Spis treści

1 Treść zadania:

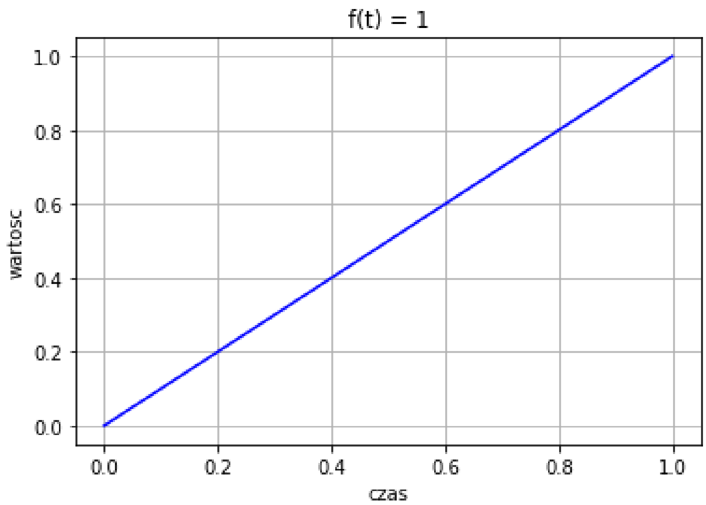
Wykorzystując pętlę for oraz metodę Eulera napisz skrypt całkujący funkcję f(t) = 1 w zakresie od 0 do 1 s. Wynik wyświetl na wykresie wraz z siatką. Osie powinny być prawidłowo podpisane.

1.1 Opis metody wykonania (kod):

```
'''
f(t_k,y_k) funckia calkowana
dt - krok calkowania
t_k obecny punkt czas symulaji
y_0 - stan poczatkwy symulacji
y_k - przyblizone rozwiazanie w punkcie t_k
y_kp1 - przyblizone rozwiazanie w unkcie t_kp1

y_kp1 = y_k + dt *f(t_k,y_k)
y(t_0) = y_0
t_kp1 = t_k +dt

'''
from numpy import zeros, linspace, pi, cos, array
import matplotlib.pyplot as plt
import math
dt = 0.01 #interwalow na okres 0.01
T = 1
N_t = int(round(T/dt)) #calkowita liczba interwalow
t = linspace(0, N_t*dt, N_t+1) #od 0 co krok dt+dt do T [0.15707963 0.31415927 0.471238
x = zeros(N_t+1) #szablon macierzy
y = zeros(N_t+1)

# Initial condition

x[0] = 0 #
```

```
y[0] = 0

# Obliczenia wedlug wzoru
for n in range(N_t):
    y[n+1] = y[n] + dt*1
    x[n+1] = x[n] + dt

fig = plt.figure()
ax1 = fig.add_subplot(111)
l1 = plt.plot(t, y, 'b-')
plt.grid()
ax1.set_title('f(t) = 1')
ax1.set_ylabel('wartosc')
ax1.set_xlabel('czas')
plt.show()
```

1.2 Wynik(wykres):



2 Treść zadania:

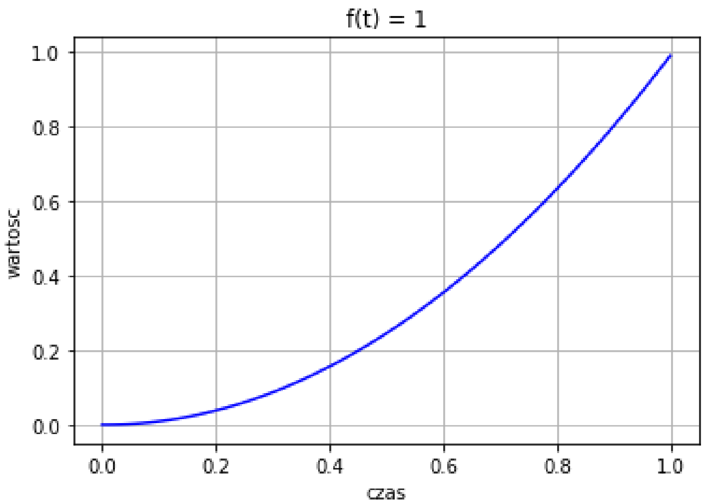
Wykorzystując pętlę for oraz metodę Eulera napisz skrypt całkujący funkcję f(t) = 2 t w zakresie od 0 do 1 s. Wynik wyświetl na wykresie wraz z siatką. Osie powinny być prawidłowo podpisane.

2.1 Opis metody wykonania (kod):

```
from numpy import zeros, linspace, pi, cos, array
import matplotlib.pyplot as plt
import math
dt = 0.01 #interwalow na okres 0.01
T = 1
N_t = int(round(T/dt)) #calkowita liczba interwalow
t = linspace(0, N_t*dt, N_t+1) #od 0 co krok dt+dt do T [0.15707963 0.31415927 0.471238
x = zeros(N_t+1) #szablon macierzy
y = zeros(N_t+1)

# Initial condition

x[0] = 0 #
y[0] = 0

# Obliczenia wedlug wzoru
for n in range(N_t):
    y[n+1] = y[n] + dt*2*x[n]
    x[n+1] = x[n] + dt

fig = plt.figure()
ax1 = fig.add_subplot(111)
l1 = plt.plot(t, y, 'b-')
plt.grid()
ax1.set_title('f(t) = 1')
ax1.set_ylabel('wartosc')
```

```
ax1.set_xlabel('czas')
plt.show()
```

2.2 Wynik(wykres):



3 Treść zadania:

Wykorzystując pętlę for oraz metodę Eulera napisz skrypt całkujący funkcję f(t) = 1 w zakresie od 0 do 1 s. Wynik wyświetl na wykresie wraz z siatką. Osie powinny być prawidłowo podpisane.

$$\dot{y}(t) = \frac{1}{0.1}u - \frac{1}{0.1}y(t) \tag{1}$$

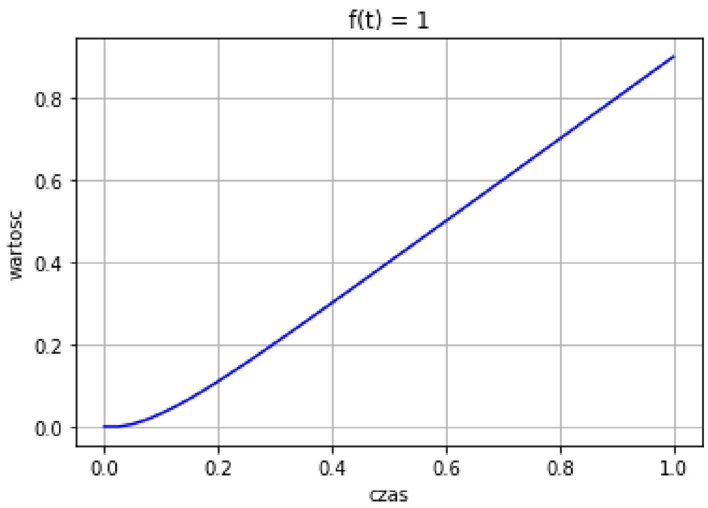
Obliczenia wykonaj w czasie od 0 do 1 s dla trzech różnych wartości kroku całkowania 0,025s, 0,01s oraz 0,000001s. Wyświetl wyniki symulacji na jednym, poprawnie sformatowanym wykresie. Jaki typ obiektu symuluje niniejsze równanie? Jak krok symulacji wpływa na wyniki?

3.1 Opis metody wykonania (kod):

```
from numpy import zeros, linspace, pi, cos, array
import matplotlib.pyplot as plt
```

```
import math
dt = 0.025 #interwalow na okres 0.01
# dt = 0.01
# dt = 0.000001
T = 1
N_t = int(round(T/dt)) #calkowita liczba interwalow
t = linspace(0, N_t*dt, N_t+1) #od 0 co krok dt+dt do T [0.15707963 0.31415927 0.471238
x = zeros(N_t+1) #szablon macierzy
y = zeros(N_t+1)
a = 1/0.1
b = 1/0.1

# Initial condition
x[0] = 0
y[0] = 0


# Obliczenia wedlug wzoru
for n in range(N_t):
    y[n+1] = y[n] + dt*((a*x[n])-(b*y[n]))
    x[n+1] = x[n] + dt

fig = plt.figure()
ax1 = fig.add_subplot(111)
l1 = plt.plot(t, y, 'b-')
plt.grid()
ax1.set_title('f(t) = 1')
ax1.set_ylabel('wartosc')
ax1.set_xlabel('czas')
plt.show()
```

3.2 Wynik(wykres):

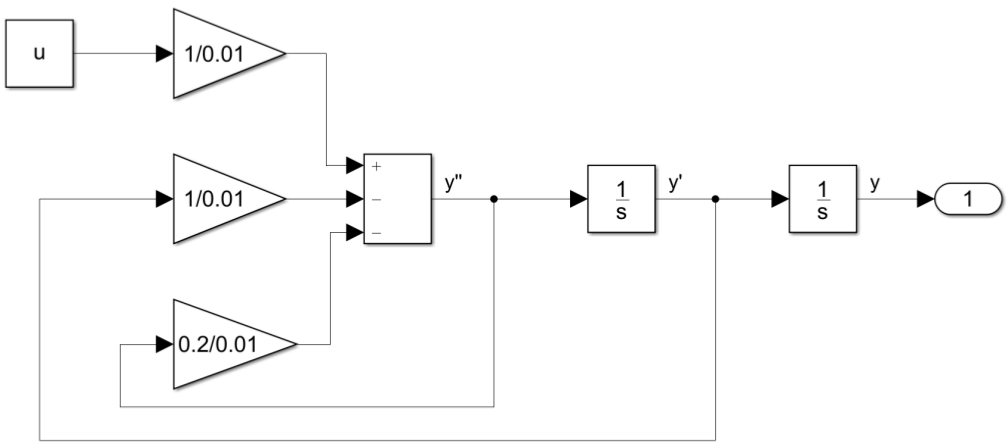


4 Treść zadania:

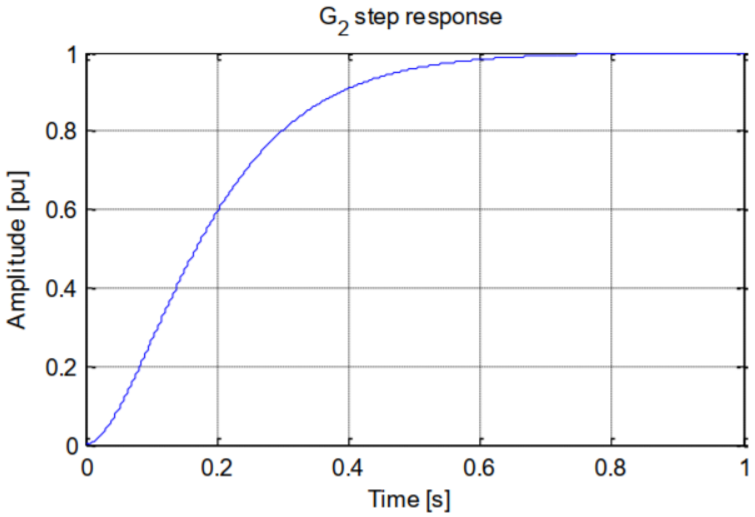
Wykorzystując pętlę for oraz metodę Eulera napisz skrypt obliczający równanie różniczkowe podane wzorem:

$$\ddot{x}(t) = \frac{0.2}{0.01}\dot{y}(t) - \frac{1}{0.01}y(t) + \frac{1}{0.01}u \tag{2}$$

Należy je podwójnie całkować. Schemat blokowy równania zamieszczono poniżej.



Obliczenia wykonaj w czasie od 0 do 1 s dla kroku całkowania równego 0,000001s. Symulacje powinny dawać następujący efekt:



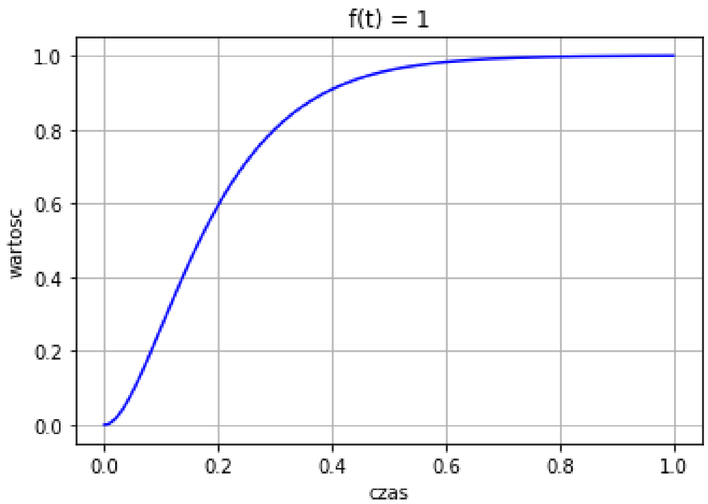
4.1 Opis metody wykonania (kod):

```
from numpy import zeros, linspace, pi, cos, array
import matplotlib.pyplot as plt
import math
dt = 0.000001 #interwalow na okres 0.01
# dt = 0.01
# dt = 0.000001
T = 1
N_t = int(round(T/dt)) #calkowita liczba interwalow
t = linspace(0, N_t*dt, N_t+1) #od 0 co krok dt+dt do T [0.15707963 0.31415927 0.471238
x = zeros(N_t+1) #szablon macierzy
y = zeros(N_t+1)
v = zeros(N_t+1)
u = zeros(N_t+1)
a = 0.2/0.01
b = 1/0.01
c = 1/0.01
```

```python
# Initial condition
x[0] = 0
y[0] = 0
v[0] = 0
# =============================================================================
# y"(x)=a*y'(x)-b*y(x)+c*u
# y'(x)=v(x)
# v'(x)=a*v(x)-b*y(x)+c*u = f(x,y,v)
# =============================================================================

# Obliczenia wedlug wzoru
for n in range(N_t):
    u[n+1] = u[n] + dt*v[n]
    v[n+1] = v[n] + dt*(-a*v[n]-b*u[n]+c*1)

# =============================================================================
# u = y
# v = y'=u'
# u' = v
# v' = a*v-b*u+c*1
#
# u[n+1] = u[n] + dt*v[n]
# v[n+1] = v[n] - dt*a*v[n]-b*u[n]+c*1
# =============================================================================

# =============================================================================
# # Obliczenia wedlug wzoru
# for n in range(N_t):
# =============================================================================
#     y[n+1] = y[n] + dt*2*x[n]
#     x[n+1] = x[n] + dt
# =============================================================================
```

```
# ==============================================================================

fig = plt.figure()
ax1 = fig.add_subplot(111)
l1 = plt.plot(t,u, 'b-')
plt.grid()
ax1.set_title('f(t) = 1')
ax1.set_ylabel('wartosc')
ax1.set_xlabel('czas')
plt.show()
```

4.2 Wynik(wykres):



5 Treść zadania:

Wykorzystując pętlę for oraz metodę Eulera napisz skrypt wykonujący symulację pracy silnika prądu stałego o równaniach stanu podanych wzorem:

$$\frac{di}{dt} = -\frac{K}{L}\omega - \frac{R}{L}i + \frac{1}{L}U \tag{3}$$

$$\frac{d\omega}{dt} = -\frac{1}{J}M + \frac{K}{J}i \tag{4}$$

gdzie: K = 1.1 – stała momentu, L = 0.00043 – indukcyjność, R = 0.36 – rezystancja, J = 0.0017 – moment bezwładności wirnika, M = 2 – moment obciążenia, U = 12 – napięcie zasilania, Obliczenia wykonaj w czasie od 0 do 0.1 s dla kroku całkowania równego 0,000001s. Wyświetl wartości prądu oraz prędkości wirnika na dwóch osobnych, poprawnie sformatowanych wykresach.

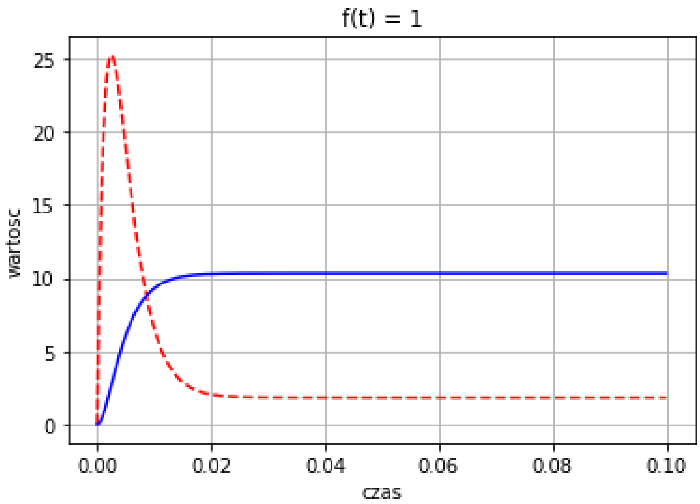
5.1 Opis metody wykonania (kod):

```
# -*- coding: utf-8 -*-
from numpy import zeros, linspace, pi, cos, array
import matplotlib.pyplot as plt
import math

T = 0.1
dt = 0.000001
N_t = int(round(T/dt)) #calkowita liczba interwalow
t = linspace(0, N_t*dt, N_t+1) #od 0 co krok dt+dt do T [0.15707963 0.31415927 0.471238
#szablon macierzy
w = zeros(N_t+1)
i = zeros(N_t+1)

K = 1.1 # stała momentu,
L = 0.00043 # indukcyjność,
R = 0.36 # rezystancja,
J = 0.017 # moment bezwładności wirnika,
M = 2 # moment obciążenia,
U = 12 # napięcie zasilania
a = K/L
b = R/L
c = 1/L
d = K/J
e = 1/J
```

```
# ============================================================================
# didt = -K/L*w-R/L*i+1/L*U
# dwdt = -1/J*M+K/J*i
# ============================================================================
print(a,'\n',b,'\n',c,'\n',d)
# Initial condition

i[0] = 0
w[0] = 0

# for n in range(N_t):
#     # i[n+1] = -a*w[n] - b*i[n] + c*U
#     # w[n+1] = -c*M + d*i[n]
#     i[n+1] = i[n] + dt*(-a*w[n] - b*i[n] + c*U)
#     w[n+1] = w[n] + dt*(-e*M + d*i[n])



for n in range(N_t):


    i[n+1] = i[n] + dt*(-a*w[n] - b*i[n] + c*U)


    w[n+1] = w[n] + dt*(-e*M + d*i[n])


fig = plt.figure()
ax1 = fig.add_subplot(111)
l1 = plt.plot(t,i, 'r--')
l1 = plt.plot(t,w, 'b-')
# l1 = plt.plot(t,U, 'g-')
plt.grid()
```

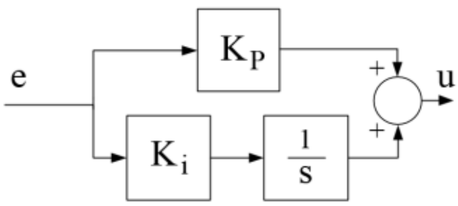
```python
ax1.set_title('f(t) = 1')
ax1.set_ylabel('wartosc')
ax1.set_xlabel('czas')
plt.show()
```

5.2 Wynik(wykres):



6 Treść zadania:

Wykorzystując skrypt z zadania 5 napisz program wykonujący symulację pracy silnika prądu stałego wraz z regulatorem prądu typu PI o strukturze podanej na rysunku:



Parametry regulatora prądu powinny wynosić:

(5) $$K_P = 0.7$$

(6) $$K_i = 1500$$

Obliczenia wykonaj w czasie od 0 do 0.01 s dla kroku całkowania równego 0,000001s i prądu zadanego równego 5 A. Wyświetl wartości prądu oraz prędkości wirnika na dwóch osobnych, poprawnie sformatowanych wykresach.

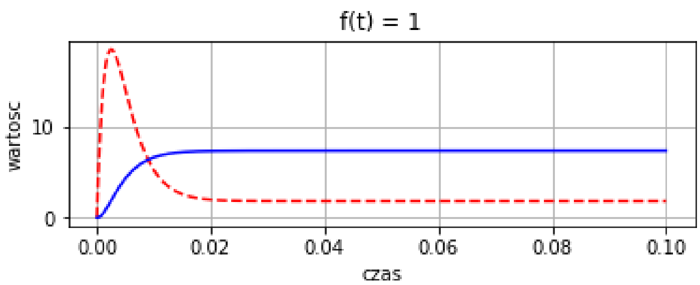
6.1 Opis metody wykonania (kod):

```
# -*- coding: utf-8 -*-
from numpy import zeros, linspace, pi, cos, array
import matplotlib.pyplot as plt
import math

T = 0.1
dt = 0.000001
N_t = int(round(T/dt)) #calkowita liczba interwalow
t = linspace(0, N_t*dt, N_t+1) #od 0 co krok dt+dt do T [0.15707963 0.31415927 0.471238
#szablon macierzy
w = zeros(N_t+1)
i = zeros(N_t+1)
q = zeros(N_t+1)
reg_prad = zeros(N_t+1)

K = 1.1 # stała momentu,
L = 0.00043 # indukcyjność,
R = 0.36 # rezystancja,
J = 0.017 # moment bezwładności wirnika,
M = 2 # moment obciążenia,
U = 12 # napięcie zasilania
a = K/L
b = R/L
c = 1/L
d = K/J
e = 1/J
```

```
#reg pradu
I = 5
Kp = 0.7
Ki = 1500


# ===============================================================================
# didt = -K/L*w-R/L*i+1/L*U
# dwdt = -1/J*M+K/J*i
# ===============================================================================
print(a,'\n',b,'\n',c,'\n',d)
# Initial condition

i[0] = 0
w[0] = 0


def Regulator_pradu(w_zad,w_wyj):

    su = 0
    uhyb_regul = w_zad - w_wyj
    p = Kp * uhyb_regul
    su = su + uhyb_regul
    i_out_dc = Ki * su
    r = p+i_out_dc

    return r




for n in range(N_t):
```

```
    reg_prad = Regulator_pradu(I,i[n+1])

    i[n+1] = i[n] + dt*w[n]
    i[n+1] = i[n] + dt*(-a*w[n] - b*i[n] + c*U-reg_prad)

    w[n+1] = w[n] + dt*i[n]
    w[n+1] = w[n] + dt*(-e*M + d*i[n])


print('reg_prad\n',reg_prad)

fig = plt.figure()
ax1 = fig.add_subplot(211)
l1 = plt.plot(t,i, 'r--')
l1 = plt.plot(t,w, 'b-')
# ax1 = fig.add_subplot(212)
# l2 = plt.plot(t,q, 'g-')
plt.grid()
ax1.set_title('f(t) = 1')
ax1.set_ylabel('wartosc')
ax1.set_xlabel('czas')
plt.show()
```

6.2 Wynik(wykres):



7 Treść zadania:

Wykorzystując skrypt z zadania 6 napisz program wykonujący symulację pracy silnika prądu stałego wraz z regulatorami prądu oraz prędkości typu PI.

Parametry regulatora prędkości powinny wynosić:

$$K_P = 19 \tag{7}$$

$$K_i = 450 \tag{8}$$

Obliczenia wykonaj w czasie od 0 do 0.05 s dla kroku całkowania równego 0,000001s i prędkości zadanej równej 100 rad/s. Wyświetl wartości prądu oraz prędkości wirnika na dwóch osobnych, poprawnie sformatowanych wykresach.

7.1 Opis metody wykonania (kod):

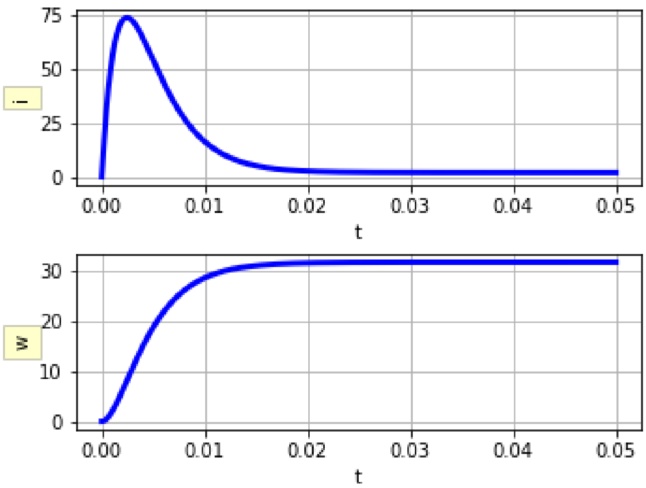
```
# -*- coding: utf-8 -*-
from numpy import zeros, linspace, pi, cos, array
import matplotlib.pyplot as plt
import math

T = 0.05
dt = 0.00001
N_t = int(round(T/dt)) #calkowita liczba interwalow
t = linspace(0, N_t*dt, N_t+1) #od 0 co krok dt+dt do T [0.15707963 0.31415927 0.471238
#szablon macierzy
w = zeros(N_t+1)
i = zeros(N_t+1)
reg_prad = zeros(N_t+1)
reg_predk = zeros(N_t+1)

K = 1.1 # stała momentu,
L = 0.00043 # indukcyjność,
```

```
R = 0.36 # rezystancja,
L = 0.017 # moment bezwładności wirnika,
M = 2 # moment obciążenia,
U = 12 # napięcie zasilania
a = K/L
b = R/L
c = 1/L
d = K/J
e = 1/J

#reg pradu
i_zad = 5
Kp = 0.7
Ki = 1500

#reg predkosci
w_zad = 100
Kpp = 19
Kip = 450

# ==============================================================================
# didt = -K/L*w-R/L*i+1/L*U
# dwdt = -1/J*M+K/J*i
# ==============================================================================
print(a,'\n',b,'\n',c,'\n',d)
# Initial condition
i[0] = 0
w[0] = 0


def Regulator_pradu(i_zad,i_silnik):

    su = 0
```

```
    uhyb_regul = i_zad - i_silnik
    p = Kp * uhyb_regul
    su = su + uhyb_regul
    w_out = Ki * su
    r = p+w_out

    return r

def Regulator_predkosci(w_zad,w_silnik):

    su = 0
    uhyb_regul =  w_zad - w_silnik
    p = Kpp * uhyb_regul
    su = su + uhyb_regul
    i_out_dc = Kip * su
    r = p+i_out_dc

    return r


for n in range(N_t):


    reg_predk = Regulator_predkosci(w_zad,w[n+1])
    reg_prad = Regulator_pradu(i_zad,i[n+1])
    in_value = U+reg_prad+reg_predk
    t[n+1] = t[n] + dt
    i[n+1] = i[n] + dt*(-a*w[n] - b*i[n] + c*U+reg_prad+reg_predk)

    w[n+1] = w[n] + dt*(-e*M + d*i[n])
```

```python
print('i\n',i)
print('\n\w\n',w)
print('\n\reg_predk\n',reg_predk)
print('\n\nreg_prad\n',reg_prad)

fig = plt.figure()
ax1 = fig.add_subplot(211)
l1 = plt.plot(t,i, 'r--')
l1 = plt.plot(t,w, 'b-')
# ax1 = fig.add_subplot(212)
# l2 = plt.plot(t,reg_prad, 'g-')
plt.grid()
ax1.set_title('f(t) = 1')
ax1.set_ylabel('wartosc')
ax1.set_xlabel('czas')
plt.show()
```

7.2 Wynik(wykres):



(9) $$u' = v, v' = -\omega^2 u$$

$$\frac{u^{n+1} - u^n}{\Delta t} = v^n \tag{10}$$

$$\frac{v^{n+1} - v^n}{\Delta t} = -\omega^2 u^n \tag{11}$$

$$x(0) = X_0, x'(0) = 0 \tag{12}$$

8 Treść zadania:

Wykorzystując skrypt z zadania 7 napisz program wykonujący symulację pracy silnika prądu stałego wraz z regulatorami prądu i prędkości typu PI oraz położenia typu P. Obliczenia wykonaj w czasie od 0 do 0.05 s dla kroku całkowania równego 0,000001s i położenia zadanego równego 1,5 rad. Wyświetl wartości prądu, prędkości wirnika oraz położenia na trzech osobnych, poprawnie sformatowanych wykresach.

8.1 Opis metody wykonania (kod):

```
# -*- coding: utf-8 -*-
from numpy import zeros, linspace, pi, cos, array
import matplotlib.pyplot as plt
import math

T = 0.05
dt = 0.00001
N_t = int(round(T/dt)) #calkowita liczba interwalow
t = linspace(0, N_t*dt, N_t+1) #od 0 co krok dt+dt do T [0.15707963 0.31415927 0.4712388
#szablon macierzy
w = zeros(N_t+1)
i = zeros(N_t+1)
fi = zeros(N_t+1)
reg_fi = zeros(N_t+1)
reg_prad = zeros(N_t+1)
```

```
reg_predk = zeros(N_t+1)

K = 1.1 # stała momentu,
L = 0.00043 # indukcyjność,
R = 0.36 # rezystancja,
J = 0.017 # moment bezwładności wirnika,
M = 2 # moment obciążenia,
U = 12 # napięcie zasilania
a = K/L
b = R/L
c = 1/L
d = K/J
e = 1/J


#reg pradu
i_zad = 5
Kp = 0.7
Ki = 1500

#reg predkosci
w_zad = 100
Kpp = 19
Kip = 450

#reg polozenia
fi_zad = 86 #1.5 rad
Kppo = 100

# ===============================================================================
# didt = -K/L*w-R/L*i+1/L*U
# dwdt = -1/J*M+K/J*i
# ===============================================================================
```

```
print(a,'\n',b,'\n',c,'\n',d)
# Initial condition
i[0] = 0
w[0] = 0
fi[0] = 0

def Regulator_pradu(i_zad,i_silnik):

    su = 0
    uhyb_regul = i_zad - i_silnik
    p = Kp * uhyb_regul
    su = su + uhyb_regul
    w_out = Ki * su
    r = p+w_out

    return r

def Regulator_predkosci(w_zad,w_silnik):

    su = 0
    uhyb_regul =  w_zad - w_silnik
    p = Kpp * uhyb_regul
    su = su + uhyb_regul
    i_out_dc = Kip * su
    r = p+i_out_dc

    return r

def Regulator_pol(fi_zad,fi_silnik):


    uhyb_regul =  fi_zad - fi_silnik
    p = Kppo * uhyb_regul
```

```python
    r = p

    return r

for n in range(N_t):

    reg_fi = Regulator_pol(fi_zad,fi[n+1])
    reg_predk = Regulator_predkosci(w_zad,w[n+1])
    reg_prad = Regulator_pradu(i_zad,i[n+1])

    U_in = c*U+(reg_prad+reg_predk+reg_fi)
    # t[n+1] = t[n] + dt
    i[n+1] = i[n] + dt*(-a*w[n] - b*i[n] + U_in)
    w[n+1] = w[n] + dt*(-e*M + d*i[n])
    #dfi/dt = w
    fi[n+1] = w[n] + dt*(w[n])


box = dict(facecolor='yellow', pad=5, alpha=0.2)

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
fig.subplots_adjust(left = 0.2, wspace = 0.4, hspace = 0.4, bottom = 0.1 )

ax1.plot(t,i[:],'b-',linewidth=3)
ax1.set_ylabel('i', bbox=box)
ax1.set_xlabel('t')
ax1.grid(True)

ax2.plot(t,w[:],'b-',linewidth=3)
ax2.set_ylabel('w', bbox=box)
ax2.set_xlabel('t')
ax2.grid(True)
```

```
ax3.set_ylabel('fi', bbox=box)
ax3.plot(t,fi[:],'k-',linewidth=3)
ax3.set_xlabel('t')
ax3.grid(True)

# ax3.yaxis.set_label_coords(xlabel, 0.5)
plt.grid(True)
plt.show()
```

8.2 Wynik(wykres):