Wiktor Jędrzejewski
Wydział Mechaniczny Energetyki i Lotnictwa,
Politechnika Warszawska,
Warszawa, Polska
jedrzejewskiwiktor@gmail.com
kierownik pracy: dr inż. Franciszek Dul

Sprawozdanie z pracy przejściowej - model silnika turboodrzutowego jednoprzepływowego sterowanego metodą LQR

14.08.2018

Streszczenie

Utworzono model silnika w środowisku Python na podstawie pracy dyplomowej ppor. Radosława Przysowy (Model przepływowy turbinowego silnika odrzutowego D-18). Do modelu zostało dołączone sterowanie LQR, gdzie zmiennymi stanu są prędkość obrotowa wału silnika oraz wydatek paliwa a zmienną sterującą jest zmiana wydatku paliwa. Model silnika odpowiednio reaguje na zmiany wydatku paliwa poprzez zmianę obrotów silnika, ciągu oraz temperatury w odpowiednich przekrojach.

Oznaczenia wartości stałych:

Oznaczenie	Nazwa	Wartość	Jednostka
I	Moment bezwładności wirnika wysokiego ciśnienia	$1,07$	$kg \cdot m^2$
A_w	Pole przekroju dyszy silnika	$875 \cdot 10^{-4}$	m^2
W	Wartość opałowa paliwa	$41,868$	$\frac{MJ}{kg}$
η_{ks}	Sprawność komory spalania	$0,965$	
η_s	Sprawność sprężarki	$0,74$	
η_T	Sprawność turbiny	$0,9$	
R	Stała gazowa powietrza	$287,43$	$\frac{J}{kg \cdot K}$
C_p	Ciepło właściwe powietrza	$1004,83$	$\frac{J}{kg \cdot K}$
C_{pp}	Ciepło właściwe spalin	$1172,3$	$\frac{J}{kg \cdot K}$
κ	Wykładnik izentropy powietrza	$1,4$	
κ_p	Wykładnik izentropy spalin	$1,33$	
σ_{H1}	Współczynnik strat ciśnienia wlotu	$0,99$	
σ_{34}	Współczynnik strat ciśnienia w komorze spalania	$0,9578$	
σ_{6e}	Współczynnik strat ciśnienia dyszy	$0,97$	
ϵ_T	Rozpręż turbiny	$1,65$	
σ_{H1}	Współczynnik strat ciśnienia wlotu	$0,99$	
T_0	Temperatura na $H = 0\,[m]$	$288,15$	K
p_0	Ciśnienie na $H = 0\,[m]$	101325	Pa

Oznaczenia wartości zmiennych:

Oznaczenie	Nazwa	Jednostka
Ma	Liczba Macha	
H	Wysokość przelotowa	m
T_{H_0}	Temperatura na wysokości przelotowej	K
p_{H_0}	Ciśnienie na wysokości przelotowej	Pa
a_{H_0}	Prędkość dźwięku na wysokości przelotowej	$\frac{m}{s}$
v_{H_0}	Prędkość przelotowa	$\frac{m}{s}$
T_1	Temperatura na wlocie	K
T_1	Ciśnienie na wlocie	Pa
T_3	Temperatura w sprężarce	K
p_3	Ciśnienie w sprężarce	Pa
T_4	Temperatura w komorze spalania	K
p_4	Ciśnienie w komorze spalania	Pa
T_6	Temperatura w turbinie	K
p_6	Ciśnienie w turbinie	Pa
T_8	Temperatura w dyszy wylotowej	K
p_{6dw}	Ciśnienie w dyszy wylotowej	Pa
p_{krdw}	Ciśnienie krytyczne dyszy wylotowej	Pa
p_8	Ciśnienie po opuszczeniu dyszy wylotowej	Pa
$\dot{m}_s$	Wydatek masowy powietrza przepływającego przez sprężarkę	$\frac{kg}{s}$
$\dot{m}_T$	Wydatek masowy powietrza przepływającego przez turbinę	$\frac{kg}{s}$
$\dot{m}_e$	Wydatek masowy powietrza przepływającego przez dyszę	$\frac{kg}{s}$
q_{pal}	Wydatek masowy paliwa	$\frac{kg}{s}$
P_T	Moc turbiny	W
P_S	Moc sprężarki	W
w_e	Prędkość wylotowa spalin	$\frac{m}{s}$
$Thrust$	Ciąg silnika	N

Spis treści

1. Temat pracy

Tematem pracy jest utworzenie modelu turbinowego silnika odrzutowego w środowisku Python, który jest sterowany poprzez regulator liniowo - kwadratowy (LQR). Na model silnika składa się funkcja wyznaczająca jednowymiarowy rozkład temperatur oraz ciśnienia wzdłuż silnika jak i funkcja wyznaczająca wartości prawych stron układu równań różniczkowych - pochodną prędkości obrotowej silnika po czasie oraz pochodną zmiany wydatku paliwa po czasie - który reprezentuje dynamikę układu. W kolejnych krokach czasowych wartości są całkowane metodą Rungego - Kutty - Fehlberga. Regulator liniowo - kwadratowy odpowiednio zmienia sygnał sterujący - w tym przypadku zmianę wydatku paliwa - aby zmienić prędkość obrotową silnika na zadaną wartość i ją ustabilizować. Regulator liniowo - kwadratowy możemy stosować do równań liniowych, jednak dynamika układu jest nieliniowa, wobec czego niezbędna jest linearyzacja układu.

2. Model silnika

Model silnika został utworzony w oparciu o pracę dyplomową ppor. Radosława Przysowy pod tytułem "Model przepływowy turbinowego silnika odrzutowego D-18". Na model składa się dynamika układu oraz wyznaczenie parametrów termodynamicznych.

2.1. Dynamika układu

Za zmienne stanu silnika przyjęto jego prędkość obrotową (silnik jednowałowy) oraz wydatek paliwa. Dynamikę układu możemy opisać poprzez poniższy układ równań różniczkowych:

$$I \cdot \frac{dn}{dt} = \left(\frac{30}{\pi}\right)^2 \cdot \frac{P_T + P_S}{n} \left[\frac{RPM}{s}\right]$$

$$\frac{dq}{dt} = u(t) \left[\frac{kg}{s^2}\right]$$

Do wyznaczenia pochodnej prędkości obrotowej silnika po czasie potrzebne są dane o momencie bezwładności części obrotowych silnika (wału, łopatek sprężarki osiowej oraz turbiny), mocy wytwarzanej przez turbinę, mocy pochłanianej przez sprężarkę oraz aktualnej prędkości obrotowej silnika. Moment bezwładności możemy znaleźć w specyfikacji technicznej silnika a prędkość obrotową silnika mierzymy w poszczególnych krokach czasowych. Moce należy wyznaczyć poprzez analizę przepływu silnika, jako że zależą one od temperatur w poszczególnych przekrojach silnika.

Zmiana wydatku paliwa zależy tylko od sygnału sterującego $u(t)$. Wynika z tego, że jeśli nie uwzględnimy sterowania w modelu silnika, to wydatek paliwa będzie stały w czasie działania silnika. Do wyznaczenia sygnału sterującego $u(t)$ w sterowaniu LQR niezbędne jest wyznaczenie macierzy K.

Zawarta w programie funkcja `rhs` zwraca wektor $\dot{x}$, który następnie możemy całkować w celu uzyskania wyniku.

2.2. Wyznaczenie parametrów termodynamicznych oraz pochodnej prędkości obrotowej po czasie

Aby wyznaczyć parametry termodynamiczne, które są potrzebne do obliczenia mocy pobieranej przez sprężarkę oraz wytwarzanej przez turbinę, należy przeanalizować przepływ czynnika roboczego wzdłuż silnika. W modelu wyznaczono jednowymiarowy rozkład temperatury i ciśnienia. Sprawności poszczególnych modułów silnika są stałe. Funkcja realizująca wyznaczenie poniższych wartości została zawarta w programie pod nazwą `rhs_J18`.

2.2.1. Wpływ wysokości lotu silnika

Temperatura oraz ciśnienie zmienia się wraz z wysokością nad poziomem morza. Jako domyślną wartość przyjęto $T_0 = 288,15\ K$ oraz $p_0 = 101325\ Pa$ dla $H = 0\ m$. Następnie przyjęto dwa przedziały, w których zamodelowano zmianę temperatury i ciśnienia:

Dla $H < 11000\,[m]$:

$$T_{H_0} = T_0 - 0.00651 \cdot H$$

$$p_{H_0} = p_0 \cdot \left(\frac{T_{H_0}}{T_0}\right)^{5.2533}$$

Dla $H \geq 11000\,m$:

$$T_{H_0} = 216.5\ K$$

$$p_{H_0} = 23000 \cdot e^{\frac{11000-H}{6318}}$$

Możemy obliczyć lokalną prędkość dźwięku oraz prędkość przelotową:

$$a_{H_0} = \sqrt{\kappa \cdot R \cdot T_{H_0}}$$

$$v_{H_0} = Ma \cdot a_{H_0}$$

2.2.2. Wlot

Zmiana parametrów termodynamicznych na wlocie zależy od geometrii wlotu, która wpływa na współczynnik strat ciśnienia oraz od aktualnej liczby Macha.

$$T_0 = T_{H_0} \cdot \left(1 + \frac{1}{2} \cdot (\kappa - 1) \cdot Ma^2\right)$$

$$p_H = p_{H_0} \cdot \left(1 + \frac{1}{2} \cdot (\kappa - 1) \cdot Ma^2\right)^{\frac{\kappa}{\kappa-1}}$$

$$p_1 = \sigma_{H1} \cdot p_H$$

2.2.3. Sprężarka

W modelu sprężarki uwzględniono zmianę sprężu w zależności od zmiany prędkości obrotowej silnika. Wydatek masowy powietrza przelatującego przez sprężarkę również zależy od prędkości obrotowej silnika.

$$n_{nom} = 11000\ RPM$$

$$\frac{d\pi_s}{dn} = \frac{\dot{m}_s}{dn} = 1.8 \cdot 10^{-4}$$

$$\pi_s = 2.1 + \frac{d\pi_s}{dn} \cdot (n - n_{nom})$$

$$p_3 = \pi_s \cdot p_1$$

$$\dot{m}_s = 7.8 + \frac{\dot{m}_s}{dn} \cdot (n - n_{nom})$$

$$T_3 = T_1 \cdot \left(1 + \left(\pi_s^{(\frac{\kappa-1}{kappa} - 1}\right) \cdot \frac{1}{\eta_s}\right)$$

2.2.4. Komora spalania

Ciepło wytwarzane w komorze spalania zwiększa temperaturę czynnika roboczego. Wydatek masowy jest powiększony o masę spalin wytwarzanych podczas procesu spalania.

$$Q_{34} = q_{pal} \cdot \eta_{ks} \cdot W$$

$$p_4 = \sigma_{34} \cdot p_3$$

$$T_4 = T_3 + \frac{Q_{34}}{C_p \cdot \dot{m}_s}$$

$$\dot{m}_{ks} = \dot{m}_s + q_{pal}$$

2.2.5. Turbina

Zmiana parametrów termodynamicznych podczas przepływu przez turbinę zależy jedynie od jej sprawności i rozprężu. Wydatek masowy nie ulega zmianie.

$$p_6 = \frac{p_4}{\epsilon_T}$$

$$\dot{m}_T = \dot{m}_{ks}$$

$$T_6 = T_4 \cdot \left(1 - \left(1 - \epsilon_T^{\frac{1-\kappa_p}{\kappa_p}}\right) \cdot \eta_T\right)$$

2.2.6. Dysza wylotowa

Przy wyznaczaniu parametrów dyszy musimy porównać dwie wartości ciśnienia - ciśnienie krytyczne oraz ciśnienie otoczenia. Po ich obliczeniu wybieramy większą z nich.

$$p_{6dw} = \sigma_{6e} \cdot p_6$$

$$p_{krdw} = p_{6dw} \cdot \frac{2}{\kappa_p + 1}^{\frac{\kappa_p}{\kappa_p - 1}}$$

$$p_8 = max\left(p_H, p_{krdw}\right)$$

$$\epsilon_{dw} = \frac{p_8}{p_{6dw}}$$

$$T_8 = T_6$$

$$\dot{m}_e = A_w \cdot p_8 \cdot \sqrt{2 \cdot \frac{\kappa_p}{\kappa_p - 1} \cdot \frac{1}{R \cdot T_6} \cdot \left(\epsilon_{dw}^{\frac{2}{\kappa_p}} - \epsilon_{dw}^{\frac{\kappa_p+1}{\kappa_p}}\right)}$$

2.2.7. Prędkość wylotowa spalin, ciąg, moc

Znając parametry termodynamiczne wszystkich kluczowych przekrojów silnika, możemy obliczyć prędkość wylotową spalin, ciąg, moce turbiny oraz sprężarki:

$$w_e = \sqrt{2 \cdot \frac{\kappa_p}{\kappa_p - 1} \cdot R \cdot T_6 \cdot \left(1 - \epsilon_{dw}^{\frac{\kappa_p - 1}{\kappa_p}}\right)}$$

$$Thrust = \dot{m}_e \cdot w_e - \dot{m}_s \cdot v_{H_0} + A_w \cdot (p_8 - p_H)$$

$$P_T = \dot{m}_T \cdot C_{pp} \cdot (T_4 - T_6)$$

$$P_S = \dot{m}_s \cdot C_p \cdot (T_1 - T_3)$$

2.2.8. Pochodna prędkości obrotowej po czasie

Końcowym wynikiem obliczeń jest pochodna prędkości obrotowej po czasie, niezbędna do rozwiązania układu równań różniczkowych:

$$\frac{dn}{dt} = \left(\frac{30}{\pi}\right)^2 \cdot \frac{P_T + P_S}{I \cdot n} \left[\frac{RPM}{s}\right]$$

2.3. Całkowanie metodą Rungego - Kutty - Fehlberga

Metody Rungego - Kutty to zbiór algorytmów numerycznych, które pozwalają na wyznaczenie przybliżonego rozwiązania układu równań różniczkowych. Do obliczenia prędkości obrotowej silnika i wydatku paliwa zastosowano metodę Rungego - Kutty - Fehlberga. Funkcja realizująca tę metodę znajduje się w programie pod nazwą `rkf45`.

Rozwiązanie algorytmu jest w postaci:

$$y_{n+1} = y_n + h \sum_{i=0}^{s} c_i k_i$$

gdzie h jest krokiem całkowania a kolejne wartości k obliczamy następująco:

$$k_0 = f(t_n, y_n)$$

$$k_1 = f(t_n + a_1 h, y_n + h(b_{10} k_0))$$

$$k_2 = f(t_n + a_2 h, y_n + h(b_{20} k_0 + b_{21} k_1))$$

$$k_3 = f(t_n + a_3 h, y_n + h(b_{30} k_0 + b_{31} k_1 + b_{32} k_2))$$

$$k_4 = f(t_n + a_4 h, y_n + h(b_{40} k_0 + b_{41} k_1 + b_{42} k_2 + b_{43} k_3))$$

$$k_5 = f(t_n + a_5 h, y_n + h(b_{50} k_0 + b_{51} k_1 + b_{52} k_2 + b_{53} k_3 + b_{54} k_4))$$

Do wyznaczenia poszczególnych wartości współczynników k jak i rozwiązania y potrzebujemy współczynników a, b, c, które zwykle prezentowane są w formie stablicowanej (Tablica Butchera):

$$
\begin{array}{c|cccccc}
0 & 0 \\
a_1 & b_{10} \\
a_2 & b_{20} & b_{21} \\
\vdots & \vdots & & \ddots \\
a_s & b_{s0} & \cdots & b_{s,s-2} & b_{s,s-1} \\
\hline
 & c_0 & \cdots & c_{s-2} & c_{s-1} & c_s
\end{array}
$$

Oznaczenia współczynników mogą się różnić w zależności od literatury. Dla metody Rungego - Kutty - Fehlberga współczynniki przyjmują następujące wartości:

$$
\begin{array}{c|cccccc}
0 & 0 \\
\frac{1}{4} & \frac{1}{4} \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} \\
\frac{12}{13} & \frac{1932}{2197} & \frac{-7200}{2197} & \frac{7296}{2197} \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & \frac{-845}{4104} \\
\frac{1}{2} & \frac{-8}{27} & 2 & \frac{-3544}{2565} & \frac{1859}{4104} & \frac{-11}{40} \\
\hline
 & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & \frac{-9}{50} & \frac{2}{55}
\end{array}
$$

Algorytm stosujemy dla każdego kroku czasowego. Zmiana w układzie będzie wynikać z zastosowania sterowania LQR. Wpływa ono bezpośrednio na drugie równanie dynamiki układu, które po scałkowaniu daje nową wartość wydatku paliwa, wpływającą na prędkość obrotową silnika.

3. Sterowanie LQR

Dla zadanego układu w postaci:

$$\dot{x} = Ax + Bu$$

gdzie x - wektor zmiennych stanu, u - wektor zmiennych sterujących, przyjmijmy jako n, m wymiary odpowiednio x i u. Możemy zdefiniować A - macierz współczynników liniowych przy zmiennych stanu o wymiarze $n \times n$, B - macierz współczynników liniowych przy zmiennych sterujących o wymiarze $n \times m$. Funkcja kosztu jest zadana w postaci:

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt$$

Optymalizacja sterowania LQR polega na minimalizacji wartości J. Zawarte w równaniu macierze Q i R są symetrycznymi, pozytywnie zdefiniowanymi macierzami o wymiarach odpowiednio $n \times n$ i $m \times m$, przy czym $Q \geq 0, R > 0$. Wartości poszczególnych elementów macierzy są dobierane przez projektanta układu.

Do wyznaczenia funkcji kosztu potrzebujemy wartości u, którą obliczamy z równania:

$$u = -R^{-1} B^T P x = -Kx$$

Wartość P jest wyznaczana z równania:

$$0 = PA + A^T P - PBR^{-1} B^T P + Q$$

znane jako algebraiczne równanie Riccatiego.

Zdefiniowany w ten sposób wektor sterujący u będzie prowadzić układ do wyzerowania, tzn. $x = 0$. Jeśli zależy nam na osiągnięciu konkretnej wartości wektora stanu x, możemy zdefiniować u w następujący sposób:

$$u = -K\left(x - x_{des}\right)$$

gdzie x_{des} jest zadanym stanym końcowym układu.

3.1. Dobór macierzy Q i R

Pomimo tego, że regulator linowo-kwadratowy jest znany od lat 70. XX wieku, to nie jest znana uniwersalna metoda wyznaczania wartości macierzy Q i R. Najprostszą metodą jest zastosowanie macierzy jednostkowych: $Q = I, R = \rho I$ i manipulacja wartością ρ do momentu uzyskania sensownej odpowiedzi układu.

Drugą metodą jest zastosowanie reguły Brysona, gdzie poszczególne elementy macierzy wyznaczamy z zależności:

$$Q_{ii} = \frac{1}{maksymalne\ akceptowalne\ x_i^2}$$

$$R_{jj} = \frac{1}{maksymalne\ akceptowalne\ u_j^2}$$

gdzie $i = 1, 2, ..., n$; $j = 1, 2, ..., m$.
Na podstawie tej metody zdefiniowano macierze Q i R jako:

$$Q = \begin{bmatrix} 0,000001 & 0 \\ 0 & 100 \end{bmatrix}$$

$$R = \begin{bmatrix} 10000 \end{bmatrix}$$

Ostatnim sposobem wyznaczania wartości macierzy Q i R jest metoda prób i błędów. Po wykonaniu kilku obliczeń zdecydowano się obniżyć wartość w macierzy R, co ostatecznie dało następujące macierze:

$$Q = \begin{bmatrix} 0,000001 & 0 \\ 0 & 100 \end{bmatrix}$$

$$R = \begin{bmatrix} 5000 \end{bmatrix}$$

3.2. Linearyzacja układu

Jednym z mankamentów sterowania LQR jest to, że jego stosowalność jest ograniczona do układów liniowych. Ten problem można obejść poprzez linearyzację układu.

Jeśli mamy zadany układ w postaci:

$$\dot{x} = f(x, u)$$

gdzie $f(x, u)$ jest rozpatrywaną funkcją nieliniową, to odpowiednie współczynniki macierzy A i B możemy wyznaczyć poprzez obliczenie odpowiednich pochodnych cząstkowych:

$$A = \frac{\delta f\,(x, u)}{\delta x}$$

$$B = \frac{\delta f\,(x, u)}{\delta u}$$

Wartość tych pochodnych możemy wyznaczyć posługując się definicją pochodnej:

$$\frac{\delta f\,(x, u)}{\delta x} = \frac{f(x + \Delta x, u) - f(x, u)}{\Delta x}$$

$$\frac{\delta f\,(x, u)}{\delta u} = \frac{f(x, u + \Delta u) - f(x, u)}{\Delta u}$$

Procedurę tę stosujemy dla każdej zmiennej wchodzącej w skład wektorów x i u. Funkcja wyznaczająca poszczególne zmienne macierzy A i B została zawarta w programie pod nazwą `matrices_AB`.

4. Język programowania Python

Program został napisany w środowisku Python. Wybrano ten język z następujących powodów:

4.1. Przejrzystość i prostota

Python jest znany ze swojej prostej składni, przez co jest świetny do zastosowań naukowych, w których nie zależy nam na optymalizacji czasu obliczeń, a na łatwości uzyskania wyników. Język wymusza na użytkowniku przejrzystość, dzięki czemu kod jest bardziej czytelny. Ponadto ogrom dodatkowych bibliotek sprawia, że Python jest językiem bardzo wszechstronnym.

4.2. Biblioteki `control` i `numpy`

Dostępność biblioteki `control`, utworzonej przez pracowników California Institute of Technology, zdecydowanie ułatwia implementację LQR. Zawarta w niej funkcja `lqr`, która jako argumenty przyjmuje macierze A, B, Q, R, zwraca macierze K, S, E, dzięki czemu możemy uniknąć rozwiązywania skomplikowanego układu równań i wykorzystać macierz K do wyznaczenia sygnału sterującego.

Biblioteka `numpy` pozwala na przeprowadzanie operacji na macierzach, co jest niezbędne do sprawnego rozwiązywania układu równań. Pomaga również przy odpowiedniej edycji danych do utworzenia odpowiednich wykresów.

4.3. Biblioteka `matplotlib`

Biblioteka `matplotlib` umożliwia tworzenie, wyświetlanie oraz zapisywanie estetycznych wykresów z szerokiego zakresu danych. Przy zastosowaniu odpowiednio małego kroku czasowego reprezentacja tych danych w typowych arkuszach kalkulacyjnych potrafi mocno obciążać moc obliczeniową oraz zasoby pamięci komputera. Tworzenie wykresów poprzez użycie `matplotlib` jest o wiele szybsze i automatyczne.

4.4. Programowanie obiektowe

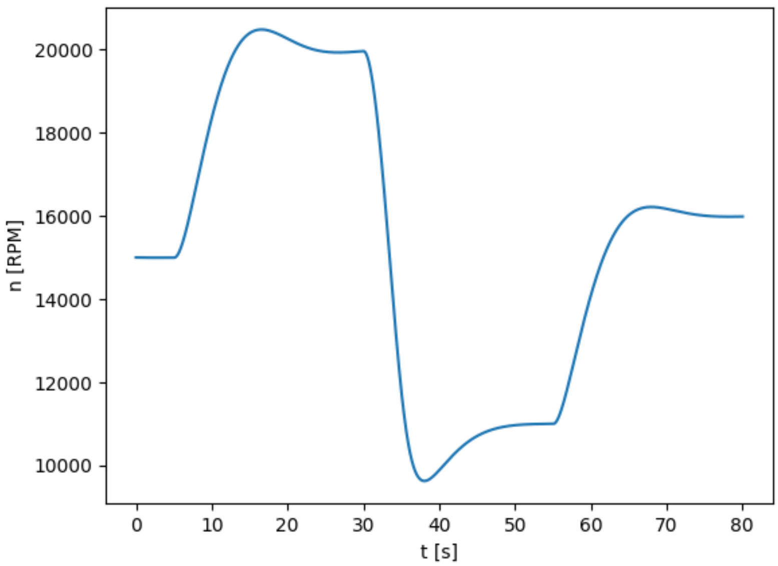
Język Python wspiera programowanie obiektowe, które możemy w pełni wykorzystać przy modelowaniu silnika. Na obiekt składa się stan oraz metody opisujące jego działanie. Wszystkie dane opisujące stan silnika, na których zależy nam najbardziej, możemy uwzględnić w stanie obiektu, przez co otrzymujemy łatwy dostęp do danych o parametrach silnika. W metodach możemy uwzględnić sposób działania silnika oraz jego dynamikę. Dzięki zastosowaniu programowania obiektowego udało się nieznacznie skrócić kod programu, poprawić jego przejrzystość oraz minimalnie przyspieszyć obliczenia względem wersji programu, w której zastosowano jedynie programowanie proceduralne.

5. Wyniki obliczeń

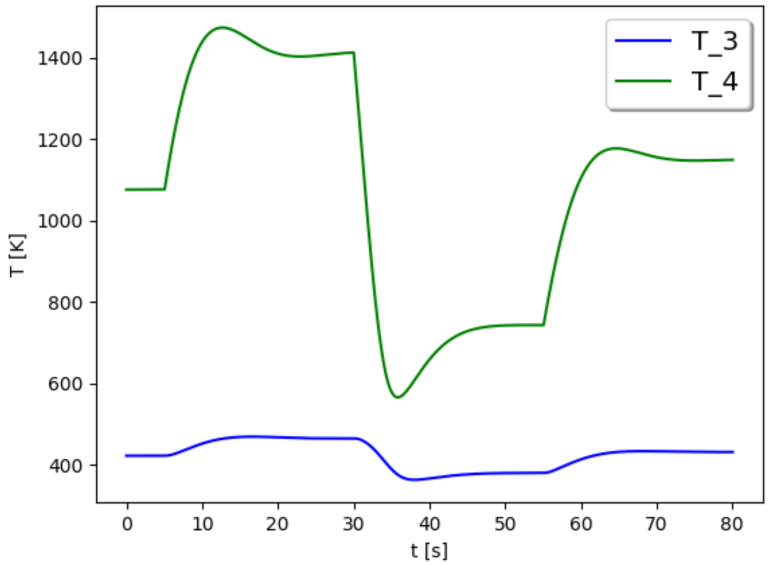
Obliczenia przeprowadzono dla wysokości przelotowej $H = 0\ m$. Dla przedziału czasowego $t \in [0, 5)$ silnik nie zmienia obrotów, utrzymuje zadaną początkową prędkość obrotową $n = 15000\ RPM$. W przedziale czasowym $t \in [5, 30)$ zadano wzrost prędkości obrotowej silnika do $n = 20000\ RPM$. Następnie w przedziale czasowym $t \in [30, 55)$ zadano obniżenie prędkości obrotowej silnika do $n = 11000\ RPM$. W ostatnim przedziale czasowym $t \in [55, 80]$ zadano wzrost prędkości obrotowej do $n = 16000\ RPM$. Wykonano wykresy prędkości obrotowej w zależności od czasu, temperatury sprężarki oraz w komorze spalania w zależności od czasu, wydatku paliwa oraz zmiany wydatku paliwa w zależności od czasu i ciągu w zależności od czasu.

Wykonano również wykresy współczynników macierzy A, B oraz K.

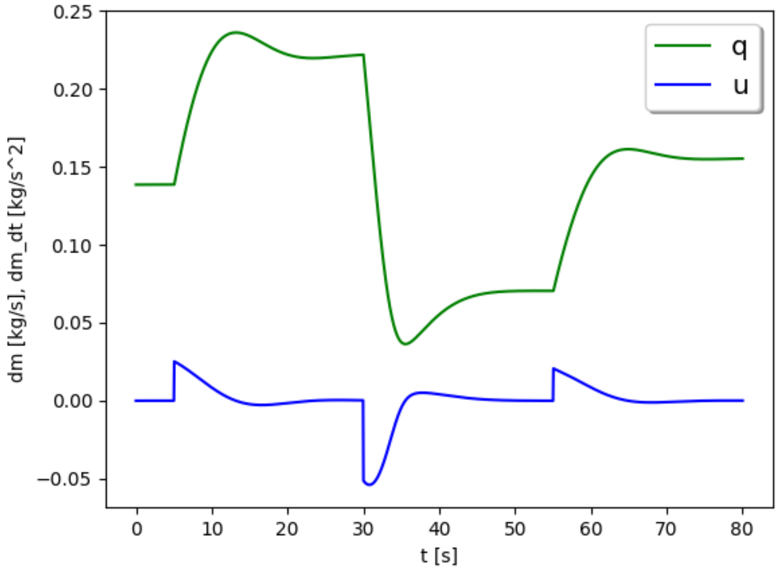
5.1. Zmiana prędkości obrotowej w czasie



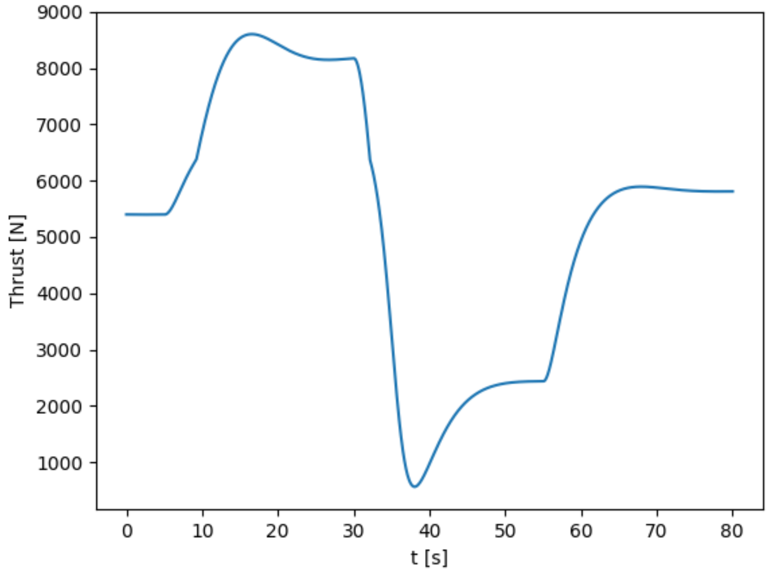
5.2. Zmiana temperatury w czasie



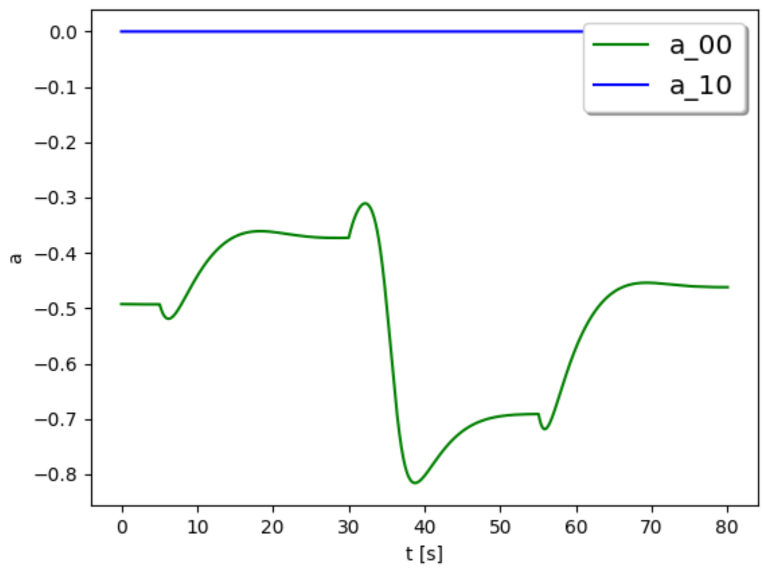
5.3. Zmiana pochodnych masy paliwa w czasie



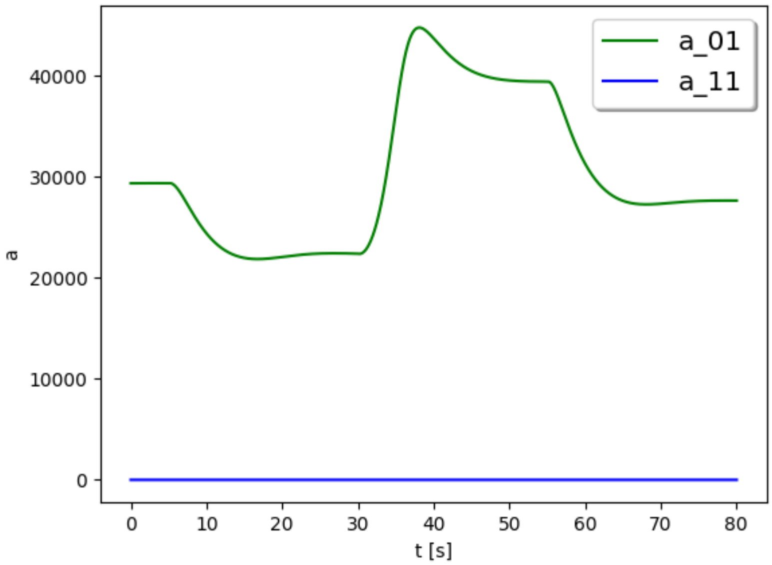
5.4. Zmiana ciągu w czasie



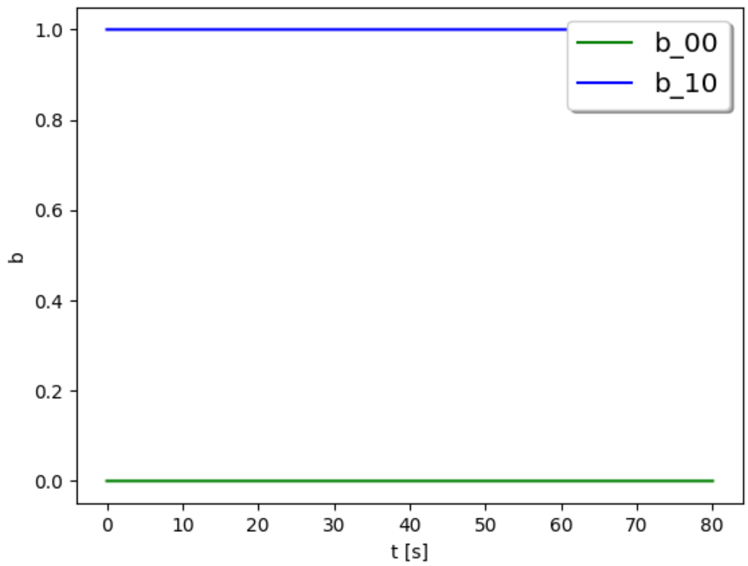
5.5. Zmiana współczynników macierzy A przy prędkości obrotowej



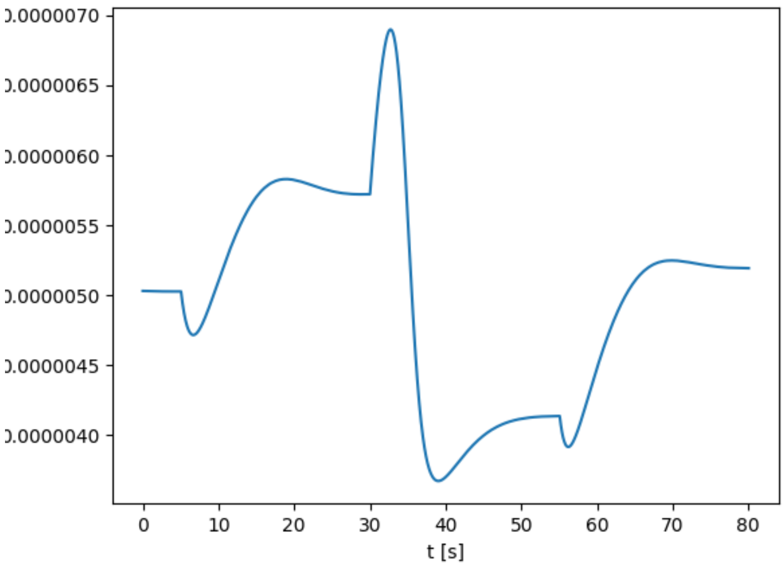
5.6. Zmiana współczynników macierzy A przy wydatku paliwa



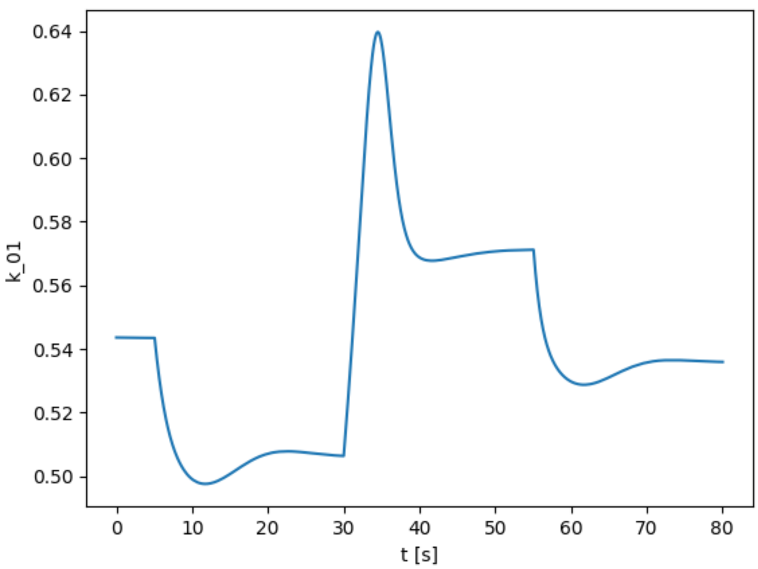
5.7. Zmiana współczynników macierzy B



5.8. Zmiana współczynnika macierzy K przy prędkości obrotowej



5.9. Zmiana współczynnika macierzy K przy wydatku paliwa



6. Kod źródłowy programu

```python
from math import exp, sqrt, pi
import numpy as np
from control import lqr
import matplotlib.pylab as plt
import time
import resource

class JetEngine:
    """Obiekt symulujacy silnik odrzutowy jednoprzeplywowy"""

    def __init__(self, x0, u0):
        """Inincjalizacja silnika - przekazanie danych o warunkach poczatkowych,
        ↪ utworzenie zmiennych przechowujacych dane o stanie silnika"""
        # warunki poczatkowe
        self.x = x0 # [RPM, kg/s]
        self.u = u0 # [kg/s^2]

        # inicjalizacja list przechowujacych temperature i cisnienie w wybranych
        ↪ przekrojach
        self.T = np.zeros(9) # [K]
        self.p = np.zeros(9) # [Pa]

        # wydatki masowe
        self.dm_s = 0 # [kg/s]
        self.dm_Twc = 0 # [kg/s]
        self.dm_e = 0 # [kg/s]

        # moce turbiny oraz sprezarki
        self.P_Twc = 0 # [W]
        self.P_s = 0 # [W]

        # ciag oraz predkosc gazow wylotowych
        self.Thrust = 0 # [N]
        self.w_e = 0 # [m/s]

    def rhs_J18(self, n_wc, q_pal):
        """Funkcja wyznaczajaca pochodna predkosci obrotowej po czasie"""

        """Stale fizyczne"""
        I_wc = 1.07 # moment bezwladnosci wirnika wysokiego cisnienia [kg*m**2]

        # A_i = 875.0 * 10**(-4.0)
        A_w = 875.0 * 10**(-4.0) # pole przekroju dyszy silnika [m**2]

        W = 41868.0 * 1000.0 # wartosc opalowa paliwa [kJ/kg]
        eta_ks = 0.965 # sprawnosc komory spalania
```

```
eta_s = 0.740 # sprawnosc sprezarki
eta_Twc = 0.9 # sprawnosc turbiny wysokiego cisnienia
R = 287.43 # stala gazowa powietrza [J/kg/K]
Cp = 1004.83 # cieplo wlasciwe powietrza [J/kg/K]
Cpp = 1172.30 # cieplo wlasciwe spalin [J/kg/K]
kappa = 1.4 # wykladnik izentropy powietrza
kappa_p = 1.33 # wykladnik izentropy mieszaniny spalin
sigma_H1 = 0.99 # wspolczynnik strat cisnienia wlotu
sigma_34 = 0.9578 # wspolczynnik strat cisnienia w komorze spalania
sigma_6e = 0.97 # wspolczynnik strat cisnienia dyszy
eps_Twc = 1.65 #+ 0.00001*(n_wc-9417.0) # rozprez turbiny wysokiego
↪ cisnienia
Ma = 0.0
H = 0.0

"""Parametry stanu ustalonego"""
# H = 0.0 m
# Ma = 0.0
# n_wc = 11000 RPM
# q_pal = 0.07 kg/s
# Pi_S = 2.1
# eps_Twc = 1.65
# T_1 = 273.15 K
# T_2 = 405 K
# T_3 = 750 K
# T_4 = 630 K
# dm = 7.8 kg/s
# Thrust = 2400 N
# tau = 2.6 s

"""Uwzglednienie wplywu wysokosci lotu na warunki atmosferyczne"""
self.T[0] = 288.15 # [K]
self.p[0] = 101325.0 # [Pa]
if H < 11000.0:
    T_H_0 = self.T[0] - 0.00651 * H # [K]
    p_H_0 = self.p[0] * (T_H_0 / self.T[0])**(5.2533) # [Pa]
else:
    T_H_0 = 216.5 # [K]
    p_H_0 = 23000.0 * exp((11000.0 - H) / 6318.0) # [Pa]
a_H_0 = sqrt(kappa * R * T_H_0)
v_H_0 = Ma * a_H_0

# rho_H_0 = p_H_0 / (R * T[1])
# dm = A_i * rho_H_0 * v_H_0

"""Wlot: i = 1"""
self.T[1] = T_H_0 * (1.0 + 0.5 * (kappa-1.0) * Ma**(2.0)) # [K]
```

```
        p_H = p_H_0 * (1.0 + 0.5 * (kappa - 1.0) * Ma**(2.0))**(kappa / (kappa -
↪       1.0)) # [Pa]
        self.p[1] = sigma_H1 * p_H # [Pa]

        # rho_1 = p[1] / (R * T[1])
        # dm = A_i * rho_H_0 * v_H_0

        """Sprezarka: i = 3"""
        n_wc_nom = 11000.0 # [RPM]
        der_dn_wc = 1.80 * 10**(-4.0)

        Pi_s = 2.1 + der_dn_wc * (n_wc - n_wc_nom) # sprez sprezarki zmienny w
↪       zaleznosci od predkosci obrotowej
        self.p[3] = Pi_s * self.p[1]
        self.dm_s = (7.8 + der_dn_wc * (n_wc - n_wc_nom)) #* p[1] / p[0] *
↪       sqrt(T[0] / T[1])
        self.T[3] = self.T[1] * (1.0 + (Pi_s**((kappa - 1.0) / kappa) - 1.0) *
↪       1.0 / eta_s)

        """Komora spalania: i = 4"""
        Q_34 = q_pal * eta_ks * W # cieplo spalania paliwa
        self.p[4] = sigma_34 * self.p[3]
        self.T[4] = self.T[3] + Q_34 / (Cp * self.dm_s)
        dm_ks = self.dm_s + q_pal

        """Turbina wysokiego cisnienia: i = 6"""
        self.p[6] = self.p[4] / eps_Twc
        self.dm_Twc = dm_ks #* p[4] / p[0] * sqrt(T[0] / T[4])
        self.T[6] = self.T[4] * (1.0 - (1.0 - eps_Twc**((1.0 - kappa_p) /
↪       kappa_p)) * eta_Twc)

        """Dysza wylotowa: i = 8"""
        p_6dw = sigma_6e * self.p[6]
        p_krdw = p_6dw * (2.0 / (kappa_p + 1.0))**(kappa_p / (kappa_p - 1.0))
        self.p[8] = max(p_H, p_krdw)
        eps_dw = self.p[8] / p_6dw
        self.T[8] = self.T[6]
        #print("T_8 = " + str(T[6]))
        #print("eps_dw = " + str(eps_dw))
        self.dm_e = A_w * self.p[8] * sqrt(2.0 * kappa_p / (kappa_p - 1.0) * 1.0
↪       / (R * self.T[6]) * (eps_dw**(2.0 / kappa_p) - eps_dw**((kappa_p + 1.0) /
↪       kappa_p)))
        #print("dm_e = " + str(dm_e))
        self.w_e = sqrt(2.0 * kappa_p / (kappa_p - 1.0) * (R * self.T[6]) * (1.0
↪       - eps_dw**((kappa_p - 1.0) / kappa_p)))

        self.Thrust = self.dm_e * self.w_e - self.dm_s * v_H_0 + A_w * (self.p[8]
↪       - p_H)
```

```
        self.P_Twc = self.dm_Twc * Cpp * (self.T[4] - self.T[6]) # moc turbiny
    ↪   wysokiego cisnienia
        self.P_s = self.dm_s * Cp * (self.T[1] - self.T[3]) # moc sprezarki

        return ((30.0 / pi)**2.0 / I_wc) * (self.P_Twc + self.P_s) / n_wc # wynik
    ↪   - dn_wc / dt

    def rhs(self, x, u):
        """Funkcja wyznaczajaca wartosci prawych stron ukladu rownan
    ↪   rozniczkowych"""

        # utworzenie listy do zwracania wynikow
        dx_dt = np.zeros(len(x))

        # obliczenie prawych stron
        dx_dt[0] = self.rhs_J18(x[0], x[1])
        dx_dt[1] = u[0]

        return dx_dt

"""Funkcje pomocnicze"""

def matrices_AB(rhs, x, u, n, m):
    """Funkcja wyznaczajaca wspolczynniki macierzy A i B sterowania LQR na
    ↪   podstawie ukladu rownan prawych stron"""

    # zdefiniowanie wartosci delty i macierzy X, U zawierajacych mozliwe
    ↪   przypadki
    d = 1.0e-6
    X = np.zeros((n, n))
    U = np.zeros((m, m))
    for i in range(n):
        X[i] = x
        X[i][i] += d
    for i in range(m):
        U[i] = u
        U[i][i] += d

    # wywolanie funkcji
    f0 = rhs(x, u)

    # wyznaczenie wartosci macierzy A
    for i in range(n):
        f1 = rhs(X[i], u)
        for j in range(n):
            A[j][i] = (f1[j] - f0[j])/d

    # wyznaczanie wartosci macierzy B
```

```
    for i in range(m):
        f1 = rhs(x, U[i])
        for j in range(n):
            B[j][i] = (f1[j] - f0[j])/d

    return A, B

def rkf45(fun, x, u, t, dt):
    """Funkcja wyznaczajaca rozwiazania rownania rozniczkowego metoda Rungego -
  ↪  Kutty - Fehlberga"""

    # wspolczynniki do obliczania kolejnych t
    a = [0,
         1./4.,
         3./8.,
         12./13.,
         1.,
         1./2.]

    # wspolczynniki do obliczania kolejnych x
    b = [[0,              0,             0,              0,                0],
         [1./4.,          0,             0,              0,                0],
         [3./32.,         9./32.,        0,              0,                0],
         [1932./2197.,    -7200./2197.,  7296./2197.,    0,                0],
         [439./216.,      -8.,           3680./513.,     -845./4104.,      0],
         [-8./27.,        2.,            -3544./2565.,   1859./4104.,
  ↪  -11./40.]]

    # wspolczynniki do obliczenia koncowego rozwiazania
    c = [16./135.,
         0,
         6656./12825.,
         28561./56430.,
         -9./50.,
         2./55.]

    y = np.zeros(len(x))

    k0 = dt * fun(x, u)

    t1 = t + a[1]*dt
    x1 = np.zeros(len(x))
    for i in range(len(x)):
        x1[i] = x[i] + b[1][0]*k0[i]
    k1 = dt * fun(x1, u)

    t2 = t + a[2]*dt
    x2 = np.zeros(len(x))
```

```
    for i in range(len(x)):
        x2[i] = x[i] + b[2][0]*k0[i] + b[2][1]*k1[i]
    k2 = dt * fun(x2, u)

    t3 = t + a[3]*dt
    x3 = np.zeros(len(x))
    for i in range(len(x)):
        x3[i] = x[i] + b[3][0]*k0[i] + b[3][1]*k1[i] + b[3][2]*k2[i]
    k3 = dt * fun(x3, u)

    t4 = t + a[4]*dt
    x4 = np.zeros(len(x))
    for i in range(len(x)):
        x4[i] = x[i] + b[4][0]*k0[i] + b[4][1]*k1[i] + b[4][2]*k2[i] +
  ↪  b[4][3]*k3[i]
    k4 = dt * fun(x4, u)

    t5 = t + a[5]*dt
    x5 = np.zeros(len(x))
    for i in range(len(x)):
        x5[i] = x[i] + b[5][0]*k0[i] + b[5][1]*k1[i] + b[5][2]*k2[i] +
  ↪  b[5][3]*k3[i] + b[5][4]*k4[i]
    k5 = dt * fun(x5, u)

    for i in range(len(x)):
        y[i] = x[i] + c[0]*k0[i] + c[1]*k1[i] + c[2]*k2[i] + c[3]*k3[i] +
  ↪  c[4]*k4[i] + c[5]*k5[i]

    return y

"""Symulacja silnika odrzutowego jednoprzeplywowego sterowanego poprzez LQR"""
"""Zmienne stanu - predkosc obrotowa silnika, wydatek paliwa - x = [n, q]"""
"""Zmienna sterujaca - zmiana wydatku paliwa - u = [dq/dt]"""
time_start = time.clock()

# utworzenie list do zapisywania wynikow do wykresow
xp1 = []
xp2 = []
Tp3 = []
Tp4 = []
up = []
Thp = []

# warunki poczatkowe
x0 = [15000.0, 0.1385] # [RPM], [kg/s]
u0 = [0] # [kg/s^2]

# zadany stan koncowy
```

```
n_des = 20000.0
q_des = x0[1] # zadajemy tylko obroty koncowe silnika
r = [n_des, q_des] # [RPM], [kg/s]

# zadane wymiary problemu
n = 2 # liczba zmiennych stanu
m = 1 # liczba zmiennych sterujacych

# deklaracja macierzy sterowania LQR
A = np.zeros((n, n)) # dynamika ukladu
B = np.zeros((n, m)) # macierz sterowania
Q = [[0.000001, 0], [0, 100]] # macierz wagi stanu ukladu
R = [5000] # macierz wagi sygnalu

# deklaracja czasu symulacji, t_0, kroku czasowego
t_max = 50.0
t_0 = 0.0
dt = 0.05
t = t_0

# inicjalizacja silnika
engine = JetEngine(x0, u0)

# petla glowna
while t <= t_max:

    # obliczenie stanu symulacji - blad wzgledny predkosci obrotowej silnika
    err_n_wc = (engine.x[0] - n_des) / n_des * 100
    # err_Thrust = (Thrust - Thrust_stab) / Thrust_stab * 100

    # obliczenie A, B
    A, B = matrices_AB(engine.rhs, engine.x, engine.u, n, m)
    # obliczenie K, S, E
    K, S, E = lqr(A, B, Q, R)
    # obliczenie bledu i odpowiadajacego u
    e = [engine.x[0] - r[0], 0]
    engine.u = [- K[0][0] * e[0] - K[0][1] * e[1]]

    # parametry silnika
    print('===============================================================')
    print('t = {time}, n_wc = {n}, err_n_wc = {err1}'.format(time=t,
↪    n=engine.x[0], err1=err_n_wc))
    print('dq_pal_dt = {q}, q_pal = {tau}'.format(q = engine.u[0], tau =
↪    engine.x[1]))
    print('P_Twc = {P1}, P_s = {P2}, q_pal = {q}'.format(P1=engine.P_Twc,
↪    P2=engine.P_s, q=engine.x[1]))
```

```
    print('T_0 = {0}, T_1 = {1},  T_3 = {2}, T_4 = {3}, T_6 = {4}, T_e = {5} [K]'.format(engine.T[0], engine.T[1], engine.T[3], engine.T[4], engine.T[6], engine.T[8]))
    print('p_0 = {0}, p_1 = {1},  p_3 = {2}, p_4 = {3}, p_6 = {4}, p_e = {5} [Pa]'.format(engine.p[0], engine.p[1], engine.p[3], engine.p[4], engine.p[6], engine.p[8]))
    print('dm_s = {s}, dm_Twc = {wc}, dm_e = {e}'.format(s=engine.dm_s, wc=engine.dm_Twc, e=engine.dm_e))
    print('w_e = {e}, Thrust = {T}'.format(e=engine.w_e, T=engine.Thrust))

    # dane do wykresow
    xp1.append(engine.x[0])
    xp2.append(engine.x[1])
    up.append(engine.u[0])
    Tp3.append(engine.T[3])
    Tp4.append(engine.T[4])
    Thp.append(engine.Thrust)

    # calkowanie
    engine.x = rkf45(engine.rhs, engine.x, engine.u, t, dt)
    t += dt

# wykresy
t_end = t
xp1 = np.array(xp1)
xp2 = np.array(xp2)
Tp3 = np.array(Tp3)
Tp4 = np.array(Tp4)
up = np.array(up)
Thp = np.array(Thp)
timer = np.linspace(t_0, t_end, num=len(xp1))

# wykres n(t)
plt.subplot(2, 2, 1)
plt.plot(timer, xp1)
plt.xlabel('t [s]')
plt.ylabel('n [RPM]')

# wykres T_3(t), T_4(t)
plt.subplot(2, 2, 2)
plt.plot(timer, Tp3, 'b-', label='T_3')
plt.plot(timer, Tp4, 'g-', label='T_4')
plt.xlabel('t [s]')
plt.ylabel('T [K]')
legend = plt.legend(loc='upper right', shadow=True, fontsize='x-large')

# wykres q_pal(t), dq_pal_dt(t)
plt.subplot(2, 2, 3)
```

```
plt.plot(timer, xp2, 'g-', label='q')
plt.plot(timer, up, 'b-', label='u')
plt.xlabel('t [s]')
plt.ylabel('dm [kg/s], dm_dt [kg/s^2]')
legend = plt.legend(loc='upper right', shadow=True, fontsize='x-large')

# wykres Thrust(t)
plt.subplot(2, 2, 4)
plt.plot(timer, Thp)
plt.xlabel('t [s]')
plt.ylabel('Thrust [N]')

print(resource.getrusage(resource.RUSAGE_SELF).ru_maxrss)
print((time.clock() - time_start))
plt.ioff()
plt.show()
```

7. Bibliografia

1. Przysowa R., Opara T., *Model przepływowy turbinowego silnika odrzutowego D-18*, Warszawa 2001
2. Murray R. M., *Lecture 2 - LQR Control*, California Institute of Technology 2006
3. DeWolf T., *Linear-Quadratic Regulation For Non-Linear Systems Using Finite Differences*, 2015
4. Hespanha J. P., *Undergraduate Lecture Notes on LQG/LQR controller design*, 2007
5. Fehlberg E., *Low-order classical Runge-Kutta formulas with step size control and their application to some heat transfer problems*, 1969