



ZÁPADOČESKÁ
UNIVERZITA
V PLZNI



Řešení přechodných dějů na transformátoru v nástrojích jazyka python: NumPy, Matplotlib, SciPy, Spyder

Cvičení PJS

doc. Ing. Karel Noháč, Ph.D., ZČU, FEL, KEE



ZÁPADOČESKÁ
UNIVERZITA
V PLZNI



FAKULTA ELEKTROTECHNICKÁ
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
ELEKTROENERGETIKY

Výpočet přechodného děje na transformátoru

Pro výpočet přechodného děje zapnutí do stavu nakrátko využít programovací jazyk python a jeho knihovny NumPy, SciPy a Matplotlib v prostředí nástroje Spyder



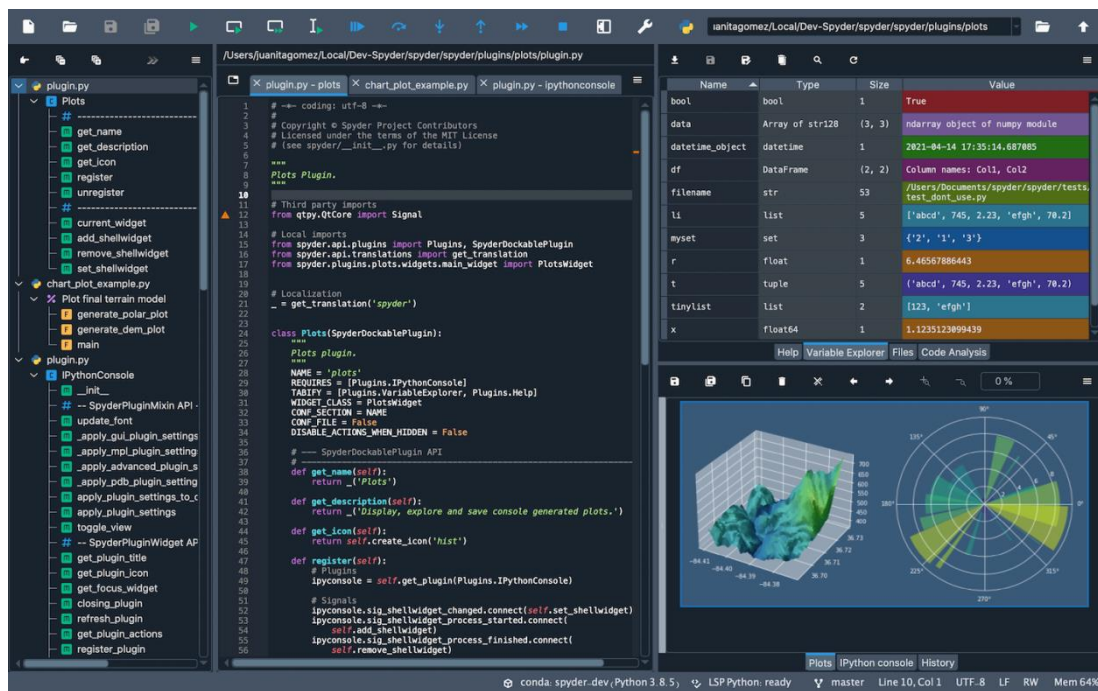
<https://www.python.org/>

<https://numpy.org/>

<https://scipy.org/>

<https://matplotlib.org/>

<https://www.spyder-ide.org/>

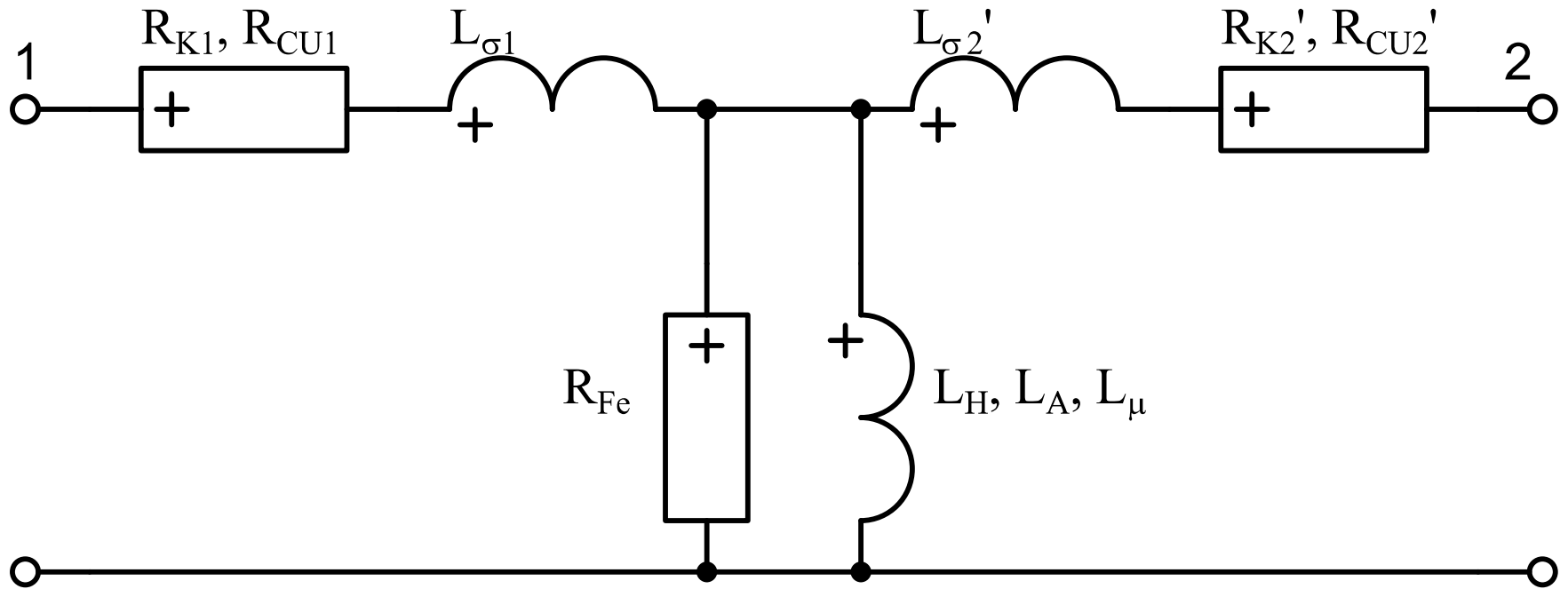


matplotlib



spyder

Náhradní schéma transformátoru



Parametry transformátoru

$$u_K = 10 \%$$

$$i_0 = 1 \%$$

$$U_{N1} = 110 \text{ kV}$$

$$U_{N2} = 22 \text{ kV}$$

$$S_{NT} = 10 \text{ MVA}$$

$$\Delta P_0 = 0.3 \%$$

$$\Delta P_K = 1.0 \%$$

$$Ukp=10.0$$

$$I0p=1.0$$

$$Un1=110.0$$

$$Un2=22.0$$

$$Snt=10.0$$

$$dP0p=0.3$$

$$dPkp=1.0$$

Parametry transformátoru

$$\omega = 2 \cdot \pi \cdot f$$

$$R_K = r_K Z_{NT} = \frac{\Delta p_{K\%}}{100} \cdot \frac{U_{N1}^2}{S_{NT}}$$

$$R_{K1} = \frac{R_K}{2}$$

$$Z_K = z_K Z_{NT} = \frac{u_{K\%}}{100} \cdot \frac{U_{N1}^2}{S_{NT}}$$

$$X_\sigma = \sqrt{Z_K^2 - R_K^2}$$

$$L_\sigma = \frac{X_\sigma}{\omega} \quad L_{\sigma 1} = \frac{L_\sigma}{2}$$

$$G_{Fe} = g_{Fe} Y_{NT} = \frac{\Delta p_{0\%}}{100} \cdot \frac{S_{NT}}{U_{N1}^2}$$

$$R_{Fe} = G_{Fe}^{-1}$$

$$Y_0 = y_0 Y_{NT} = \frac{i_{0\%}}{100} \cdot \frac{S_{NT}}{U_{N1}^2}$$

$$X_H = \left(\sqrt{Y_0^2 - G_{Fe}^2} \right)^{-1} \quad L_H = \frac{X_H}{\omega}$$

frekv=50.0

omega=2.0*np.pi*frekv

Rk=(dPkp/100.0)*(Un12/Snt)**

Rk1=Rk/2.0

Zk=(Ukp/100.0)*(Un12/Snt)**

Xs=np.sqrt(Zk2-Rk**2)**

Ls=Xs/omega

Ls1=Ls/2.0

Gfe=(dP0p/100.0)*(Snt/Un12)**

Rfe=1/Gfe

Y0=(I0p/100.0)*(Snt/Un12)**

Xh=1/np.sqrt(Y02-Gfe**2)**

Lh=Xh/omega

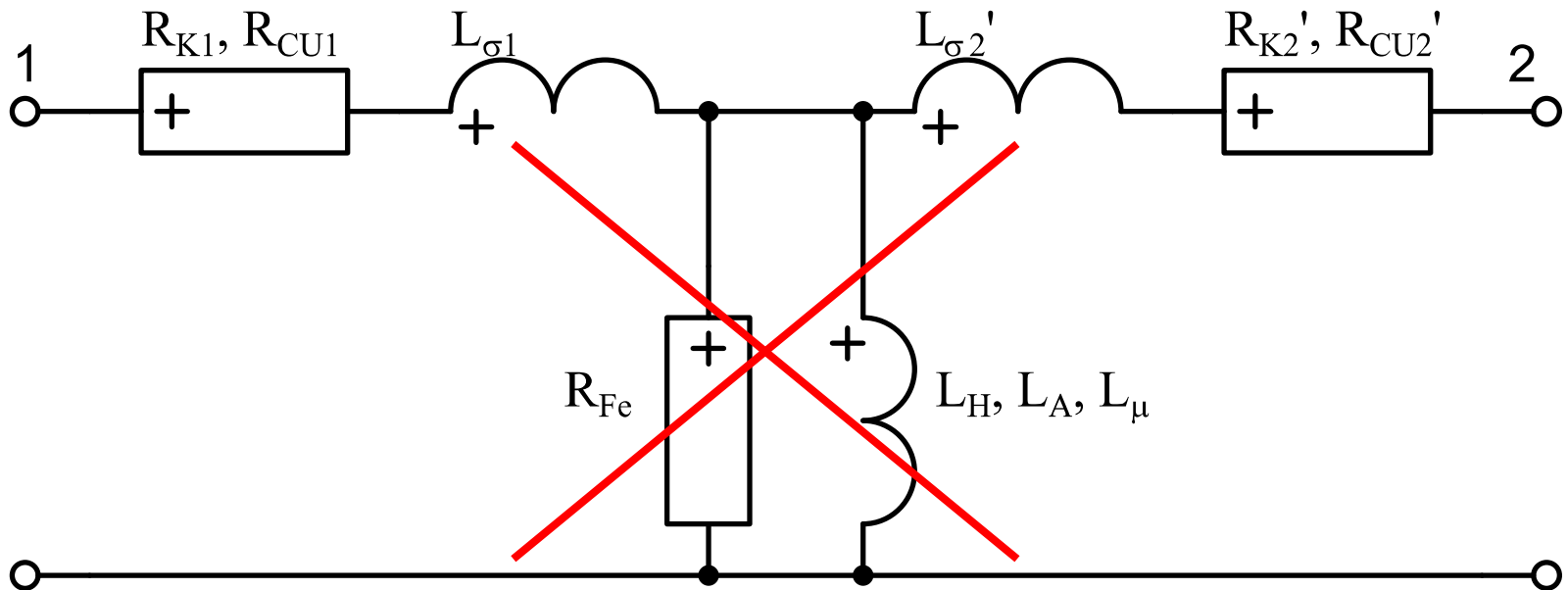
Transformátor nakrátko

Řešení numerickou metodou se zanedbáním příčné části:

$$L_{\sigma} \frac{di_K}{dt} + i_K \cdot R_K = \frac{u_K [\%]}{100} U_m \sin(\omega \cdot t) \quad U_m = \frac{U_n}{\sqrt{3}} \sqrt{2}$$



$$\frac{di_K}{dt} = \frac{\frac{u_K [\%]}{100} U_m \sin(\omega \cdot t) - i_K \cdot R_K}{L_{\sigma}}$$



Transformátor nakrátko

Řešení numerickou metodou se zanedbáním příčné části:

$$L_{\sigma} \frac{di_K}{dt} + i_K \cdot R_K = \frac{u_K [\%]}{100} U_m \sin(\omega \cdot t) \quad U_m = \frac{U_n}{\sqrt{3}} \sqrt{2}$$



$$\frac{di_K}{dt} = \frac{\frac{u_K [\%]}{100} U_m \sin(\omega \cdot t) - i_K \cdot R_K}{L_{\sigma}}$$

```
# Reseni prubehu proudu nakratko numerickou metodou
Ukm=Ukp/100.0*Un1/np.sqrt(3)*np.sqrt(2)
```

```
def ydot(y,t):
    return (Ukm*np.sin(omega*t)-y*Rk)/Ls
t = np.linspace(0., 0.1, 1000)
y0 = 0
y = odeint(ydot, y0, t)*1000.0
```

```
plt.plot(t,y)
plt.grid(True)
plt.show()
```

Zpracování pomocí integrační metody pro ODE

Transformátor nakrátko

The image shows the Spyder Python IDE interface. The left pane contains a Python script for solving an ODE using the Runge-Kutta method. The right pane shows the Variable Explorer with a table of variables and their values. The bottom pane shows the IPython console with the execution of the script.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.integrate import odeint
4
5 Ukp=10.0
6 I0p=1.0
7 Un1=110.0
8 Un2=22.0
9 Snt=10.0
10
11 dP0p=0.3;
12 dPkp=1.0;
13
14 frekv=50.0;
15 omega=2.0*np.pi*frekv;
16
17 Rk=(dPkp/100.0)*(Un1**2/Snt);
18 Rk1=Rk/2.0;
19 Zk=(Ukp/100.0)*(Un1**2/Snt);
20 Xs=np.sqrt(Zk**2-Rk**2);
21 Ls=Xs/omega;
22 Ls1=Ls/2.0;
23
24 Gfe=(dP0p/100.0)*(Snt/Un1**2);
25 Rfe=1/Gfe;
26 Y0=(I0p/100.0)*(Snt/Un1**2);
27 Xh=1/np.sqrt(Y0**2-Gfe**2);
28 Lh=Xh/omega;
29
30 # Reseni prubehu proudu nakratko numerickou metodou
31 Ukm=Ukp/100.0*Un1/np.sqrt(3)*np.sqrt(2)
32
33 def ydot(y,t):
34     return (Ukm*np.sin(omega*t)-y*Rk)/Ls
35
36 t = np.linspace(0., 0.1, 1000)
37 y0 = 0
38 y = odeint(ydot, y0, t)*1000.0
39
40 plt.plot(t,y)
41 plt.grid(True)
42 plt.show()
43
```

Nam	Type	Size	Value
dP0p	float	1	0.3
dPkp	float	1	1.0
frekv	float	1	50.0
Gfe	float	1	2.4793388429752066e-06
I0p	float	1	1.0
Lh	float64	1	403.7521067488283
Ls	float64	1	0.38322434881024886
Ls1	float64	1	0.19161217440512443
omega	float	1	314.1592653589793
Rfe	float	1	403333.3333333333
Rk	float	1	12.1
Rk1	float	1	6.05
Snt	float	1	10.0
t	Array of float64	(1000,)	[0. 0.0001001 0.0002002 ... 0.0997998 0.09989...
Ukm	float64	1	8.981462390204987
Ukp	float	1	10.0
Un1	float	1	110.0
Un2	float	1	22.0
Xh	float64	1	126842.4652433521
Xs	float64	1	120.39347988990102

```
In [34]: runfile('C:/WORK/Python/CviceniPJS2021_2.py', wdir='C:/WORK/Python')
In [35]:
```

Zpracování pomocí integrační metody pro ODE

Transformátor nakrátko

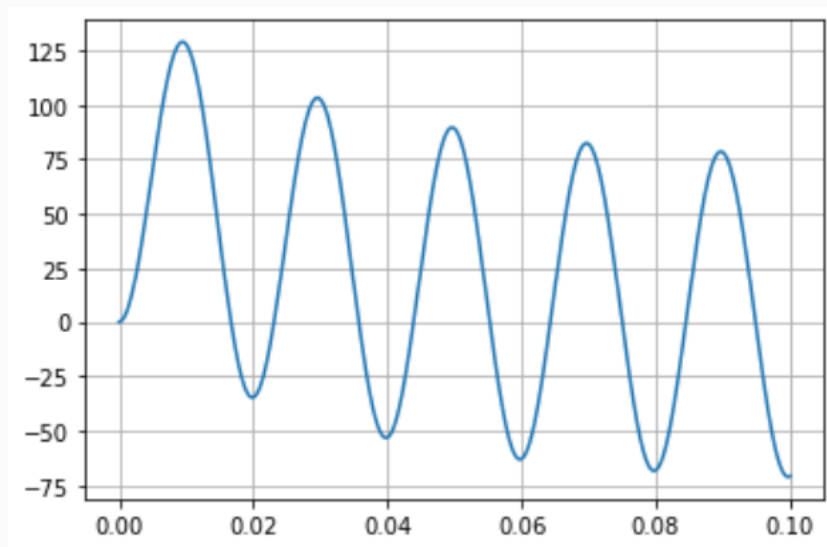
Spyder (Python 3.9)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\WORK\Python

C:\WORK\Python\CviceniPJS2021_2.py

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.integrate import odeint
4
5 Ukp=10.0
6 I0p=1.0
7 Un1=110.0
8 Un2=22.0
9 Snt=10.0
10
11 dP0p=0.3;
12 dPkp=1.0;
13
14 frekv=50.0;
15 omega=2.0*np.pi*frekv;
16
17 Rk=(dPkp/100.0)*(Un1**2/Snt);
18 Rk1=Rk/2.0;
19 Zk=(Ukp/100.0)*(Un1**2/Snt);
20 Xs=np.sqrt(Zk**2-Rk**2);
21 Ls=Xs/omega;
22 Ls1=Ls/2.0;
23
24 Gfe=(dP0p/100.0)*(Snt/Un1**2);
25 Rfe=1/Gfe;
26 Y0=(I0p/100.0)*(Snt/Un1**2);
27 Xh=1/np.sqrt(Y0**2-Gfe**2);
28 Lh=Xh/omega;
29
30 # Reseni prubehu proudu nakratko numerickou metodou
31 Ukm=Ukp/100.0*Un1/np.sqrt(3)*np.sqrt(2)
32
33 def ydot(y,t):
34     return (Ukm*np.sin(omega*t)-y*Rk)/Ls
35
36 t = np.linspace(0., 0.1, 1000)
37 y0 = 0
38 y = odeint(ydot, y0, t)*1000.0
39
40 plt.plot(t,y)
41 plt.grid(True)
42 plt.show()
43
```



165 %

Help Variable Explorer Plots Files

Console 1/A x

```
In [34]: runfile('C:/WORK/Python/CviceniPJS2021_2.py', wdir='C:/WORK/Python')
In [35]:
```

IPython console History

Kite: ready LSP Python: ready custom (Python 3.9.5) Line 32, Col 1 UTF-8-BOM CRLF RW Mem 41%

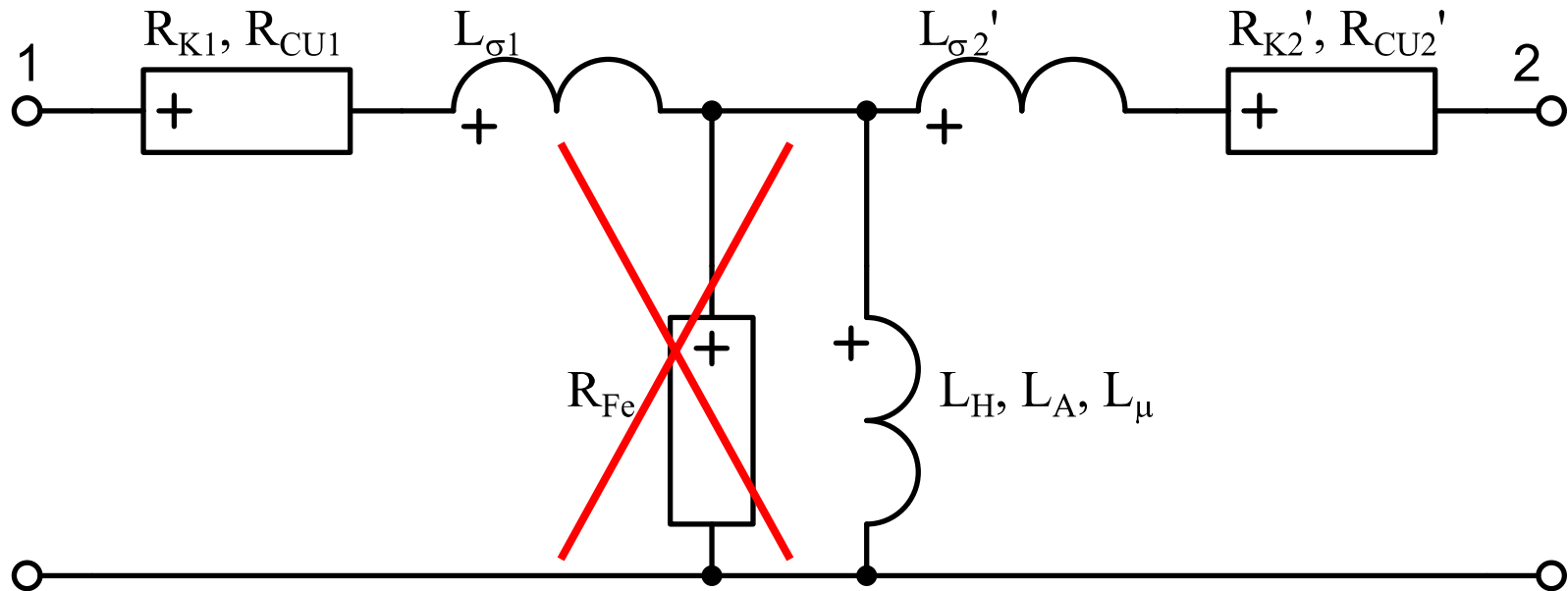
Zpracování pomocí integrační metody pro ODE

Transformátor nakrátko

Řešení numerickou metodou bez zanedbání L_H :

$$\frac{L_\sigma}{2} \frac{di_1}{dt} + i_1 \cdot \frac{R_K}{2} + L_h \frac{di_1}{dt} - L_h \frac{di_2}{dt} = \frac{u_K [\%]}{100} U_m \sin(\omega \cdot t)$$

$$-L_h \frac{di_1}{dt} + L_h \frac{di_2}{dt} + \frac{L_\sigma}{2} \frac{di_2}{dt} + i_2 \cdot \frac{R_K}{2} = 0 \quad \rightarrow$$



Transformátor nakrátko

Řešení numerickou metodou bez zanedbání L_H :

$$\frac{L_\sigma}{2} \frac{di_1}{dt} + i_1 \cdot \frac{R_K}{2} + L_h \frac{di_1}{dt} - L_h \frac{di_2}{dt} = \frac{u_K [\%]}{100} U_m \sin(\omega \cdot t)$$
$$- L_h \frac{di_1}{dt} + L_h \frac{di_2}{dt} + \frac{L_\sigma}{2} \frac{di_2}{dt} + i_2 \cdot \frac{R_K}{2} = 0 \quad \rightarrow$$

$$P_1 = \frac{L_\sigma}{2} + L_h \quad P_2 = \frac{L_\sigma}{2} + L_h - \frac{L_h^2}{P_1}$$

$$\frac{di_1}{dt} = \frac{\frac{u_K [\%]}{100} U_m \sin(\omega \cdot t) - i_1 \cdot \frac{R_K}{2} - L_h \cdot i_2 \cdot \frac{R_K}{2} \cdot \frac{1}{P_1}}{P_2}$$

$$\frac{di_2}{dt} = \frac{\frac{L_h \cdot \frac{u_K [\%]}{100} U_m \sin(\omega \cdot t)}{P_1} - \frac{L_h \cdot i_1 \cdot \frac{R_K}{2}}{P_1} - i_2 \cdot \frac{R_K}{2}}{P_2}$$

Transformátor nakrátko

$$P_1 = \frac{L_\sigma}{2} + L_h$$

$$P_2 = \frac{L_\sigma}{2} + L_h - \frac{L_h^2}{P_1}$$

$$\frac{di_1}{dt} = \frac{\frac{u_K[\%]}{100} U_m \sin(\omega \cdot t) - i_1 \cdot \frac{R_K}{2} - L_h \cdot i_2 \frac{R_K}{2} \cdot \frac{1}{P_1}}{P_2}$$

$$\frac{di_2}{dt} = \frac{\frac{L_h \cdot \frac{u_K[\%]}{100} U_m \sin(\omega \cdot t)}{P_1} - \frac{L_h \cdot i_1 \cdot \frac{R_K}{2}}{P_1} - i_2 \frac{R_K}{2}}{P_2}$$



Reseni prubehu proudu nakratko numerickou metodou

```
def ydot(y, t):
```

```
    Un1=110
```

```
    Ukp=10
```

```
    Um=Un1/np.sqrt(3)*np.sqrt(2)
```

```
    omega=314.16
```

```
    Rk = 12.100
```

```
    Ls = 0.38322
```

```
    Lh = 403.75
```

```
    pom1=Ls/2+Lh
```

```
    pom2=Ls/2+Lh-Lh**2/pom1
```

```
    return [(Ukp/100*Um*np.sin(omega*t) - y[0]*Rk/2 - Lh*y[1]*Rk/2/pom1)/pom2 ,
```

```
            (Lh*Ukp/100*Um*np.sin(omega*t)/pom1 - Lh*y[0]*Rk/2/pom1 - y[1]*Rk/2)/pom2 ]
```

```
t = np.linspace(0., 0.1, 1000)
```

```
y0 = [ 0, 0 ]
```

```
y = odeint(ydot, y0, t)*1000.0
```

```
plt.plot(t, y)
```

```
plt.grid(True)
```

```
plt.show()
```

Zpracování pomocí integrační metody pro ODE

Transformátor nakrátko

The image shows the Spyder Python IDE interface. The code editor on the left contains the following Python code:

```
1 # Reseni prubehu proudu nakratko numerickou metodou
2
3 import numpy as np
4 import matplotlib.pyplot as plt
5 from scipy.integrate import odeint
6
7 def ydot(y,t):
8     Un1=110
9     Ukp=10
10    Um=Un1/np.sqrt(3)*np.sqrt(2)
11    omega=314.16
12    Rk = 12.100
13    Ls = 0.38322
14    Lh = 403.75
15    pom1=Ls/2+Lh
16    pom2=Ls/2+Lh-Lh**2/pom1
17    return [ (Ukp/100*Um*np.sin(omega*t)-y[0]*Rk/2-Lh*y[1]*Rk/2/I
18
19 t = np.linspace(0., 0.1, 1000)
20 y0 = [ 0, 0 ]
21 y = odeint(ydot, y0, t)*1000.0
22
23 plt.plot(t,y)
24 plt.grid(True)
25 plt.show()
26
```

The plot window on the right displays a sinusoidal wave with an amplitude of approximately 125 and a period of about 0.02. The x-axis ranges from 0.00 to 0.10, and the y-axis ranges from -75 to 125.

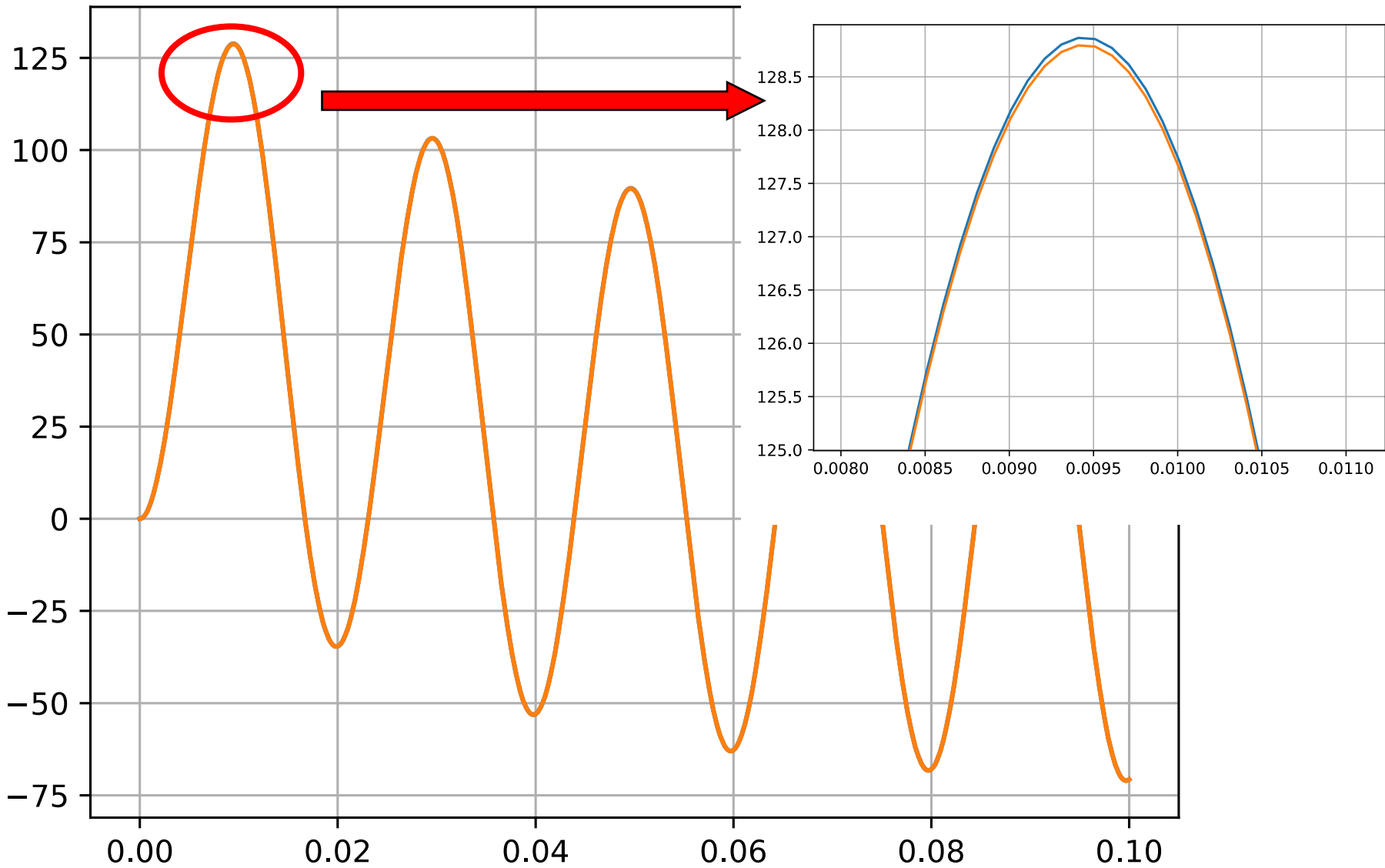
The IPython console at the bottom shows the following execution commands:

```
In [34]: runfile('C:/WORK/Python/CviceniPJS2021_2.py', wdir='C:/WORK/Python')
In [35]: runcell(0, 'C:/WORK/Python/CviceniPJS2021_3.py')
In [36]:
```

Zpracování pomocí integrační metody pro ODE

Transformátor nakrátko

Řešení numerickou metodou bez zanedbání L_H :



Zpracování pomocí integrační metody pro ODE

Transformátor nakrátko

Řešení implicitní numerickou metodou bez zanedbání L_H :

$$\frac{L_\sigma}{2} \frac{di_1}{dt} + i_1 \cdot \frac{R_K}{2} + L_h \frac{di_1}{dt} - L_h \frac{di_2}{dt} = \frac{u_K [\%]}{100} U_m \sin(\omega \cdot t)$$

$$-L_h \frac{di_1}{dt} + L_h \frac{di_2}{dt} + \frac{L_\sigma}{2} \frac{di_2}{dt} + i_2 \cdot \frac{R_K}{2} = 0 \quad \rightarrow$$

$$P_1 = \frac{L_\sigma}{2} + L_h \qquad P_2 = \frac{L_\sigma}{2} + L_h - \frac{L_h^2}{P_1}$$

$$\frac{di_1}{dt} = \frac{\frac{u_K [\%]}{100} U_m \sin(\omega \cdot t) - i_1 \cdot \frac{R_K}{2} - L_h \cdot i_2 \cdot \frac{R_K}{2} \cdot \frac{1}{P_1}}{P_2}$$

$$\frac{di_2}{dt} = \frac{\frac{L_h \cdot \frac{u_K [\%]}{100} U_m \sin(\omega \cdot t)}{P_1} - \frac{L_h \cdot i_1 \cdot \frac{R_K}{2}}{P_1} - i_2 \cdot \frac{R_K}{2}}{P_2}$$

Transformátor nakrátko

Řešení implicitní numerickou metodou bez zanedbání L_H :

$$L_{\sigma 1} \frac{di_1}{dt} + i_1 \cdot R_{K1} + L_h \frac{di_1}{dt} - L_h \frac{di_2}{dt} = U_{km} \sin(\omega \cdot t)$$

$$U_{km} = \frac{u_K [\%]}{100} U_m$$

$$-L_h \frac{di_1}{dt} + L_h \frac{di_2}{dt} + L_{\sigma 1} \frac{di_2}{dt} + i_2 \cdot R_{K1} = 0 \quad \rightarrow$$

```
# Reseni prubehu proudu nakratko numerickou metodou
Ukm=Ukp/100.0*Un1/np.sqrt(3)*np.sqrt(2)
```

```
m= GEKKO()
m.time = np.linspace(0.0, 0.1, 1000)
t = m.Param(value=m.time)
m.options.IMODE=4
i1 = m.Var(value=0.0)
i2 = m.Var(value=0.0)
```

```
# Ls1*i1' + Rk1*i1 + Lh*i1' + Lh*i2' = Ukm*sin(omega*t)
```

```
# - Lh*i1' + Lh*i2' + Ls1*i2' + Rk1*i2 = 0
```

```
m.Equation(Ls1*i1.dt() + Rk1*i1 + Lh*i1.dt() - Lh*i2.dt() == Ukm*m.sin(omega*t))
```

```
m.Equation(-Lh*i1.dt() + Lh*i2.dt() + Ls1*i2.dt() + Rk1*i2 == 0)
```

```
m.solve(dispatch=False)
plt.plot(t,i1)
plt.plot(t,i2)
plt.grid()
plt.show()
```

Zpracování pomocí soustavy implicitních diferenciálních rovnic
a nástroje GEKKO <https://github.com/BYU-PRISM/GEKKO>

Transformátor nakrátko

The image shows the Spyder Python IDE interface. The main window displays a Python script for solving a system of implicit differential equations using the GEKKO library. The script defines parameters, sets up the GEKKO model, and solves it. A variable explorer on the right shows the state of the model variables. The IPython console at the bottom shows the execution output.

```
1 # Reseni prubehu proudu nakratko numerickou metodou
2
3 from gekko import GEKKO
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 Ukp=10.0
8 I0p=1.0
9 Un1=110.0
10 Un2=22.0
11 Snt=10.0
12
13 dP0p=0.3
14 dPkp=1.0
15
16 frekv=50.0
17 omega=2.0*np.pi*frekv
18
19 Rk=(dPkp/100.0)*(Un1**2/Snt)
20 Rk1=Rk/2.0
21 Zk=(Ukp/100.0)*(Un1**2/Snt)
22 Xs=np.sqrt(Zk**2-Rk**2)
23 Ls=Xs/omega
24 Ls1=Ls/2.0
25
26 Gfe=(dP0p/100.0)*(Snt/Un1**2)
27 Rfe=1/Gfe
28 Y0=(I0p/100.0)*(Snt/Un1**2)
29 Xh=1/np.sqrt(Y0**2-Gfe**2)
30 Lh=Xh/omega
31
32 # Reseni prubehu proudu nakratko numerickou metodou
33 Ukm=Ukp/100.0*Un1/np.sqrt(3)*np.sqrt(2)
34
35 m= GEKKO()
36 m.time = np.linspace(0.0, 0.1, 1000)
37 t = m.Param(value=m.time)
38 m.options.IMODE=4
39 i1 = m.Var(value=0.0)
40 i2 = m.Var(value=0.0)
41
42 # Ls1*i1' + Rk1*i1 + Lh*i1' + Lh*i2' = Ukm*sin(omega*t)
43 # -Lh*i1' + Lh*i2' + Ls1*i2' + Rk1*i2 = 0
44 m.Equation(Ls1*i1.dt() + Rk1*i1 + Lh*i1.dt() - Lh*i2.dt() == Ukm*m.sin(omega*t))
45 m.Equation(-Lh*i1.dt() + Lh*i2.dt() + Ls1*i2.dt() + Rk1*i2 == 0)
46
47 m.solve(dis= False)
48 plt.plot(t,i1)
49 plt.plot(t,i2)
50 plt.grid()
51 plt.show()
```

Nam	Type	Size	Value
dP0p	float	1	0.3
dPkp	float	1	1.0
frekv	float	1	50.0
Gfe	float	1	2.4793388429752066e-06
I0p	float	1	1.0
i1	gk_variable.GKVariable	1000	GKVariable object of gekko.gk_variable module
i2	gk_variable.GKVariable	1000	GKVariable object of gekko.gk_variable module
Lh	float64	1	403.7521067488283
Ls	float64	1	0.38322434881024886
Ls1	float64	1	0.19161217440512443
m	gekko.GEKKO	1	GEKKO object of gekko.gekko module
omega	float	1	314.1592653589793
Rfe	float	1	403333.3333333333
Rk	float	1	12.1
Rk1	float	1	6.05
Snt	float	1	10.0
t	gk_parameter.GKParameter	1000	GKParameter object of gekko.gk_parameter module
Ukm	float64	1	8.981462390204987
Ukp	float	1	10.0
Un1	float	1	110.0
Un2	float	1	22.0
Xh	float64	1	126842.4652433521
Xs	float64	1	120.39347988990102
Y0	float	1	8.264462809917356e-06
Zk	float	1	121.0

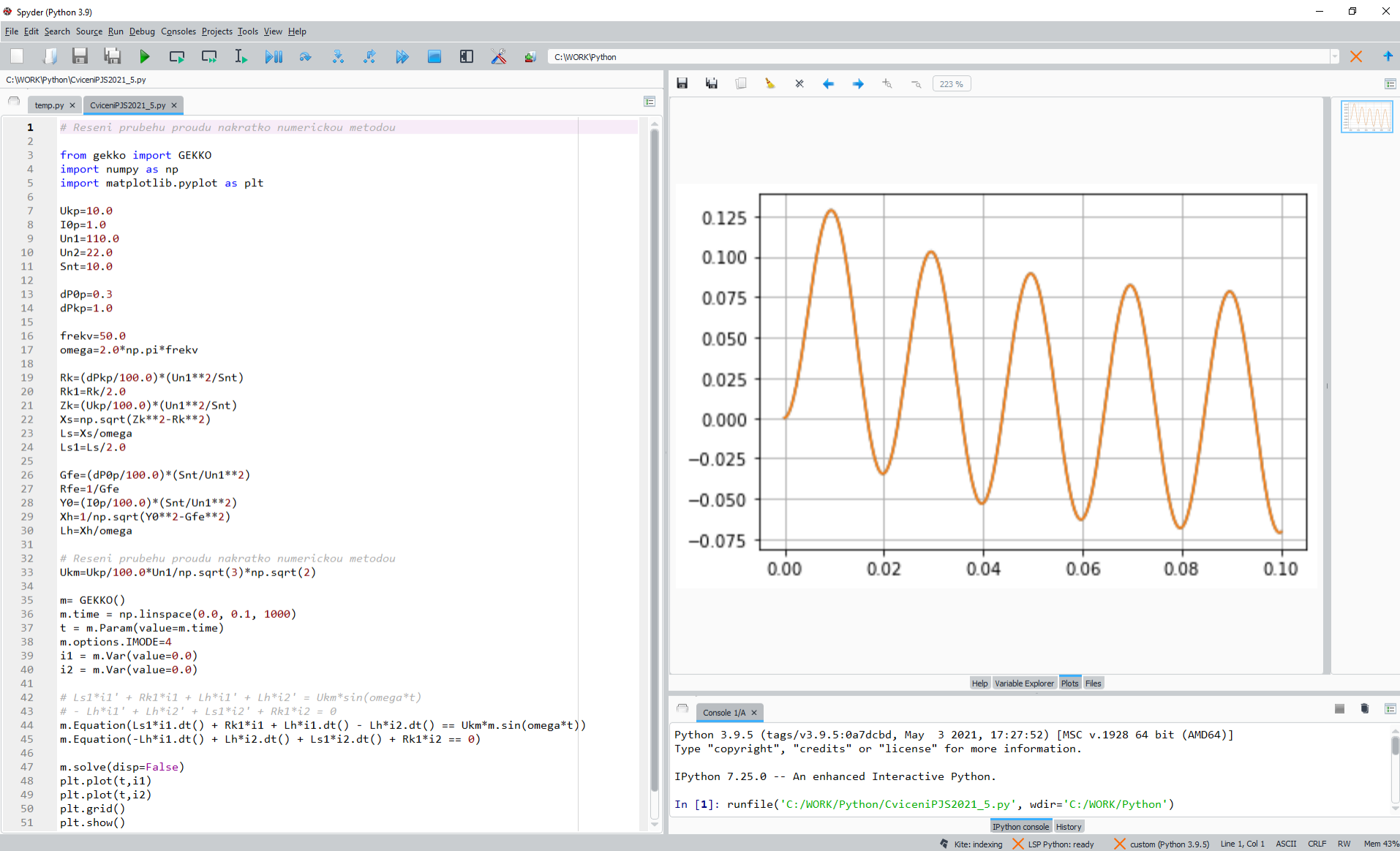
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.25.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/WORK/Python/CviceniPJS2021_5.py', wdir='C:/WORK/Python')

Zpracování pomocí soustavy implicitních diferenciálních rovnic
a nástroje GEKKO <https://github.com/BYU-PRISM/GEKKO>

Transformátor nakrátko



The screenshot displays the Spyder Python IDE interface. The left pane shows a Python script named `temp.py` with the following code:

```
1 # Reseni prubehu proudu nakratko numerickou metodou
2
3 from gekko import GEKKO
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 Ukp=10.0
8 I0p=1.0
9 Un1=110.0
10 Un2=22.0
11 Snt=10.0
12
13 dP0p=0.3
14 dPkp=1.0
15
16 frekv=50.0
17 omega=2.0*np.pi*frekv
18
19 Rk=(dPkp/100.0)*(Un1**2/Snt)
20 Rk1=Rk/2.0
21 Zk=(Ukp/100.0)*(Un1**2/Snt)
22 Xs=np.sqrt(Zk**2-Rk**2)
23 Ls=Xs/omega
24 Ls1=Ls/2.0
25
26 Gfe=(dP0p/100.0)*(Snt/Un1**2)
27 Rfe=1/Gfe
28 Y0=(I0p/100.0)*(Snt/Un1**2)
29 Xh=1/np.sqrt(Y0**2-Gfe**2)
30 Lh=Xh/omega
31
32 # Reseni prubehu proudu nakratko numerickou metodou
33 Ukm=Ukp/100.0*Un1/np.sqrt(3)*np.sqrt(2)
34
35 m= GEKKO()
36 m.time = np.linspace(0.0, 0.1, 1000)
37 t = m.Param(value=m.time)
38 m.options.IMODE=4
39 i1 = m.Var(value=0.0)
40 i2 = m.Var(value=0.0)
41
42 # Ls1*i1' + Rk1*i1 + Lh*i1' + Lh*i2' = Ukm*sin(omega*t)
43 # -Lh*i1' + Lh*i2' + Ls1*i2' + Rk1*i2 = 0
44 m.Equation(Ls1*i1.dt() + Rk1*i1 + Lh*i1.dt() - Lh*i2.dt() == Ukm*m.sin(omega*t))
45 m.Equation(-Lh*i1.dt() + Lh*i2.dt() + Ls1*i2.dt() + Rk1*i2 == 0)
46
47 m.solve(disp=False)
48 plt.plot(t,i1)
49 plt.plot(t,i2)
50 plt.grid()
51 plt.show()
```

The right pane shows a plot of the current waveforms i_1 and i_2 over time. The x-axis represents time from 0.00 to 0.10 seconds, and the y-axis represents current from -0.075 to 0.125. The plot shows two oscillating waveforms, one in orange and one in blue, representing the currents in the transformer windings.

The bottom pane shows the IPython console output:

```
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.25.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/WORK/Python/CviceniPJS2021_5.py', wdir='C:/WORK/Python')
```

Zpracování pomocí soustavy implicitních diferenciálních rovnic
a nástroje GEKKO <https://github.com/BYU-PRISM/GEKKO>