Lab 2 – Równania różniczkowe drugiego rzędu. Chaos deterministyczny.

Arkadiusz Kurnik, Jan Cichoń

Zadanie 1:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from scipy.special import jn

#ZADANIE_1#

print("ZADANIE 1")

def forced_damped_oscillator(t, y, r, F0, omega, omega0):
    x, v = y
    dxdt = v
    dvdt = (F0 * np.cos(omega * t) - r * v - omega0**2 * x)
    return np.array([dxdt, dvdt])

def euler_method(f, y0, t0, tf, dt, r, F0, omega, omega0):
    t = np.arange(t0, tf, dt)
    y = np.zeros((len(t), len(y0)))
    y[0] = y0
    for i in range(1, len(t)):
        y[i] = y[i-1] + dt * f(t[i-1], y[i-1], r, F0, omega, omega0)
    return t, y

def analytic_solution(t, x0, u0, r, F0, omega, omega0):
    if omega0 != omega:
        A = (F0 / (omega0**2 - omega**2))
        C1 = x0 - A
        C2 = (u0 + r * x0) / omega0
        x_p = A * np.cos(omega * t)
    else:
        A = F0 / (2 * omega0)
        C1 = x0
        C2 = (u0 + r * x0) / omega0
        x_p = A * t * np.sin(omega0 * t)

    x_h = C1 * np.cos(omega0 * t) + C2 * np.sin(omega0 * t)

    return x_h + x_p, x_h, x_p

```

```python
cases = [
    (0, 0, 0, 1, 7, 5, 20),
    (0, 1, 1, 1, 7, 5, 20),
    (0, 1, 1, 1, 5.1, 5, 100),
    (2, 1, 1, 1, 7, 5, 40),
    (2, 0, 0, 1, 7, 5, 40),
    (2, 1, 1, 1, 5.1, 5, 40)
]

t0, dt = 0.0, 0.001

for r, x0, u0, F0, omega, omega0, tf in cases:
    y0 = [x0, u0]
    t, y_numerical = euler_method(forced_damped_oscillator, y0, t0, tf, dt, r, F0, omega, omega0)
    x_analytic, x_h, x_p = analytic_solution(t, x0, u0, r, F0, omega, omega0)
    error = np.sum(np.abs(y_numerical[:, 0] - x_analytic))

    fig, axs = plt.subplots(2, 1, figsize=(10, 8))

    axs[0].plot(t, y_numerical[:, 0], label='Numeryczne', alpha=0.7)
    axs[0].plot(t, x_analytic, label='Analityczne', linestyle='dashed')
    axs[0].set_xlabel('t')
    axs[0].set_ylabel('x(t)')
    axs[0].set_title(f'r={r}, x0={x0}, u0={u0}, F0={F0}, omega={omega}, omega0={omega0}')
    axs[0].legend()
    axs[0].grid(True)
    axs[0].text(0.05, 0.95, f'Błąd całkowity: {error:.2e}', transform=axs[0].transAxes, fontsize=12,
                verticalalignment='top', bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.5))

    axs[1].plot(t, x_h, label='Homogeniczne')
    axs[1].plot(t, x_p, label='Partykularne', linestyle='dashed')
    axs[1].set_xlabel('t')
    axs[1].set_ylabel('x(t)')
    axs[1].set_title('Rozkład na składową homogeniczną i partykularną')
    axs[1].legend()
    axs[1].grid(True)

    plt.tight_layout()
    plt.show()

    print(f'Błąd całkowity dla przypadku r={r}, x0={x0}, u0={u0}, F0={F0}, omega={omega}, omega0={omega0}: {error:.2e}')

```

```python
# Wykres Amplituda w stanie ustalonym vs Częstotliwość względna
r_values = [0.1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 20]
relative_frequencies = np.linspace(0.1, 2.0, 500)
plt.figure(figsize=(10, 5))

for r in r_values:
    amplitudes = []
    for omega_rel in relative_frequencies:
        omega = omega_rel * cases[0][5]
        if cases[0][5] != omega:
            A = cases[0][3] / np.sqrt((cases[0][5]**2 - omega**2)**2 + (r * omega)**2)
        else:
            A = cases[0][3] / (r * omega)
        amplitudes.append(A)
    plt.plot(relative_frequencies, amplitudes, label=f'r={r}')

plt.xlabel('Częstotliwość względna (ω/ω0)')
plt.ylabel('Amplituda')
plt.title('Amplituda w stanie ustalonym vs Częstotliwość względna')
plt.legend()
plt.grid(True)
plt.show()
```

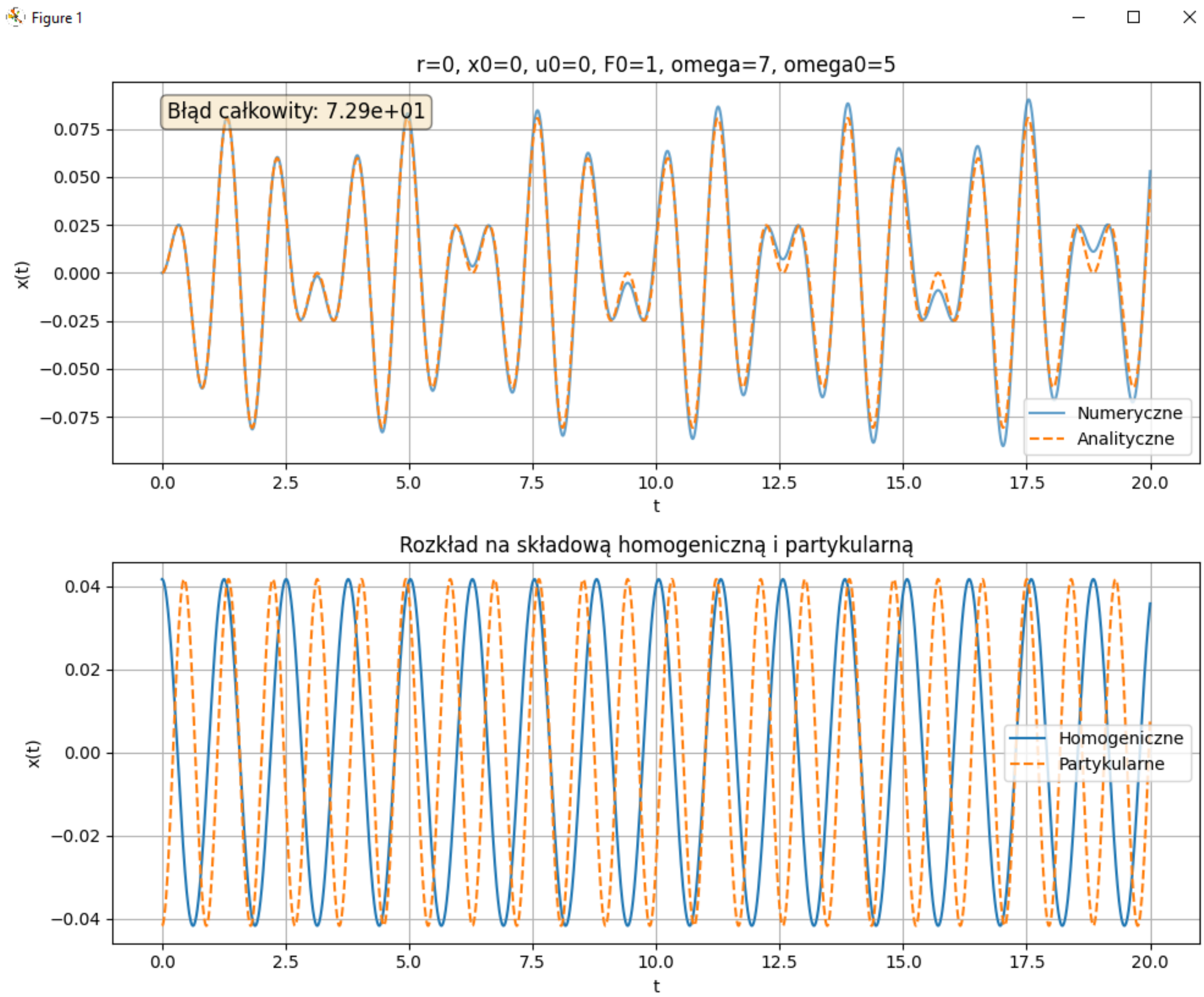


Figure 1
r=0, x0=0, u0=0, F0=1, omega=7, omega0=5
Błąd całkowity: 7.29e+01
x(t)
t
Numeryczne
Analityczne
Rozkład na składową homogeniczną i partykularną
x(t)
t
Homogeniczne
Partykularne

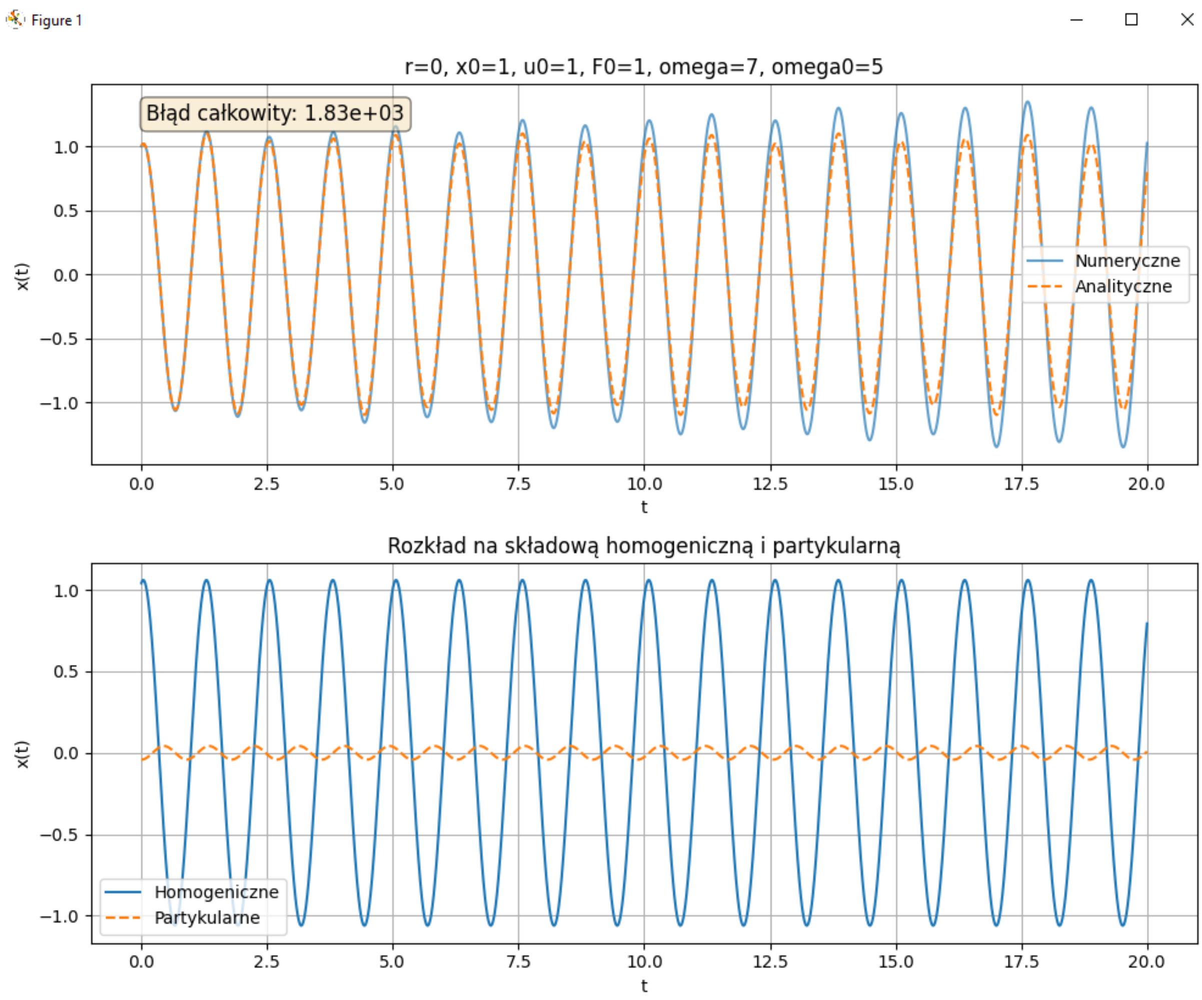


Figure 1
r=0, x0=1, u0=1, F0=1, omega=7, omega0=5
Błąd całkowity: 1.83e+03
Numeryczne
Analityczne
x(t)
t
Rozkład na składową homogeniczną i partykularną
Homogeniczne
Partykularne
x(t)
t

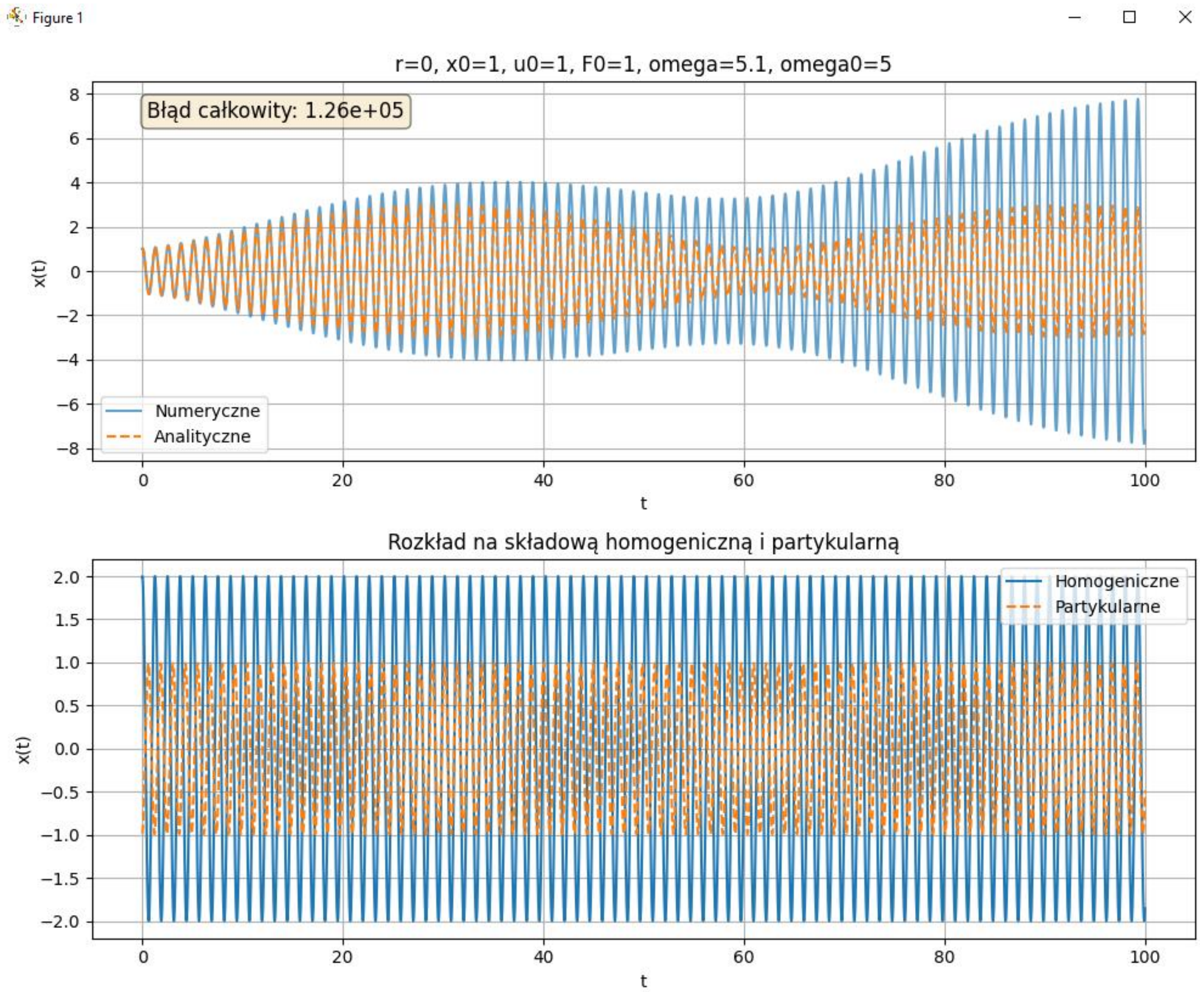


Figure 1
r=0, x0=1, u0=1, F0=1, omega=5.1, omega0=5
Błąd całkowity: 1.26e+05
Numeryczne
Analityczne
x(t)
t
Rozkład na składową homogeniczną i partykularną
Homogeniczne
Partykularne
x(t)
t

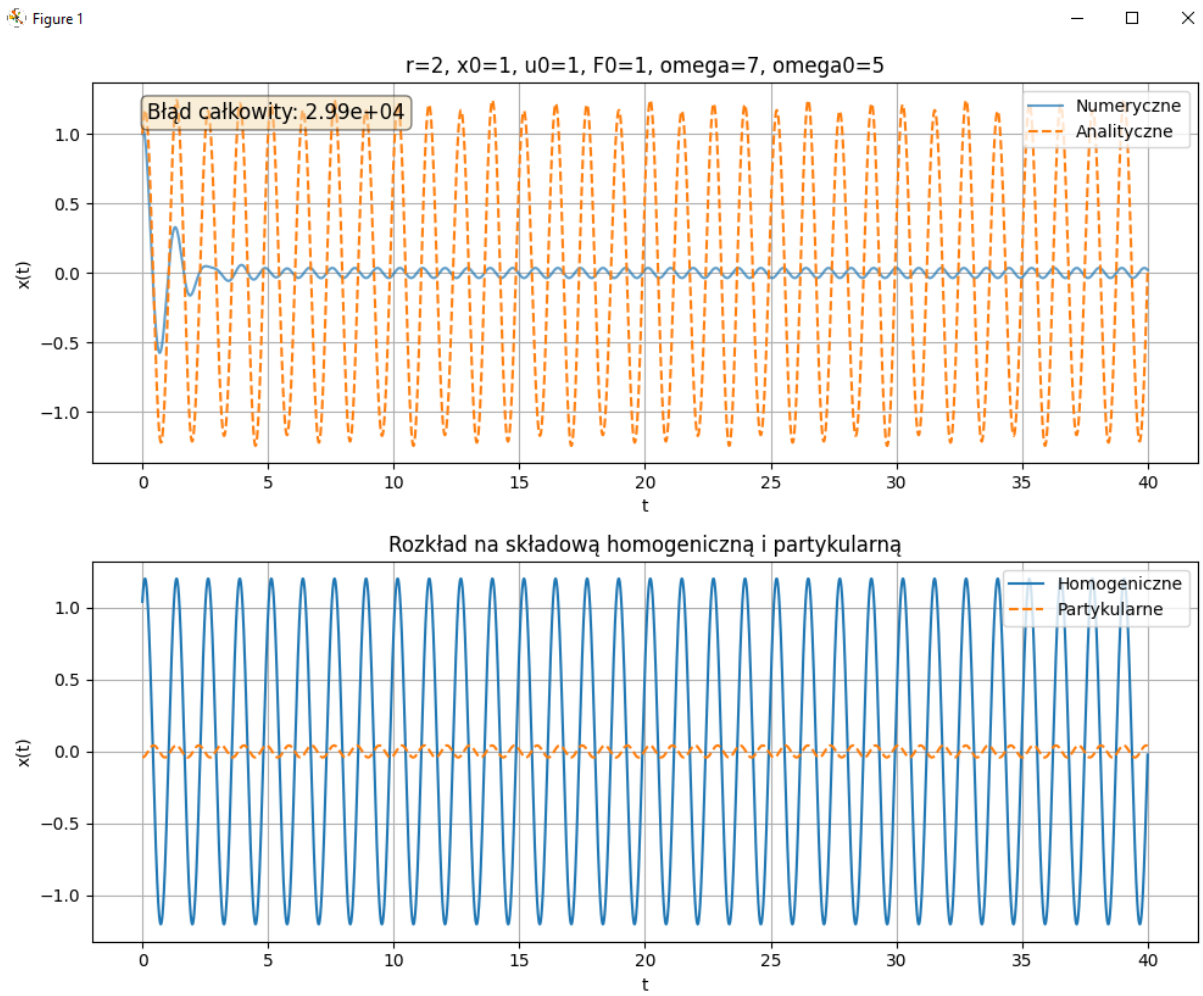


Figure 1
r=2, x0=1, u0=1, F0=1, omega=7, omega0=5
Błąd całkowity: 2.99e+04
Numeryczne
Analityczne
x(t)
t
Rozkład na składową homogeniczną i partykularną
Homogeniczne
Partykularne
x(t)
t

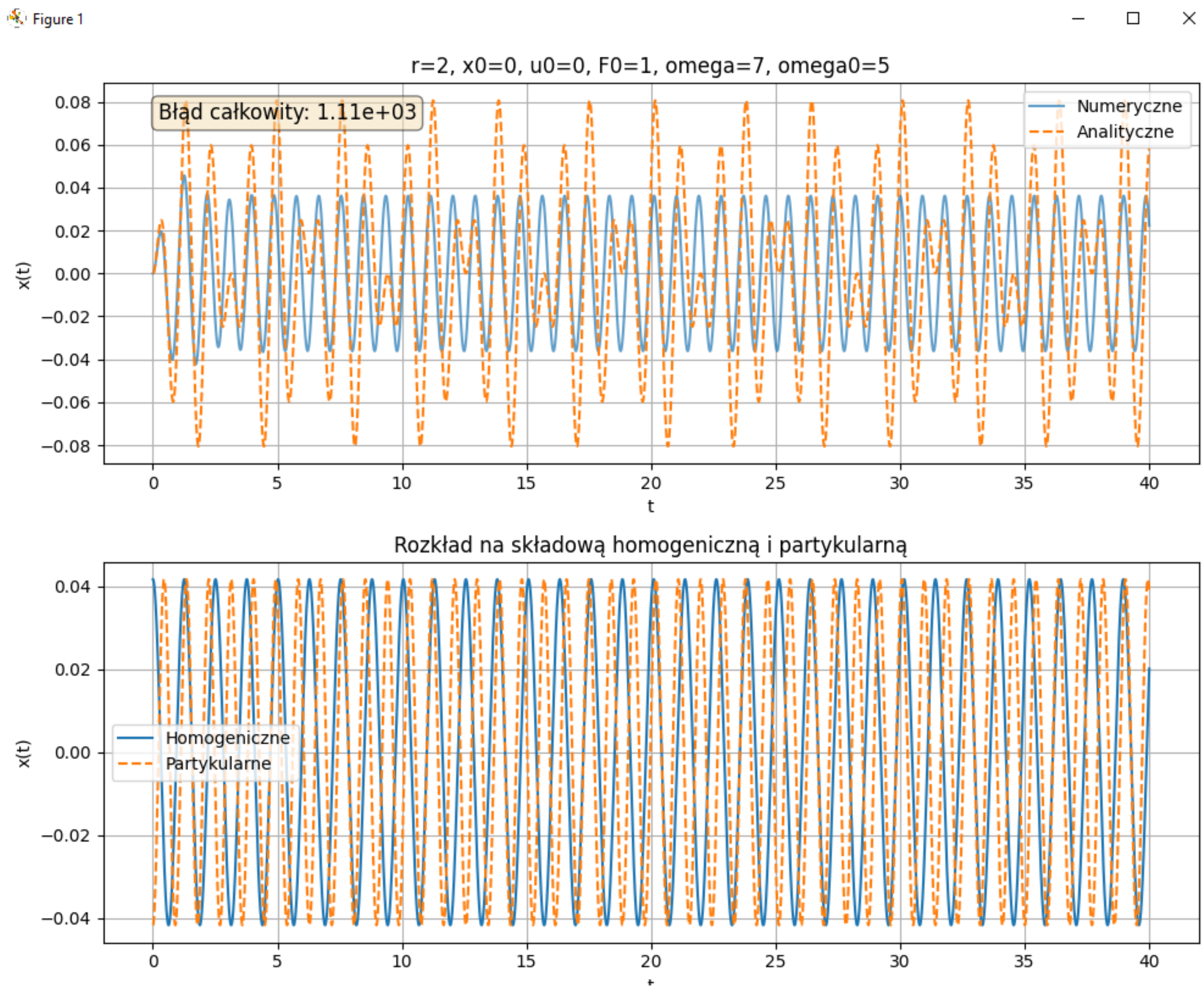


Figure 1
r=2, x0=0, u0=0, F0=1, omega=7, omega0=5
Błąd całkowity: 1.11e+03
Numeryczne
Analityczne
x(t)
t
Rozkład na składową homogeniczną i partykularną
Homogeniczne
Partykularne
x(t)
t

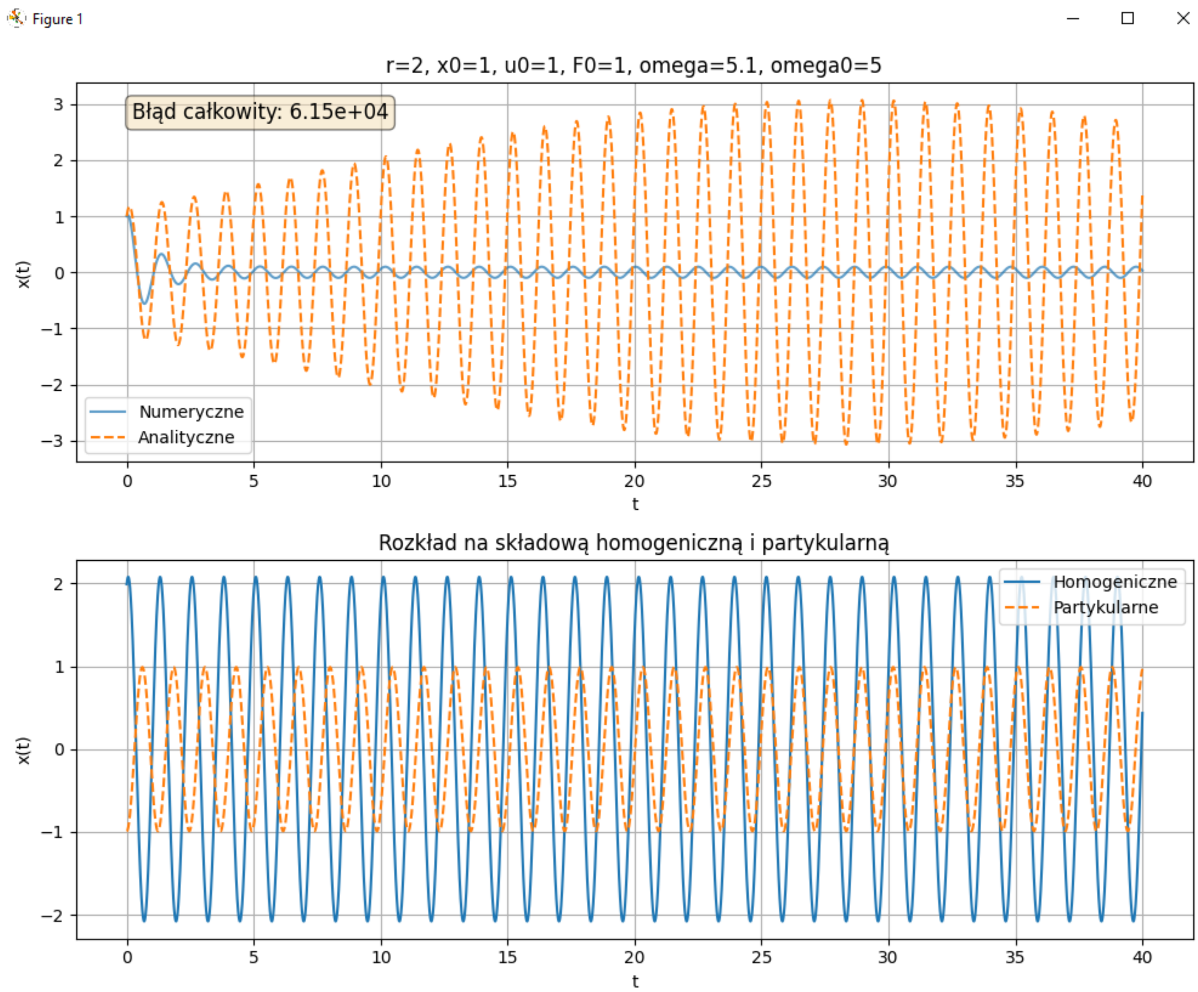


Figure 1
r=2, x0=1, u0=1, F0=1, omega=5.1, omega0=5
Błąd całkowity: 6.15e+04
x(t)
t
Numeryczne
Analityczne
Rozkład na składową homogeniczną i partykularną
Homogeniczne
Partykularne

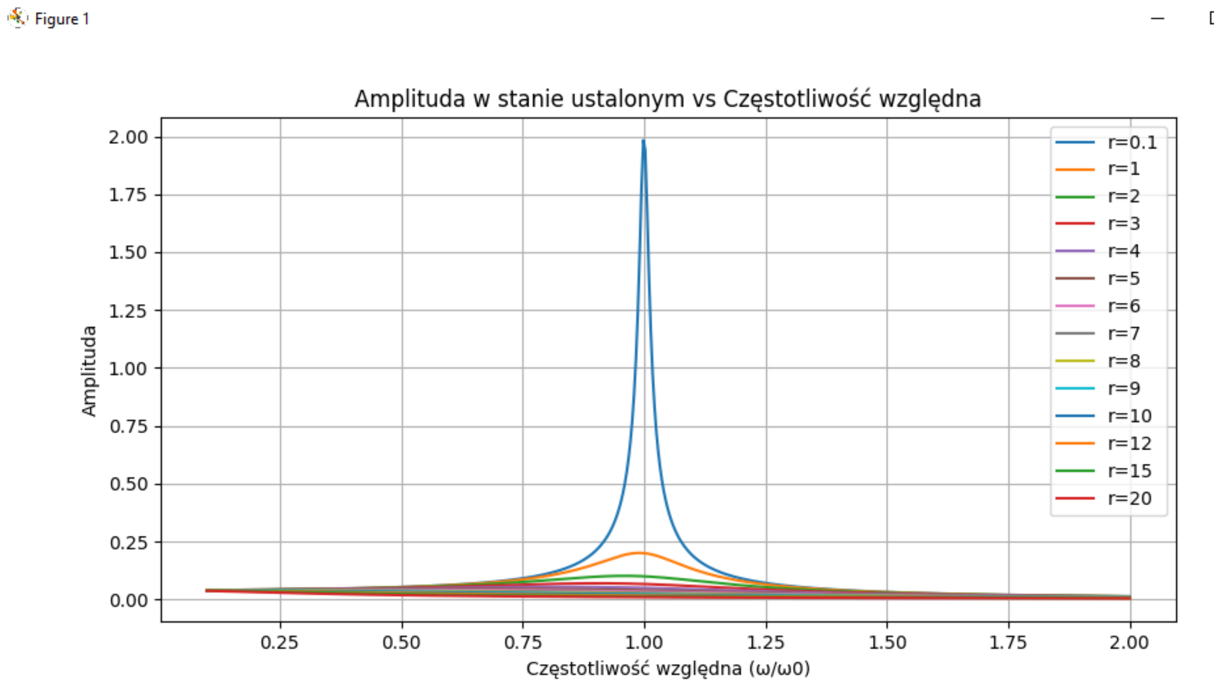
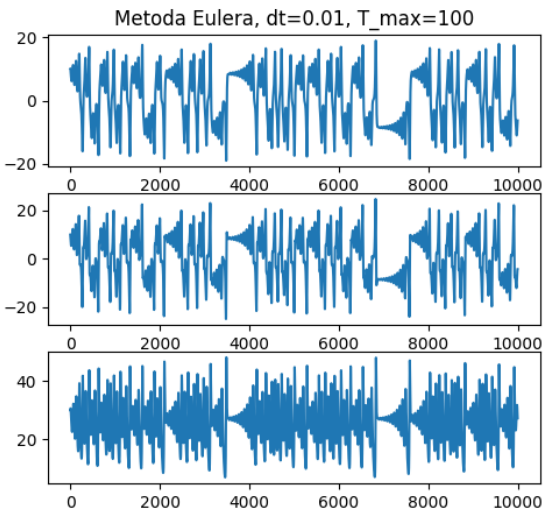


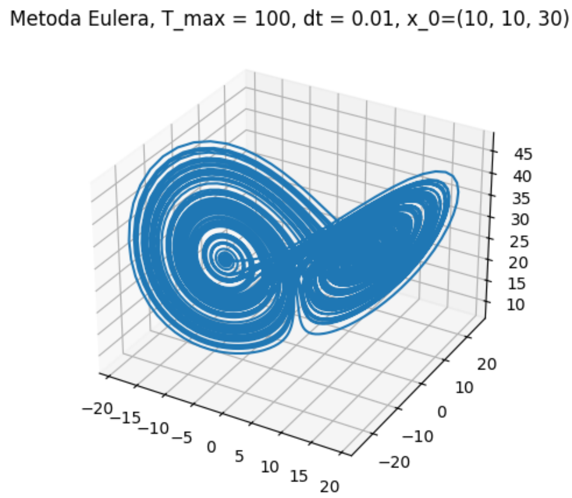
Figure 1
Amplituda w stanie ustalonym vs Częstotliwość względna
Amplituda
Częstotliwość względna (ω/ω0)
r=0.1
r=1
r=2
r=3
r=4
r=5
r=6
r=7
r=8
r=9
r=10
r=12
r=15
r=20

Zadanie 2:

```python
#ZADANIE_2#

print("ZADANIE 2")

T_max = 100
dt = 0.01

ro = 10
r = 28
b = 8/3

x = np.zeros(int(T_max/dt))
y = np.zeros(int(T_max/dt))
z = np.zeros(int(T_max/dt))

x[0] = 10
y[0] = 10
z[0] = 30

def x_n(x, y, n):
    return x[n-1] + dt * ro * (y[n-1] - x[n-1])

def y_n(x, y, z, n):
    return y[n-1] + dt * (((r - z[n-1]) * x[n-1]) - y[n-1])

def z_n(x, y, z, n):
    return z[n-1] + dt * (x[n-1] * y[n-1] - b * z[n-1])

for n in range(1, int(T_max/dt)):
    x[n] = x_n(x, y, n)
    y[n] = y_n(x, y, z, n)
    z[n] = z_n(x, y, z, n)

```

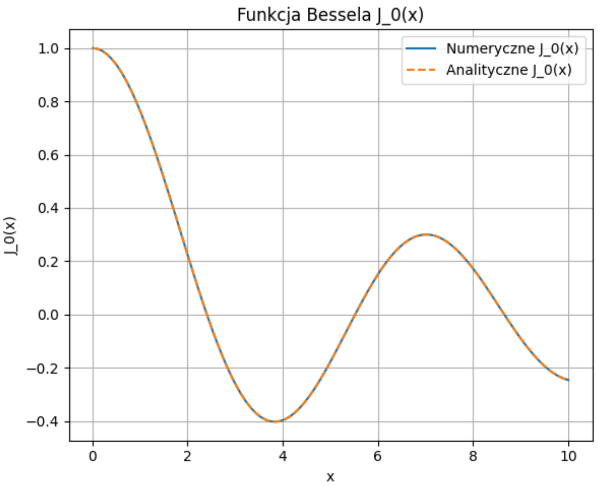
```python

fig = plt.figure(figsize=(12, 5))
gs = fig.add_gridspec(3, 2)

ax = fig.add_subplot(gs[:,1], projection='3d')
ax.set_title(f'Metoda Eulera, T_max = {T_max}, dt = {dt}, x_0={int(x[0]), int(y[0]), int(z[0])}')
ax.plot(x, y, z)

bx = fig.add_subplot(gs[0,0])
bx.set_title(f'Metoda Eulera, dt={dt}, T_max={T_max}')
bx.plot(range(0, int(T_max/dt)), x)

by = fig.add_subplot(gs[1,0])
by.plot(range(0, int(T_max/dt)), y)

bz = fig.add_subplot(gs[2,0])
bz.plot(range(0, int(T_max/dt)), z)

plt.show()
```

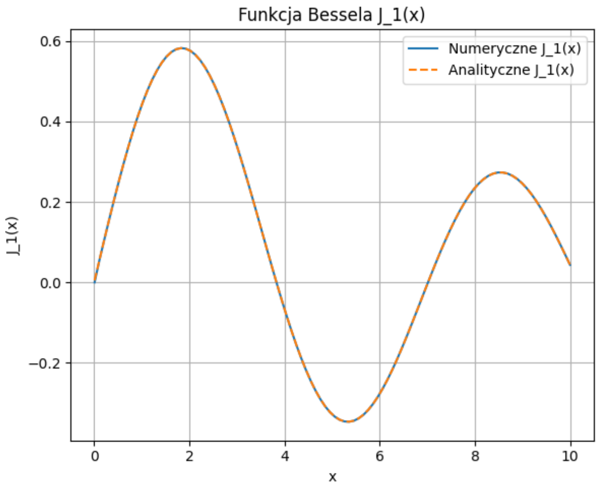




Zadanie 3:

```python
#ZADANIE_3#

print("ZADANIE 3")

# Definicja równania Bessela
def bessel_ode(x, Y, n):
    y, dy = Y
    d2y = -(x * dy + (x**2 - n**2) * y) / x**2
    return [dy, d2y]

# Rozwiązanie równania dla n=0 i n=1
x_span = (0.01, 10)
x_eval = np.linspace(*x_span, 100)

# Warunki początkowe dla funkcji Bessela
init_conditions = {
    0: [1, 0],  # J_0(0) = 1, J_0'(0) = 0
    1: [0, 1]  # J_1(0) = 0, J_1'(0) = 1
}

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

for i, n in enumerate([0, 1]):
    # Użycie metody RK45 z odpowiednią tolerancją
    sol = solve_ivp(bessel_ode, x_span, init_conditions[n], t_eval=x_eval, args=(n,), method='RK45', rtol=1e-10, atol=1e-12)

    # Wykresy
    axes[i].plot(x_eval, sol.y[0], label=f'Numeryczne J_{n}(x)')
    axes[i].plot(x_eval, jn(n, x_eval), '--', label=f'Analityczne J_{n}(x)')
    axes[i].set_xlabel("x")
    axes[i].set_ylabel(f"J_{n}(x)")
    axes[i].set_title(f"Funkcja Bessela J_{n}(x)")
    axes[i].legend()
    axes[i].grid()

plt.tight_layout()
plt.show()
```


Funkcja Bessela J_0(x)
Numeryczne J_0(x)
Analityczne J_0(x)
J_0(x)
x


Funkcja Bessela J_1(x)
Numeryczne J_1(x)
Analityczne J_1(x)
J_1(x)
x