i ln \hat{} I_i + \sum_{i=0}^{T} \hat{} I_i$$

Ile wynosił współczynnik reprodukcji $R_0 = \beta/\gamma$ w każdym przypadku?

Rozwiązanie

Biblioteki

In [ ]:
```
import numpy as np
import scipy
import matplotlib.pyplot as plt
from scipy.optimize import minimize
from scipy.optimize import fsolve
```

Dane

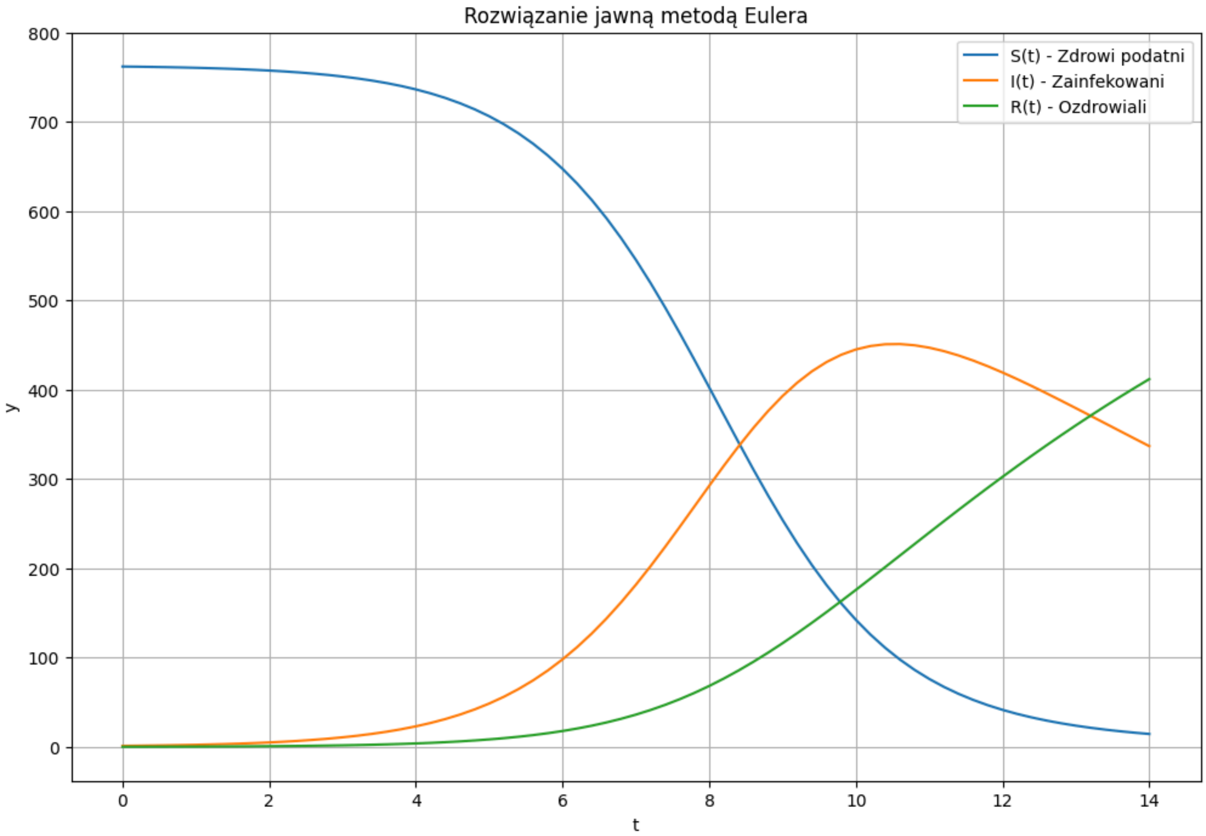
In [ ]:
```
S = 762
I = 1
R = 0

N = S + I + R
beta = 1
gamma = 1/7
y0 = [S,I,R]
h = 0.2
t_values = np.arange(0, 14.2, h)

S_values_explicit = np.zeros(len(t_values))
I_values_explicit = np.zeros(len(t_values))
R_values_explicit = np.zeros(len(t_values))

S_values_explicit[0]= S
I_values_explicit[0]= I
R_values_explicit[0]= R

# Funkcje pochodnych
def dS_dt(S, I, R):
    return -beta * S * I / N

def dI_dt(S, I, R):
    return beta * S * I / N - gamma * I

def dR_dt(S, I, R):
    return gamma * I
```

Jawna metoda Eulera

In [ ]:
```
k = 1
for t in t_values:
    if t==0: continue
    R_old = R
    S_old = S
    I_old = I
    R = R_old + h*gamma*I_old
    S = S_old - h*beta*I_old*S_old/N
    I = I_old + h*beta*I_old*S_old/N - h*gamma*I_old
    S_values_explicit[k] = S
    I_values_explicit[k] = I
    R_values_explicit[k] = R
    k+=1
```

Wykres

In [ ]:
```
plt.figure(figsize=(12, 8))
plt.title("Rozwiązanie jawną metodą Eulera")
plt.plot(t_values, S_values_explicit, label='S(t) - Zdrowi podatni')
plt.plot(t_values, I_values_explicit, label='I(t) - Zainfekowani')
plt.plot(t_values, R_values_explicit, label='R(t) - Ozdrowiali')
plt.xlabel('t')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.show()
```

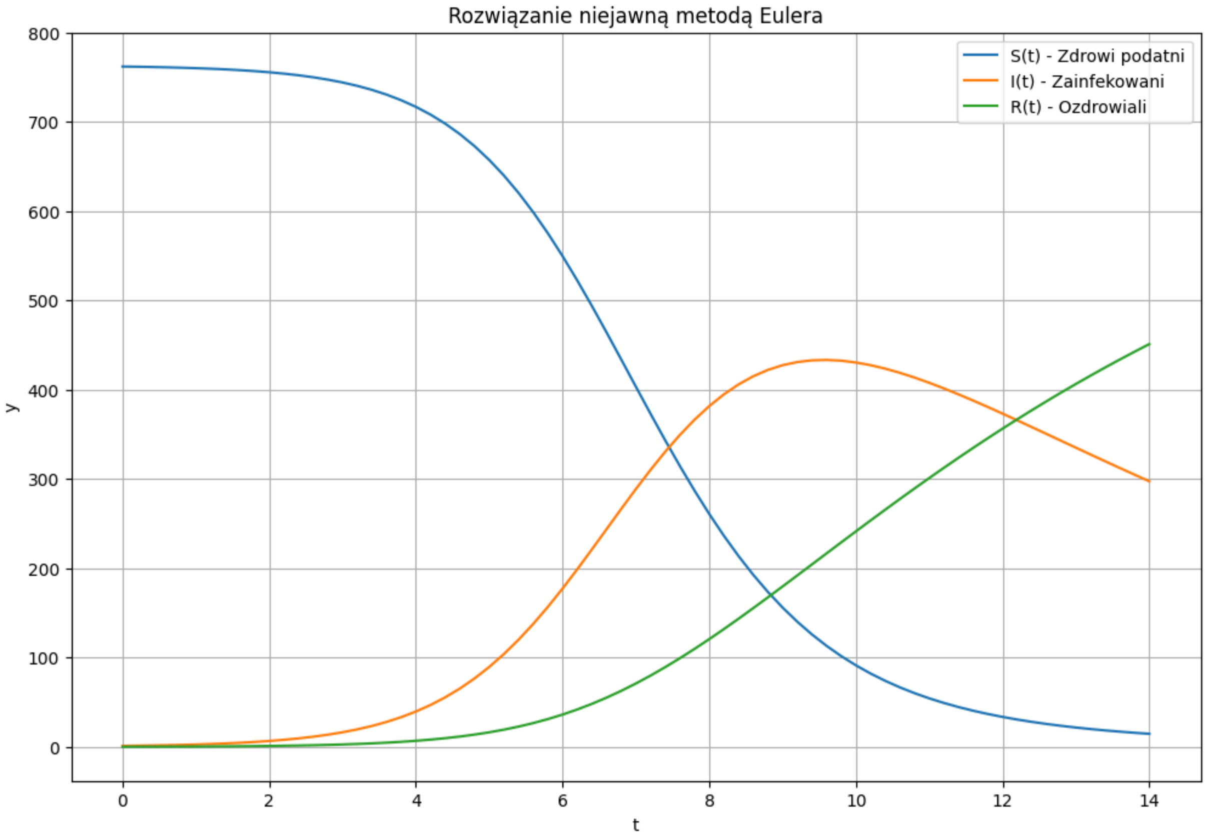


Niejawna metoda Eulera

In [ ]:
```
def f(t, y):
    S, I, R = y
    dS = -beta * I * S / N
    dI = beta * I * S / N - gamma * I
    dR = gamma * I
    return np.array([dS, dI, dR])

def implicit_euler_method(y0, h, t_values):
    y_values = np.zeros((len(t_values), len(y0)))
    y_values[0] = y0

    for k in range(1, len(t_values)):
        t_next = t_values[k]
        y_prev = y_values[k-1]

        def func(y_next):
            return y_next - y_prev - h * f(t_next, y_next)

        y_next = fsolve(func, y_prev)
        y_values[k] = y_next

    return y_values

y_implicit = implicit_euler_method(y0, h, t_values)
```

Wykres

In [ ]:
```
plt.figure(figsize=(12, 8))
plt.title("Rozwiązanie niejawną metodą Eulera")
plt.plot(t_values, y_implicit[:, 0],label='S(t) - Zdrowi podatni')
plt.plot(t_values, y_implicit[:, 1],label='I(t) - Zainfekowani')
plt.plot(t_values, y_implicit[:, 2],label='R(t) - Ozdrowiali')
plt.xlabel('t')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.show()
```

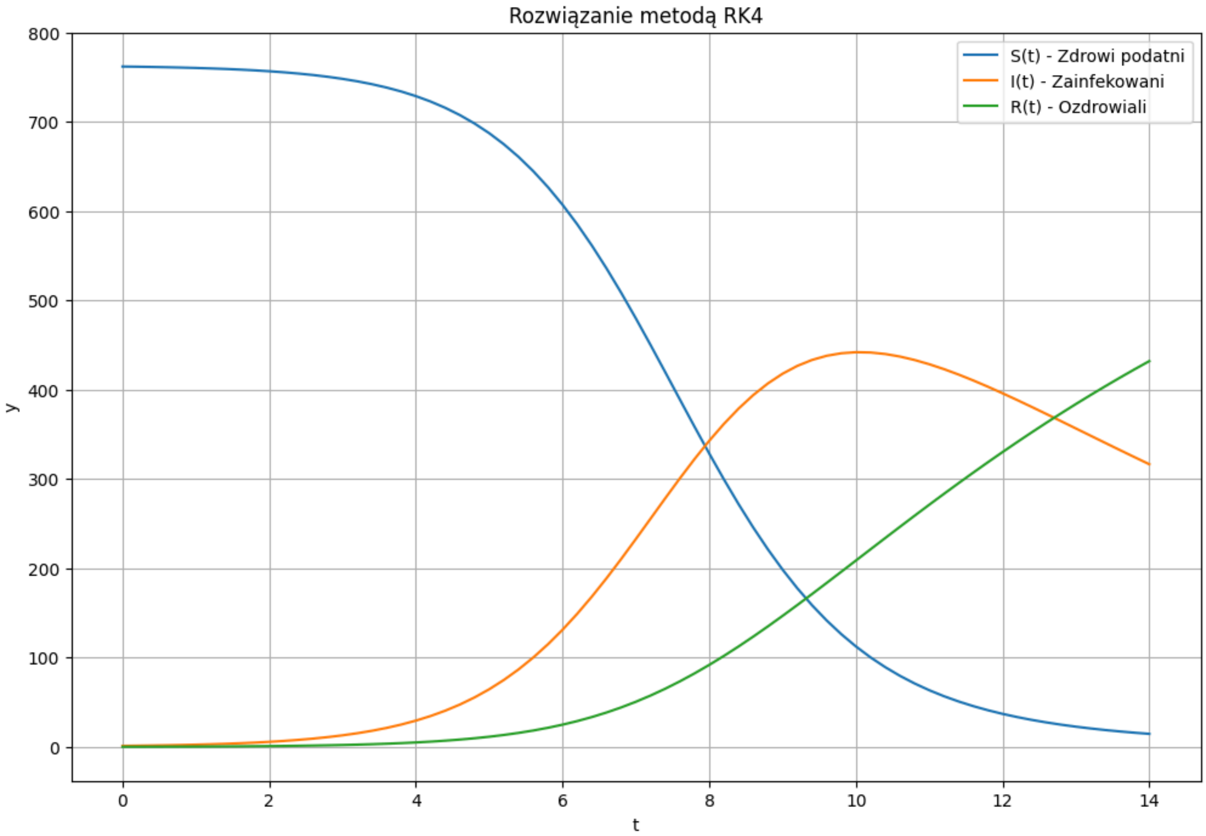


Metoda Rungego-Kutty czwartego rzędu

In [ ]:
```
def sir_model(t, y, beta, gamma):
    S, I, R = y
    dSdt = np.array(-beta / N * I * S)
    dIdt = np.array(beta / N * I * S - gamma * I)
    dRdt = np.array(gamma * I)
    return np.array([dSdt, dIdt, dRdt])

def rk4_method(f, t, y0, h, beta, gamma):
    n = len(t)
    y = np.zeros((n, len(y0)))
    y[0] = y0

    for i in range(n - 1):
        k1 = np.array(f(t[i], y[i], beta, gamma))
        k2 = np.array(f(t[i] + h/2, y[i] + h*k1/2, beta, gamma))
        k3 = np.array(f(t[i] + h/2, y[i] + h*k2/2, beta, gamma))
        k4 = np.array(f(t[i] + h, y[i] + h*k3, beta, gamma))
        y[i + 1] = y[i] + (h / 6) * (k1 + 2*k2 + 2*k3 + k4)

    return y


y_rk4 = rk4_method(sir_model, t_values, y0, 0.2, beta, gamma)
```

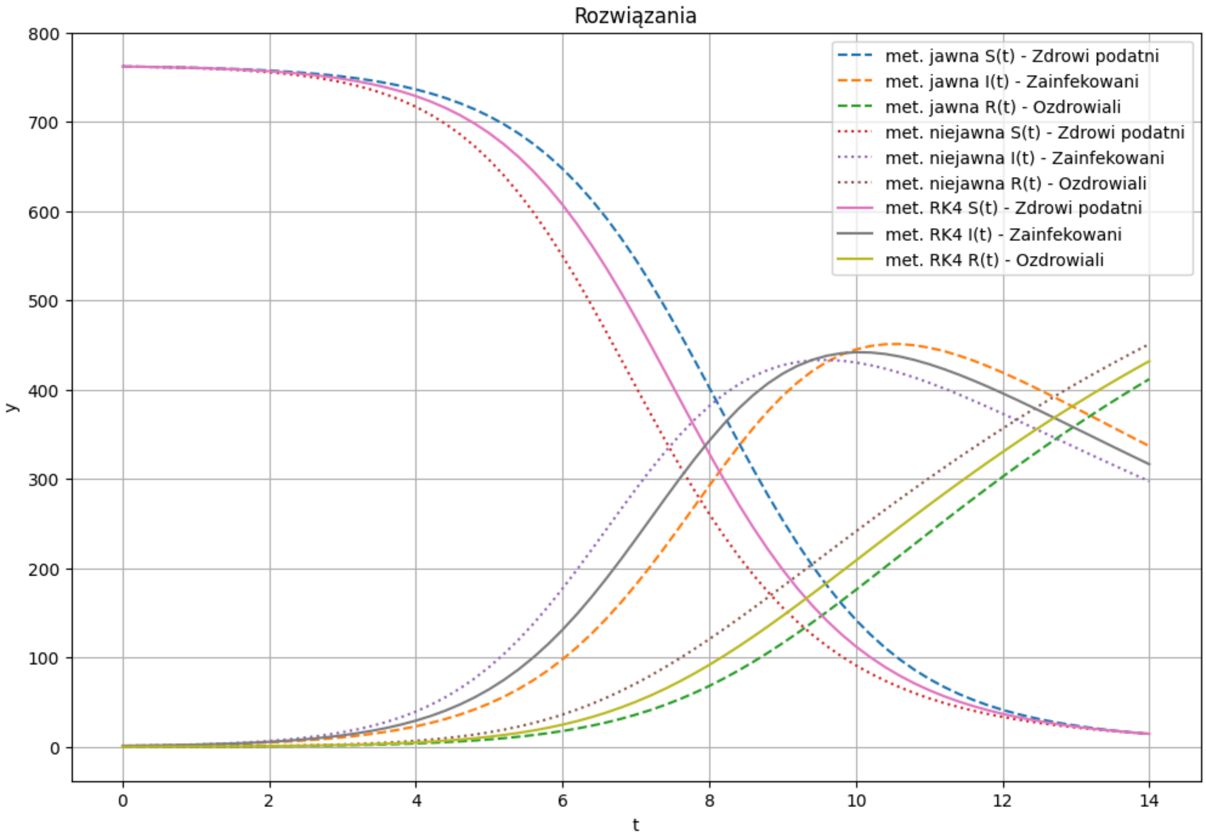
Wykres

In [ ]:
```
plt.figure(figsize=(12, 8))
plt.title("Rozwiązanie metodą RK4")
plt.plot(t_values, y_rk4[:, 0], label='S(t) - Zdrowi podatni')
plt.plot(t_values, y_rk4[:, 1], label='I(t) - Zainfekowani')
plt.plot(t_values, y_rk4[:, 2],  label='R(t) - Ozdrowiali')
plt.xlabel('t')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.show()
```



Wszystkie rozwiązania na jednym wykresie

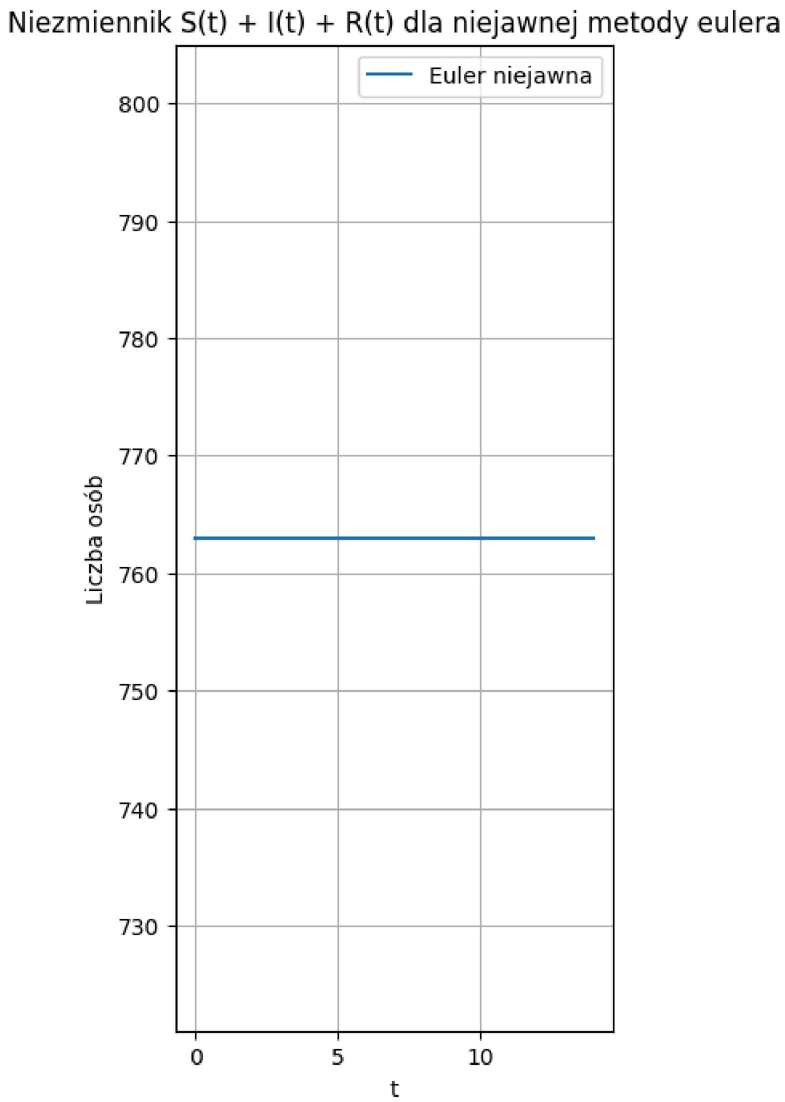
In [ ]:
```
plt.figure(figsize=(12, 8))
plt.title("Rozwiązania")
plt.plot(t_values, S_values_explicit, label='met. jawna S(t) - Zdrowi podatni', linestyle='dash
plt.plot(t_values, I_values_explicit, label='met. jawna I(t) - Zainfekowani', linestyle='dashed
plt.plot(t_values, R_values_explicit, label='met. jawna R(t) - Ozdrowiali', linestyle='dashed')
plt.plot(t_values, y_implicit[:, 0],label='met. niejawna S(t) - Zdrowi podatni', linestyle='dot
plt.plot(t_values, y_implicit[:, 1],label='met. niejawna I(t) - Zainfekowani', linestyle='dotte
plt.plot(t_values, y_implicit[:, 2],label='met. niejawna R(t) - Ozdrowiali', linestyle='dotted'
plt.plot(t_values, y_rk4[:, 0], label='met. RK4 S(t) - Zdrowi podatni')
plt.plot(t_values, y_rk4[:, 1], label='met. RK4 I(t) - Zainfekowani')
plt.plot(t_values, y_rk4[:, 2],  label='met. RK4 R(t) - Ozdrowiali')
plt.xlabel('t')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.show()
```

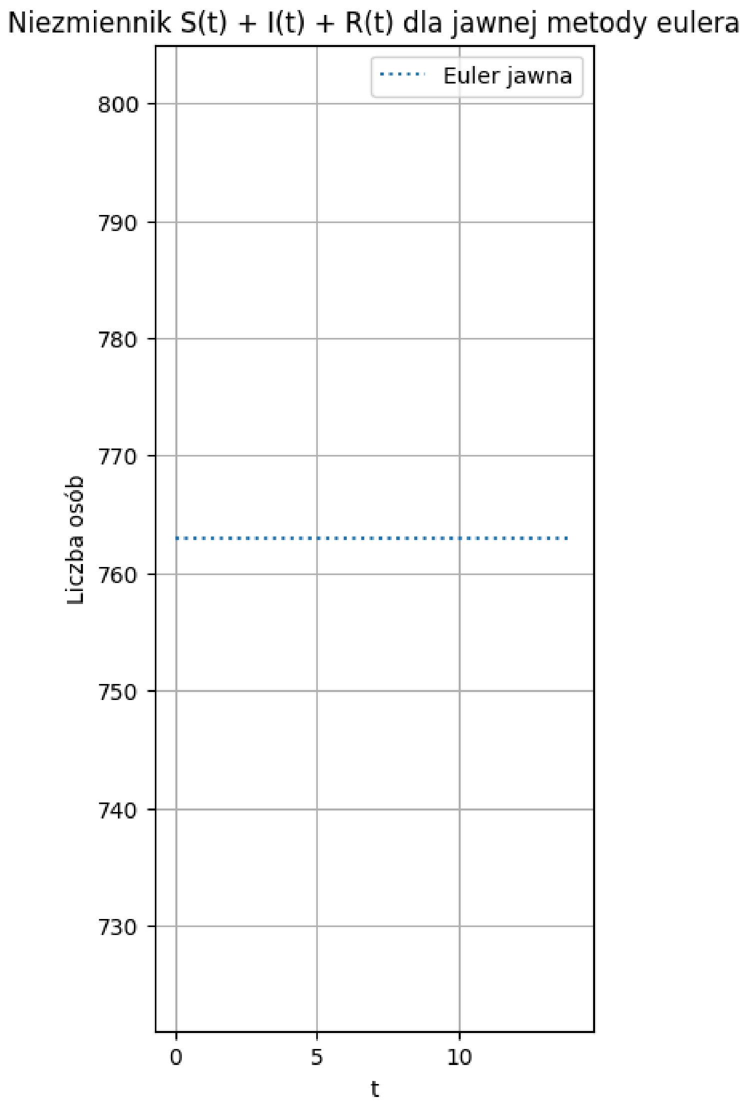


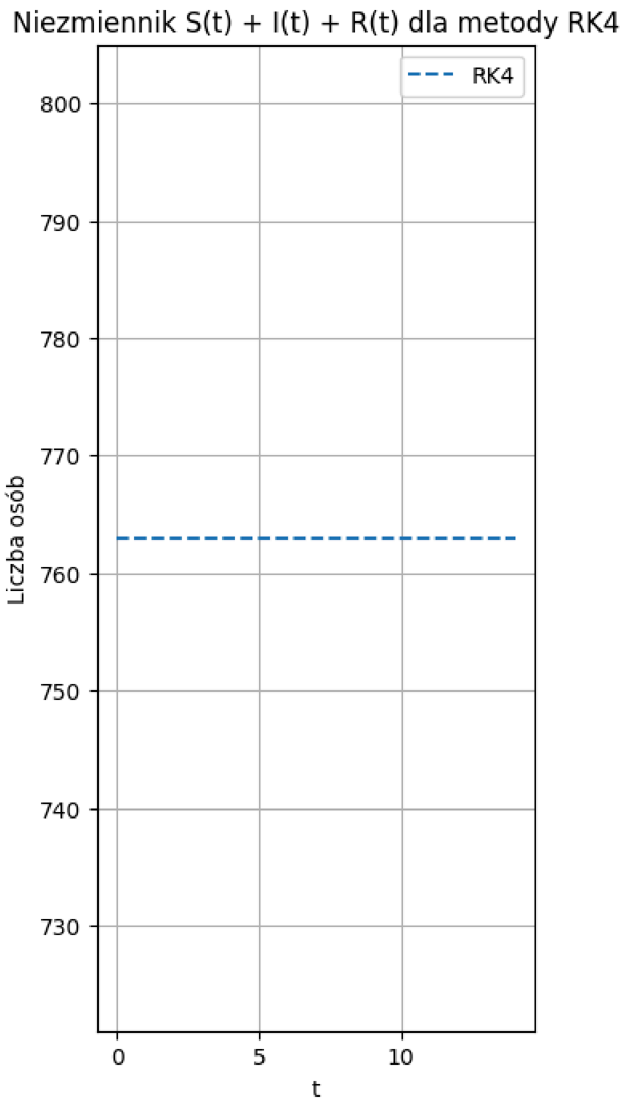
Wyniki uzyskane przy pomocy kazdej z metod są do siebie bardzo zblizone.

Niezmiennik S(t) + I(t) + R(t) dla różnych metod

In [ ]:
```
plt.figure(figsize=(12, 8))
plt.subplot(1,3,1)
plt.plot(t_values, y_implicit[:, 0] + y_implicit[:, 1] + y_implicit[:, 2], label='Euler niejawn
plt.xlabel('t')
plt.ylabel('Liczba osób')
plt.legend()
plt.title('Niezmiennik S(t) + I(t) + R(t) dla niejawnej metody eulera')
plt.grid()

plt.figure(figsize=(12, 8))
plt.subplot(1,3,2)
plt.plot(t_values, S_values_explicit + I_values_explicit + R_values_explicit,linestyle='dotted'
plt.xlabel('t')
plt.ylabel('Liczba osób')
plt.legend()
plt.title('Niezmiennik S(t) + I(t) + R(t) dla jawnej metody eulera')
plt.grid()


plt.figure(figsize=(12, 8))
plt.subplot(1,3,3)
plt.plot(t_values, y_rk4[:, 0] + y_rk4[:, 1] + y_rk4[:, 2], linestyle='dashed',label='RK4')
plt.xlabel('t')
plt.ylabel('Liczba osób')
plt.legend()
plt.title('Niezmiennik S(t) + I(t) + R(t) dla metody RK4')
plt.grid()

plt.show()
```


Niezmiennik S(t) + I(t) + R(t) dla niejawnej metody eulera
Euler niejawna
800
790
780
770
760
750
740
730
Liczba osób
0
5
10
t


Niezmiennik S(t) + I(t) + R(t) dla jawnej metody eulera
Euler jawna
Liczba osób
t
800
790
780
770
760
750
740
730
0
5
10



Jak widać na powyzszym wykresie niezmienniki S(t) + I(t) + R(t) dla kazdej z metod nie zmieniają swojej wartości. Dzięki temu mozemy stwierdzic, ze metody zostaly zaimplementowane poprawnie.

In [ ]:

```
true_infected = np.array([1, 3, 6, 25, 73, 222, 294, 258, 237, 191, 125, 69, 27, 11, 4])
days = np.arange(15)


def cost_function(params, t, true_infected):
    beta, gamma = params
    h = 1
    y_rk4 = rk4_method(sir_model, days, y0, h, beta, gamma)
    I_pred = y_rk4[:,1]
    solution = np.sum((np.array(true_infected) - I_pred) ** 2)
    return solution


def log_likelihood_function(params, t, true_infected):
    beta, gamma = params
    t_span = (0, 14)
    h = 1
    y_rk4 = rk4_method(sir_model, days, y0, h, beta, gamma)
    I_pred = y_rk4[:,1]
    solution = np.sum(-np.array(true_infected * np.log(I_pred)) + I_pred)
    return solution


# options = {'maxiter': 200, 'maxfun': 300, 'disp': False}
initial_guess = [beta, gamma]

result = minimize(cost_function, initial_guess, args=(days, true_infected), method='Nelder-Mead
beta_est, gamma_est = result.x
R0_est = beta_est / gamma_est
```

```
print(f"beta = {round(beta_est,2)}, gamma = {round(gamma_est,2)}, R0 = {round(R0_est,2)}")

result_log_likelihood = minimize(log_likelihood_function, initial_guess, args=(days, true_infec
beta_est_log, gamma_est_log = result_log_likelihood.x
R0_est_log = beta_est_log / gamma_est_log
print(f"beta = {round(beta_est_log,2)}, gamma = {round(gamma_est_log,2)}, R0 = {round(R0_est_lo
```

```
beta = 1.67, gamma = 0.45, R0 = 3.75
beta = 1.69, gamma = 0.48, R0 = 3.52
```

Oto wyznaczone za pomocą obu metod wartości współczynników β,γ:

metoda	β	γ	R_0
suma kwadratów reszt	1.67	0.45	3.75
log-likelihood	1.69	0.48	3.52

Wychodzi na to, ze w obu przypadkach średni czas choroby to trochę ponad 2 dni.

$R_0 > 1$ więc każdy zainfekowany człowiek średnio zaraża więcej niż jedną osobę, co prowadzi do rozprzestrzeniania się epidemii.

Bibliografia

prezentacja QOrdinary differential equations - Marcin Kuta

Michael T. Heath,http://heath.cs.illinois.edu/scicomp/notes/cs450_chapt09.pdf