Teoria Regulacji, Wtorek 17:05-18:45

Jan Bronicki 249011

Zadanie 2 z Listy 2 ("mini-projekt")

Dla systemu o następującej transmitancji:

$$K(s) = \frac{1}{(s+1)(s+2)}$$

Należy wyznaczyć pobudzienie na wejście:

$u(t) = 1(t)$

Z następującymi warunkami początkowymi:

$y(0) = 1$
$y'(0) = 2$

System dodatkowo można przedstawić graficznie za pomocą schematu blokowego jako:

Na początku wiedząc, że $Y(s) = K(s) \cdot U(s)$ obliczamy $y_1(t)$:
Gdzie za $U(s)$ podstawiamy to czemu równałoby się $u(t)$ po transformacie Laplace'a:

$$Y(s) = \frac{1}{(s+1)(s+2)} \cdot \frac{1}{s}$$
$$Y(s) = \frac{A}{s} + \frac{B}{s+1} + \frac{C}{s+2}$$
$$\begin{cases} A = \frac{1}{2} \\ B = -1 \\ C = \frac{1}{2} \end{cases}$$
$$Y(s) = \frac{\frac{1}{2}}{s} + \frac{-1}{s+1} + \frac{\frac{1}{2}}{s+2}$$

Następnie stosując wzory na odwrotną transformatę Laplace'a uzyskujemy wynik w dziedzinie czasu:

$$y_1(t) = \frac{1}{2}e^{-2t} - e^{-t} + \frac{1}{2}$$

Możemy takie coś osiągnąć również stosując biblioteką taką jak SymPy pozwalającą Nam na używanie zapisów symbolicznych w Python'ie

In [1]:
```
# Biblioteka SymPY
import sympy as sp
# NumPy używana do numerycznych operacji matematycznych
import numpy as np
# Matplotlib służąca do wizualizacji
import matplotlib.pyplot as plt
```

In [2]:
```
# Definiujemy obiekty biblioteki SymPy
t, y, s = sp.symbols('t y s')
# Tworzymy rownanie
Ys = 1/(s*(s+1)*(s+2))
Ys
```

Out[2]:
$$\frac{1}{s\left(s+1\right)\left(s+2\right)}$$

Następnie dokonujemy rozbicia na ułamki

In [3]:
```
Ys.apart()
```

Out[3]:
$$\frac{1}{2\left(s+2\right)} - \frac{1}{s+1} + \frac{1}{2s}$$

In [4]:
```
y1 = sp.expand(sp.inverse_laplace_transform(Ys.apart(), s, t))
y1
# theta(t) to 1(t) w bibliotece SymPy
```

Out[4]:

$$\frac{\theta(t)}{2} - e^{-t}\theta(t) + \frac{e^{-2t}\theta(t)}{2}$$

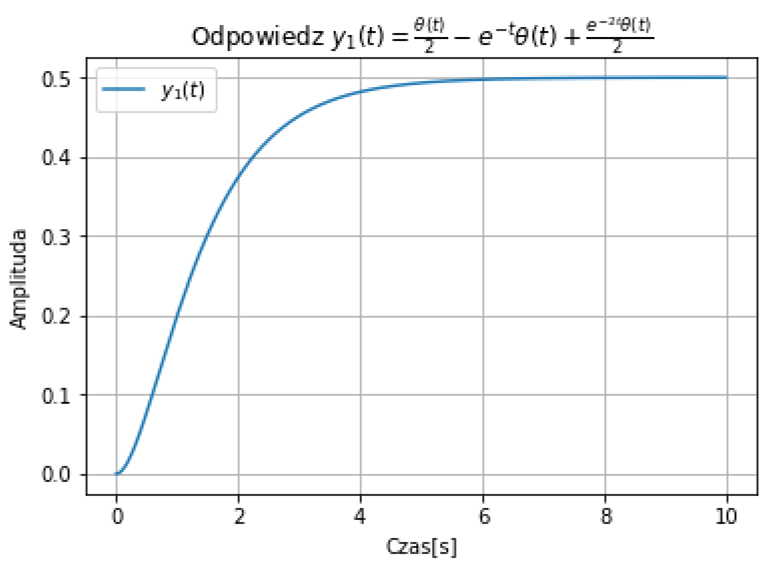
Teraz możemy narysować rozwiązanie $y_1(t)$

In [5]:
```
'''
time bedzie nasza osia czasu,
a y1_time odpowiedza jaka dostaniemy w konkretnym punkcie czasu
'''
time = np.arange(0, 10, 0.01)
y1_time = np.arange(0, 10, 0.01)

y1_lambda = sp.lambdify(t, y1, modules=['numpy', 'sympy'])

for each in range(0, len(time)):
    y1_time[each] = y1_lambda(time[each])
```

Teraz możemy narysować wynik rozwiązania numerycznego poprzez zamienienie równania symbolicznego biblioteki SymPy na funkcję (lambdę) w Pythonie z implementacją w NumPy'u dzięki funkcji:

```
sympy.utilities.lambdify(symfunc, implementation)
```

In [6]:
```
plt.plot(time, y1_time, label=("$y_{1}(t)$"))
plt.grid(True)
plt.title("Odpowiedz "+"$"+"y_{1}(t) = "+sp.latex(y1)+"$")
plt.xlabel("Czas[s]")
plt.ylabel("Amplituda")
plt.legend()
plt.show()
```



Następnie na podstawie równania charakterystycznego $s(s+1)(s+2)$ możemy dojść do oryginalnego równania różniczkowego:

$$Y(s) = \frac{1}{(s+1)(s+2)} \cdot \frac{1}{s}$$
$$Y(s)\left[s^2+3s+2\right] = \frac{1}{s}$$
$$y'' + 3y' + 2y = u(t)$$
$$s^2Y(s) - sy(0) - y' + 3sY(s) - 3y(0) + 2Y(s) = 0$$

gdzie $y(0) = 1$ oraz $y'(0) = 2$

$$Y(s) = \frac{s+5}{s^2+3s+2}$$
$$Y(s) = \frac{s+5}{(s+2)(s+1)}$$
$$Y(s) = \frac{A}{s+1} + \frac{B}{s+2}$$
$$\begin{cases} A = 4 \\ B = -3 \end{cases}$$
$$Y(s) = \frac{4}{s+1} + \frac{-3}{s+2}$$

Następnie stosujemy odwrotną transformatę Laplace'a:

$$y_2(t) = 4e^{-t} - 3e^{-2t}$$

Możemy takie coś osiągnąć również stosująć biblioteką taką jak SymPy pozwalającą Nam na używanie zapisów symbolicznych w Python'ie

In [7]:
```
Ys = (s+5)/((s**2)+3*s+2)
Ys
```

Out[7]: $$\frac{s+5}{s^2+3s+2}$$

In [8]:
```
Ys.factor()
```

Out[8]: $$\frac{s+5}{(s+1)(s+2)}$$

In [9]:
```
Ys.apart()
```

Out[9]: $$-\frac{3}{s+2} + \frac{4}{s+1}$$

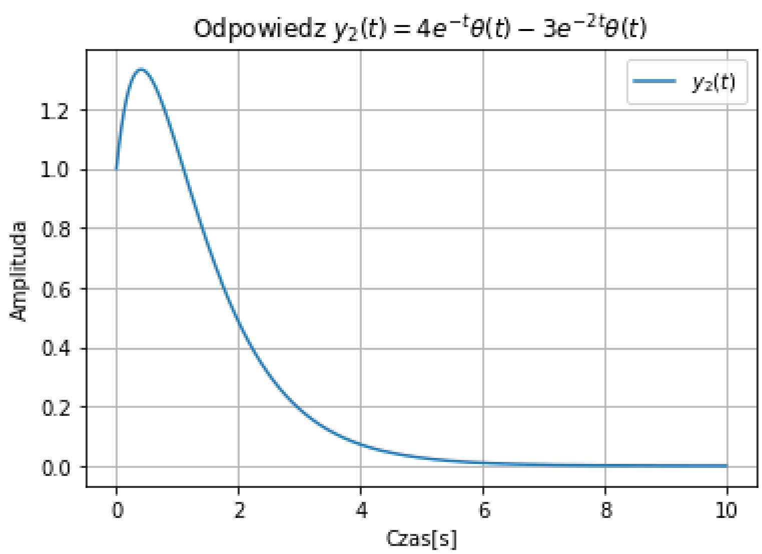
In [10]:
```
y2 = sp.expand(sp.inverse_laplace_transform(Ys.apart(), s, t))
y2
```

Out[10]: $4e^{-t}\theta(t) - 3e^{-2t}\theta(t)$

Teraz zamieniamy postać symboliczną na funkcję z, której uzyskamy wartości numeryczne:

In [11]:
```
'''
time bedzie nasza osia czasu,
a y2_time odpowiedza jaka dostaniemy w konkretnym punkcie czasu
'''
time = np.arange(0, 10, 0.01)
y2_time = np.arange(0, 10, 0.01)

y2_lambda = sp.lambdify(t, y2, modules=['numpy', 'sympy'])

'''
Z powodu implementacji Heaviside w SymPy'u obecna funkcja nie bedzie mogla zos
tac wyliczona,
dla 0, dlatego wpisujemy jej wartosc dla 0 reczne
'''
y2_time[0] = 1.0
for each in range(1, len(time)):
    y2_time[each] = y2_lambda(time[each])
```

In [12]:
```
plt.plot(time, y2_time, label=("$y_{2}(t)$"))
plt.grid(True)
plt.title("Odpowiedz "+"$"+"y_{2}(t) = "+sp.latex(y2)+"$")
plt.xlabel("Czas[s]")
plt.ylabel("Amplituda")
plt.legend()
plt.show()
```



Teraz możemy końcowo dodać nasze dwie otrzymane funkcje $y_1(t)$ oraz $y_2(t)$:

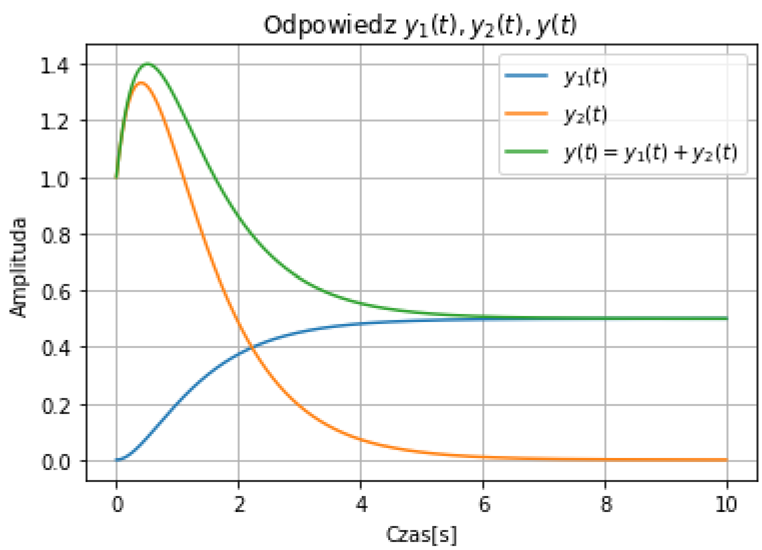
$$y(t) = y_1(t) + y_2(t) = 3e^{-t} - \frac{5}{2}e^{-2t} + \frac{1}{2}$$

Następnie rysujemy otrzymane $y(t)$ wraz z $y_1(t) + y_2(t)$, dla porównania:

In [13]:
```
y = y1+y2
y
```

Out[13]:
$$\frac{\theta\left(t\right)}{2} + 3e^{-t}\theta\left(t\right) - \frac{5e^{-2t}\theta\left(t\right)}{2}$$

In [14]:
```
# y1(t)
plt.plot(time, y1_time, label=("$y_{1}(t)$"))

# y2(t)
plt.plot(time, y2_time, label=("$y_{2}(t)$"))

plt.plot(time, y1_time+y2_time, label=("$y(t)=y_{1}(t)+y_{2}(t)$"))
plt.grid(True)
plt.title("Odpowiedz "+"$"+"y_{1}(t), y_{2}(t), y(t)"+"$")
plt.xlabel("Czas[s]")
plt.ylabel("Amplituda")
plt.legend()
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: