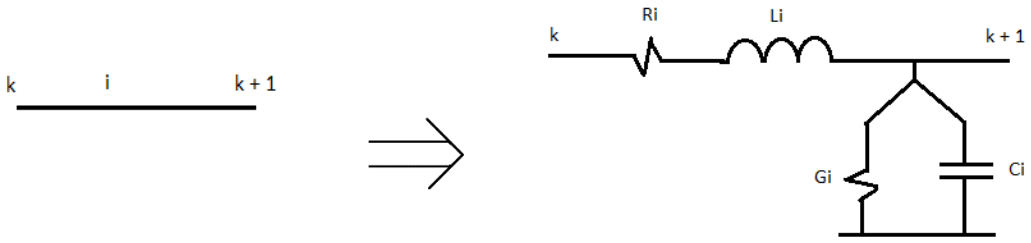
Definisanje udarnih karakteristika uzemljivača i način određivanja primenom programa u MATLAB-u

Pri projektovanju gromobranske zaštite potrebno je poznavati impedansu uzemljivača koju ovaj suprodstavlja odvođenju udarnih struja pri atmosferskim pražnjenjima. Ove struje se, kao što je poznato, karakterišu velikim apmlitudama i vremenom uspostavljanja ovih amplituda od nekoliko μs i trajanjem od nekoliko desetina μs. Pomenuta impedansa se naziva udarna impedansa. Ona je obično proporcionalna otporu rasprostiranja uzemljivača za struje industrijske učestanosti. Faktor proporcijalnosti ima veće vrednosti kod dužih uzemljivača, manjih struja odvođenja i malih specifičnih otpornosti tla. Kod dužih uzemljivača i malih specifičnih otpornosti tla dolazi do izražaja induktivnosti uzemljivača koja kod strmih strujnih talasa povećava impedansu uemljivača. Pri manjim vrednostima struje odvođenja uticaj joizacije tla i električnih pražnjena, koji dovode do sniženja otpornosti uzemljenja, nisu izraziti. Kod velikih vrednosti specifične otpornosti tla i udarnih struja, u tlu se uspostavljaju jaka elektična polja koja dovode do jonizacije tla, čak i do uspostavljanja pražnjena preko električnog luka. Do jonizacije dolazi pri električnom polju reda (2-4) kV/cm, a do lučnih pražnjena pri poljima većim od reda (6-10) kV/cm. Obe navedene pojave smanjuju otpornost rasprostiranja koje mogu biti manje nego pri strujama industrijske učestanosti.

Da bi se formirao program za određivanje udarnih karakteristika uzemljivača potrebno je prvo prvo imati matematički model uzemljivača.

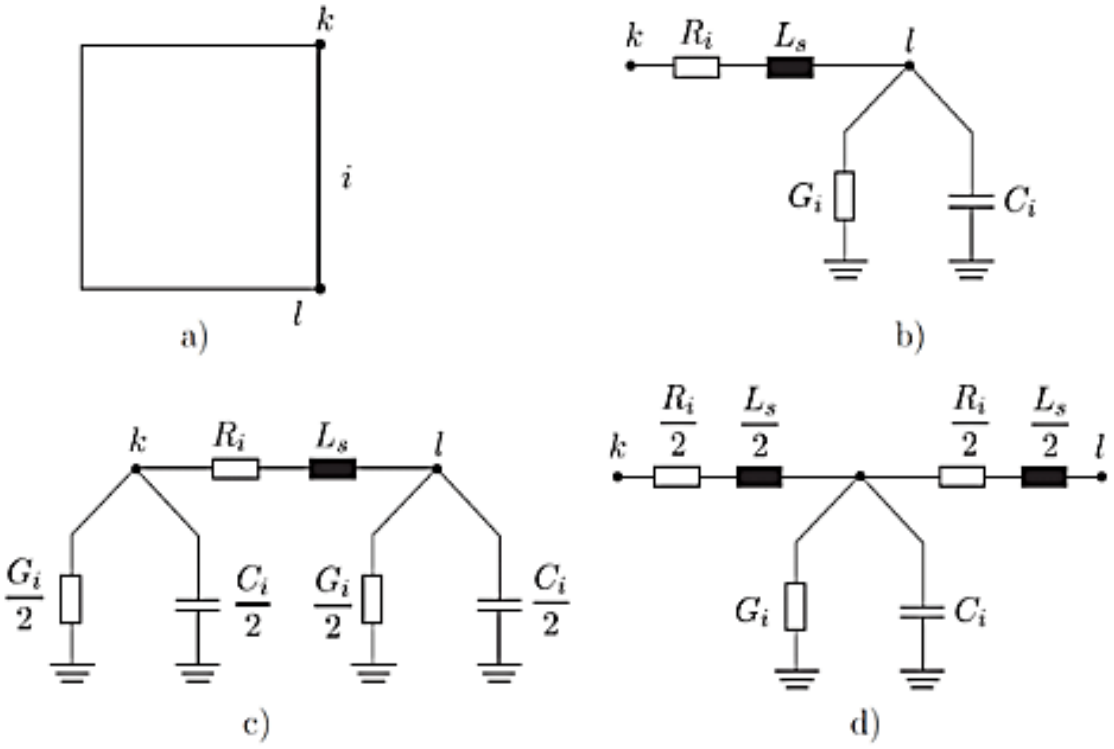
Jedan i-ti element uzemljivača se moze predstaviti ekvivalentnom šemom sa skoncentrisanim parametrima na sledeći način:



Slika: Zamenska šema jednog elementa uzemljivača

Matematički model uzemljivača, sa n čvorova i n grana, se moze predstaviti preko matrica na sledeći način:

$$\begin{bmatrix} \frac{du}{dt} \\ \frac{di}{dt} \end{bmatrix} = -\begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix}^{-1} \left(\begin{bmatrix} G & K \\ -K^T & R \end{bmatrix} \cdot \begin{bmatrix} u \\ i \end{bmatrix} - \begin{bmatrix} i_c \\ e_l \end{bmatrix} \right) \qquad (1)$$



Slika: a) Dispozicija uzemljivača sa označenim elementom i; b) Obrnuta Γ šema i-tog elementa uzemljivača; c) Π šema i-tog elementa uzemljivača; d) T šema i-tog elementa uzemljivača

Ovaj sistem se može napisati i u sledećem obliku:

$$C \cdot \frac{du}{dt} = G \cdot u + K \cdot i - ic \qquad (2)$$

$$L \cdot \frac{di}{dt} = -K^T \cdot u + R \cdot i - e_c \qquad (3)$$

Pri čemu je jednačina 2 jednačina po strujama, a 3 jednačina po naponu.

Parametri koji figurušu u sistemu:

[u] – vektor napona čije su dimenzije $n_{čvorova}$ x 1

[i] – vektor struja čije su dimenzije n_{grana} x 1

Ukoliko je $n_{čvorova} = n_{grana} = n$:

[G] – matrica provodnosti čije su dimenzije n x n

Za uzemljivač koji se posmatra pretpostavimo da je matrica [r], matrica sopstvenih i međusobnih otpornosti poznata:

$$[r] = \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{n1} & \cdots & r_{nn} \end{bmatrix} \qquad (4)$$

```
r=[15.8 2.68 1.48 2.68;
   2.68 15.8 2.68 1.48;
   1.48 2.68 15.8 2.68;
   2.68 1.48 2.68 15.8];
```

pomoću koje možemo izračunati matricu G, kao inverznu matricu:

$$[\mathrm{G}] = [r]^{-1} \qquad (5)$$

```
G=inv(r);
```

Provodnost je srazmerna sa dubinom ukopvanja uzemljivača *h* i obrnuto srazmerna otpornosti tla *ρ (G~h/ρ).*

[C] – matrica kapacitivnosti čije su dimenzije n x n

Matrica C se računa na sledeći način:

$$[\mathrm{C}] = \varepsilon \cdot \rho \cdot [G] \qquad (6)$$

```
C=8.85.*10.^(-6).*10.*100.*G;
```

Vrednosti kapacitivnosti se zadaju u μF, a kapacitivnost je direktno srazmerna permeabilnosti ε_r.

[L] – matrica induktivnosti čije su dimenzije n x n

Ova matrica je zadata na isti način kao r, i predstavlja spostvene i međusobne induktivnosti uzemljivača:

$$[L] = \begin{bmatrix} L_{11} & \cdots & L_{1n} \\ \vdots & \ddots & \vdots \\ L_{n1} & \cdots & L_{nn} \end{bmatrix} \qquad (7)$$

```
L=[15.7 0 -0.93 0;
   0 15.7 0 -0.93;
   -0.93 0 15.7 0;
   0 -0.93 0 15.7];
```

[R] – matrica rednih otpornosti uzemljivača čije su dimenzije n x n

$$R_i = \rho_i \cdot \frac{l_i}{S_i} = \frac{l_i}{\sigma_i \cdot S_i} = 0{,}0036\Omega \qquad (8)$$

ρ_i je specifična otpornost materijla od koga je napravljen uzemljivač, l_i je duzina i-tog dela uzemljivača, a S_i je poprečni presek i-tog dela uzemljivača.

Matrica R je matrica koja na glavnoj dijagonali ima vrednosti $R_i$dok su ostali elementi 0, formiramo na sledeći način:

$$[R] = \begin{bmatrix} R_i & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R_i \end{bmatrix} \qquad (9)$$

```
Ri=10/(55.56e6*50e-6);

R=eye(4).*Ri;
```

Ova matrica ne zavisi od dubine ukopavanja uzemljivača, kao ni od specifične otpornosti okolnog tla.

[i_c] – vektor struja ekvivalentnih strujnih izvora priključenih u čvorove uzemljivača, čije su dimenzije n x 1

[e_l] – vektor elektromotornih sila ekvivalentnih strujnih izvora priključenih na grane uzemljvača, čije su dimenzije n x 1

[K] – matrica koincidencije

$$[K] = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \qquad (10)$$

```
K=[1 0 0 -1;

  -1 1 0 0;

  0 -1 1 0;

  0 0 -1 1];
```

Kod elementa a_{ij}, i predstavlja broj čvora uzemljivača, a j broj grane. Ako grana:

- izlazi iz čvora taj element ima vrednot 1
- ulazi u čvora taj element ima vrednost -1
- ako nema kontakt sa čvorom taj element ima vrednost 0

Matematički model možemo prikazati i na sledeći način:

$$\begin{bmatrix} \frac{du}{dt} \\ \frac{di}{dt} \end{bmatrix} = -[A]^{-1}([B] \cdot [y] - [ii]) \qquad (11)$$

```
yprime=-inv(A)*(B*y-ii);
```

gde su:

[A] – matrica dimenzija 2n x 2n

$$[\mathrm{A}] = \begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix} \qquad (12)$$

```
A=[C zeros(4);
  zeros(4) L];
```

[B] – matrica dimenzija 2n x 2n

$$[\mathrm{B}] = \begin{bmatrix} G & K \\ -K^T & R \end{bmatrix} \qquad (13)$$

```
B=[G K;
  -K' R];
```

[y] – matrica dimenzija 2n x 1

$$[y] = \begin{bmatrix} u \\ i \end{bmatrix} \qquad (14)$$

[ii] – matrica dimenzija 2n x 1 i predstavlja početne uslove, koji su u ovom slučaju svi nula, osim struje u čvoru gde se injektira struja groma.

$$[ii] = \begin{bmatrix} i_c \\ e_l \end{bmatrix} \qquad (15)$$

```
ii=[5.*(exp((-0.693./30).*t)-exp((-2.746./1.5).*t));zeros(7,1)];
```

Udarne karakteristike uzemljivača:

- udarna impedansa uzemljivača:

$$Z(t) = \frac{u_i(t)}{i_i(t)} \qquad (16)$$

```
ig=[5.*(exp((-0.693./30).*t)-exp((-2.746./1.5).*t))];
z=y(:,1)./(ig+eps);
```

gde je i čvor injektiranja.

- konvencijalna impedansa:

$$Z_k = \frac{u_{imax}(t)}{i_{imax}(t)} \qquad (3.17)$$

- udarni koeficijent

$$\alpha(t) = \frac{Z(t)}{R_s} \qquad (18)$$

gde je R_s stacionarna otpornost.

```
rstac=1./(ones(4,1)'*inv(r)*ones(4,1));
alfa=z./rstac;
```

zk=max(y(:,1))./max(ig);

- konvencionalni kojeficijent

$$\alpha_k = \frac{Z_k}{R_s} \qquad (19)$$

alfak=zk./rstac;

Formiramo funkciju u program MATLAB sa ulaznim veličinama koje definišu posmatrani uzemljivač, a koji je u ovom slučaju kvadratnog oblika i sadrži četiri čvora i četiri grane, pomoću prethodno objašnjenog matematičkog modela.

$$yprime = -inv(A).*(B.*y - ii) \qquad (20)$$

$$A = \begin{bmatrix} C & zeros(4); \\ zeros(4) & L \end{bmatrix} \qquad (21)$$

$$B = \begin{bmatrix} G & K; \\ -K' & R \end{bmatrix} \qquad (22)$$

$$ii = [ig; zeros(7,1)] \qquad (23)$$

gde je i_g struja groma, u ovom slučaju injektirana u prvi čvor. Matrica ii ima dimenzije 8 x 1 i ukoliko je struja injektirana u i-ti čvor, element i1 će imati vrednost injektirane struje. Jednačina 20 predstavlja jednačinu 11, matematičkog modela, napisana u programu MATLAB, 21 ekvivalntna je jednačini 12, izraz 22 ekvivalentan je jednačini 13, a 23 jednačini 15.

Matematički model je napisan u obliku diferencijalnog sistema jednačina koji je potrebno rešiti da bismo dobili prikaz vremenske promene napona čvorova i struje grana, kao i udarne karakteristike uzemljivača. Funkcija kojom se rešava diferencijalni sistem jednačina je ode45. Ona praktično predstavlja metodu Runge-Kutta, dok broj 4 znači da je to metoda četvrtog reda, a 5 da je greška srazmerna petom koraku.

$$[t, y] = ode45('ulaz', [t_0\ t_k], zeros(8,1)) \qquad (24)$$

$'ulaz'$ predstavlja funkciju u kojoj smo definisali sistem, t_0 je početna, a t_k krajnja vrednost interval za rešavanje diferencijalnog sistema, dok $zeros(8,1)$ predstavlja početne uslove za rešavanje diferencijalnog sistema.

Crvenom bojom kod formula su naznačene komande u Matlabu.

Ulazni podaci su:

Dubina ukopavanja uzemljivača h = 0.5 m, specifična otpornost tla je ρ = 100 Ωm, poprečni presek bakarnog užeta uzemljivača je S = 50 mm^2. Dužina stranice kvadrata je l = 10 m, relativna dielektrična konstanta je ε_r = 10. Što se tiče udarne struje koja se injektira, njena amplitude je I = 5 kA, a T1 i T2 redom iznose 1.5 μs i 30 μs.

Kompletan Matlab kod je dat u nastavku:

1. **ulaz.m**

```
function yprime=ulaz(t,y);

r=[15.8 2.68 1.48 2.68;

   2.68 15.8 2.68 1.48;

   1.48 2.68 15.8 2.68;

   2.68 1.48 2.68 15.8];

G=inv(r);

C=8.85.*10.^(-6).*10.*100.*G;

L=[15.7 0 -0.93 0;

    0 15.7 0 -0.93;

    -0.93 0 15.7 0;

    0 -0.93 0 15.7];

K=[1 0 0 -1;

   -1 1 0 0;

   0 -1 1 0;

   0 0 -1 1];

A=[C zeros(4);

   zeros(4) L];

Ri=10/(55.56e6*50e-6);

R=eye(4).*Ri;

B=[G K;

   -K' R];

ii=[5.*(exp((-0.693./30).*t)-exp((-2.746./1.5).*t));zeros(7,1)];

yprime=-inv(A)*(B*y-ii);

return
```

izlaz.m

```
clear
```

```
[t,y]=ode45('ulaz', [0 10],zeros(8,1));
ig=[5.*(exp((-0.693./30).*t)-exp((-2.746./1.5).*t))];
z=y(:,1)./(ig+eps);
r=[15.8 2.68 1.48 2.68;
   2.68 15.8 2.68 1.48;
   1.48 2.68 15.8 2.68;
   2.68 1.48 2.68 15.8];
rstac=1./(ones(4,1)'*inv(r)*ones(4,1));
alfa=z./rstac;
zk=max(y(:,1))./max(ig);
alfak=zk./rstac;
subplot(2,2,1), plot(t,y(:,1:4)),grid
subplot(2,2,2), plot(t,y(:,5:8)),grid
subplot(2,2,3), plot(t,z,t,alfa),grid
subplot(2,2,4), plot(t,y(:,1),t,z,t,ig),grid
return
```

2. hevu.m

```
function yprime=hevu(t,y);
r=[15.8 2.68 1.48 2.68;
   2.68 15.8 2.68 1.48;
   1.48 2.68 15.8 2.68;
   2.68 1.48 2.68 15.8];
G=inv(r);
C=8.85.*10.^(-6).*10.*100.*G;
L=[15.7 0 -0.93 0;
    0 15.7 0 -0.93;
    -0.93 0 15.7 0;
    0 -0.93 0 15.7];
```

```
K=[1 0 0 -1;
   -1 1 0 0;
   0 -1 1 0;
   0 0 -1 1];
A=[C zeros(4);
   zeros(4) L];
Ri=10/(55.56e6*50e-6);
R=eye(4).*Ri;
B=[G K;
   -K' R];
ii=[1;zeros(7,1)];
yprime=-inv(A)*(B*y-ii);
return
```

hevi.m

```
clear
[t,y]=ode45('hevu', [0 5],zeros(8,1));
r=[15.8 2.68 1.48 2.68;
   2.68 15.8 2.68 1.48;
   1.48 2.68 15.8 2.68;
   2.68 1.48 2.68 15.8];
rstac=1./(ones(4,1)'*inv(r)*ones(4,1))
Zm=max(y(:,1));
R1=Zm
R2=Zm.*rstac./(Zm-rstac)
plot(t,y(:,1)),grid
return
```

3. **struja.m**

```
clear
t=0:350;
i=4000*sin(2*pi/50*t).*exp(-0.0057*t);
ii=abs(i);
iii=i.^2;
subplot(211),plot(t,i,t,ii),grid
subplot(212),plot(t,iii),grid
return
```

kolel.m

```
function f=kolel(t)
f=abs(4000.*sin(2.*pi./50.*t).*exp(-0.0057.*t)).*10.^(-6);
return
```

q.m

```
clear
q=quad('kolel',0,350,1e-5)
return
```

topimp.m

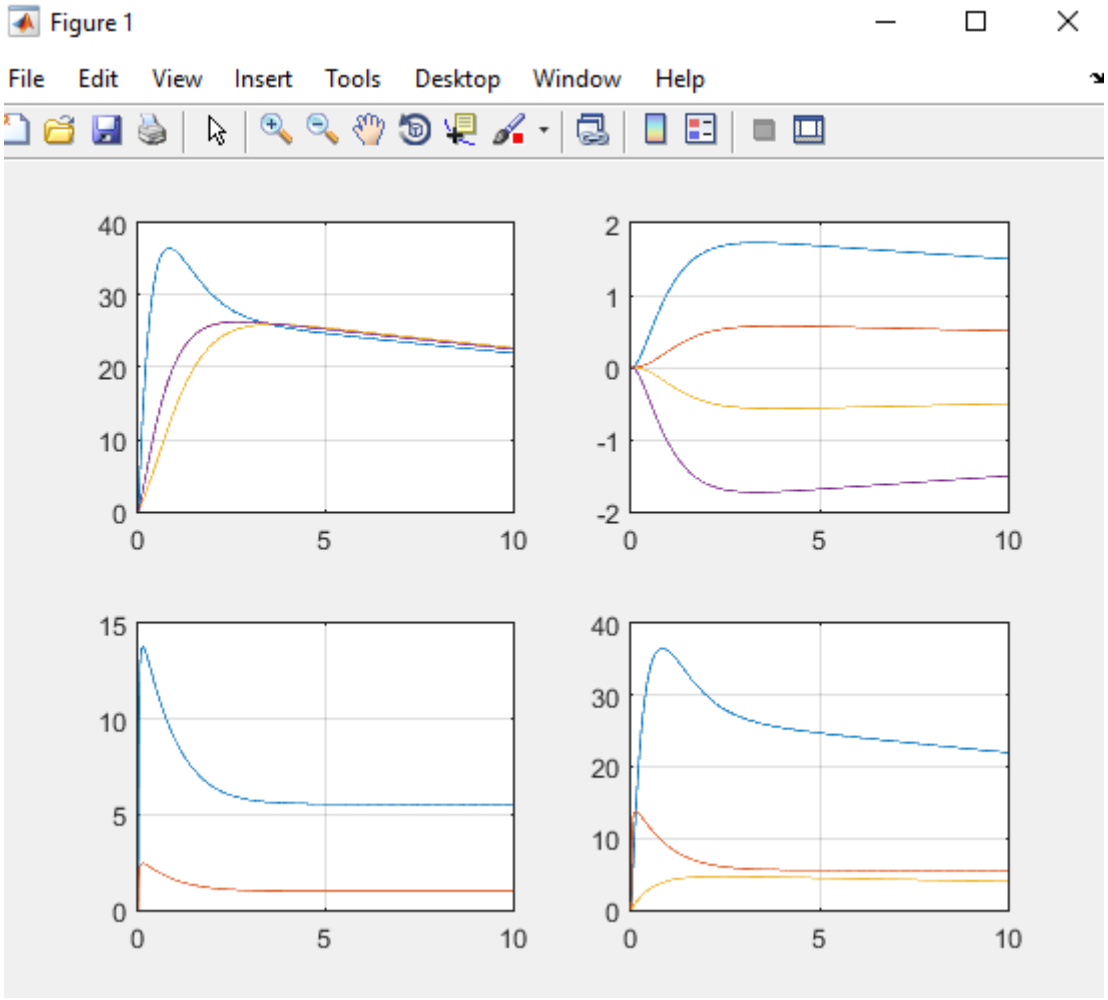
```
function f=topimp(t)
f=(4000.*sin(2.*pi./50.*t).*exp(-0.0057.*t)).^2.*10^(-6);
return
```

impuls.m

```
clear
impuls=quad('topimp',0,350,1e-5)
return
```

Rezultati Matlab koda mogu se videte na narednim slikama:

Slika: Vrednosti konvencionalne impedanse i konvencionalnog koeficijenta izračunate primenom programa MATLAB



Slika: Prikaz grafika dobijenih pomoću programa MATLAB

Osnove programa Python i konverzija koda iz MATLAB-a

(ko je zainteresovan neka proba, nije potrebno za ispit)

Python je interpreterski jezik – radi bez potrebe za kompajliranjem odnosno prevođenjem u izvršni oblik, interaktivni, objektno orjentisan - korisnici manipulišu strukturama podataka nazvanim objektom, programski jezik. Nastao je 1991. godine, a ime je dobio po istoimenoj seriji po kojoj je program i dobio ime "Leteći cirkus Montija Pajtona", imajući za cilj da prikaže da programiranje može da bude lagano i intresantno. Sintaksa jezika je lagana i razumljiva, omogućavajući korisniku da više vremna posveti problemu, nego izradi koda, a opet, ovo je moćan programski jezik. To dokazuje i to što ga koriste neke od najvećih kompanija, kao sto je YouTube, Google, Pinterest, Dropbox, Mozila, NASA kao i mnoge druge. Najveća prednost Pythona je sto je to besplatan, a opet ima širok spektar biblioteka, što pokriva sve od funkcija operativnog sistema do struktura podataka potrebnih za gradnju web-servera. Veliki izbor literature i snimaka koji pomažu pri savladavanju ovog programskog jezika. Veliki broj korisnika olakšava i pronalaženje rešenja velikog spektra problema.

Sintaksa Pythona je case sensitive, što znači da se u naredbama razlikuju mala i velika slova, ime promenljive ne sme početi brojem, a može se sastoati od malih, velikih slova kao i donje crte_. **Dodeljivanje vrednosti se vrši pomoću =,** a promenljive mogu biti tipa **int (celi brojevi), float (realni brojevi), bool (1 za tačno, 0 na netačno), string (slova)**…

Jednolinijski komenari počinju sa #, a viselinijske označavamo sa tri navodnika na početku i kraju komentara, koji mogu biti ‘ ili “.

U interaktivnom načinu rada, naredba se interpretira i izvrši čim se potvrdi njen unos tasterom Enter.

Naredba za ispisavanje je **print().** Ukoliko se ispisuje tekst, on mora biti stavljen pod navodnike (moze ‘ i “).

Na raspolaganju su naredbe **if, else, elif, kao i petlje while, for**. Raspolaže i opcijom definisanja funkcija, radi lakseg i sažetijeg pisanja koda.

Za konvertovanje koda iz MATLAB-a u Python potrebne su gore navedene opcije koje omogućava ovaj programski jezik kao i korišćenje biblioteka za formiranje nizova, matrica, iscrtavanja grafika i drugih naredbi.

Matrice A i B (21 i 22) za zadate vrednosti parametara matrica i vekotra (C, L, R, K, G) su formirane pomoću for petlji.

Za $n_{čvorova} = n_{grana} = n$ uzemljivača prikazan je primer kako je formirana matrica A:

```
A = np.zeros((2*n, 2*n))
for i in range(n):
    for j in range(n):
            A[i][j] = C[i][j]

for i in range(n, 2*n):
    for j in range(n, 2*n):
            A[i][j] = L[i-n][j-n]
```

Primetićemo da je bitno uvlačenje redova i da od toga zavisi koji deo koda prpada kojoj naredbi/petlji (sto se u drugim programskim jezicima obično označava pomoću zagrada {}),

kao i da je korišceno **np**. što je na početku koda uzeto kao skraćenica za biblioteku **nummpy** koja nam omogućava formiranje nizova i matrica u kodu.

Inverzna matrica se računa pomoću naredbe np.linalg.inv( ), sabiranje i oduzimanje se vrši operatorima + i -, a množenje i deljenje * I /, dok se množenje i deljenje matrica vrši pomoću a.dot(b) i np.divide(a,b).

Funkcija ode45 (3.24) koja se u MATLAB-u koristi za rešavanje diferencijalnog sistema jednačina ovde je zamenjena sledecim linjama koda:

```
def RK4_step(y, t, dt):
    k1 = S(y,t)
    k2 = S(y+0.5*k1*dt, t+0.5*dt)
    k3 = S(y+0.5*k2*dt, t+0.5*dt)
    k4 = S(y+k3*dt, t+dt)
    #return dt * S(y,t)
    return dt * (k1 + 2*k2 + 2*k3 + k4) /6

for t in time:
    y = y + RK4_step(y, t, delta_t)
```

gde je predstavlja jednačinu 20 u MATLAB-u

```
def S(y,t):
    return A_inv.dot(B.dot(y)-ii(t))
```

U narednom tekstu je kod u Python-u:

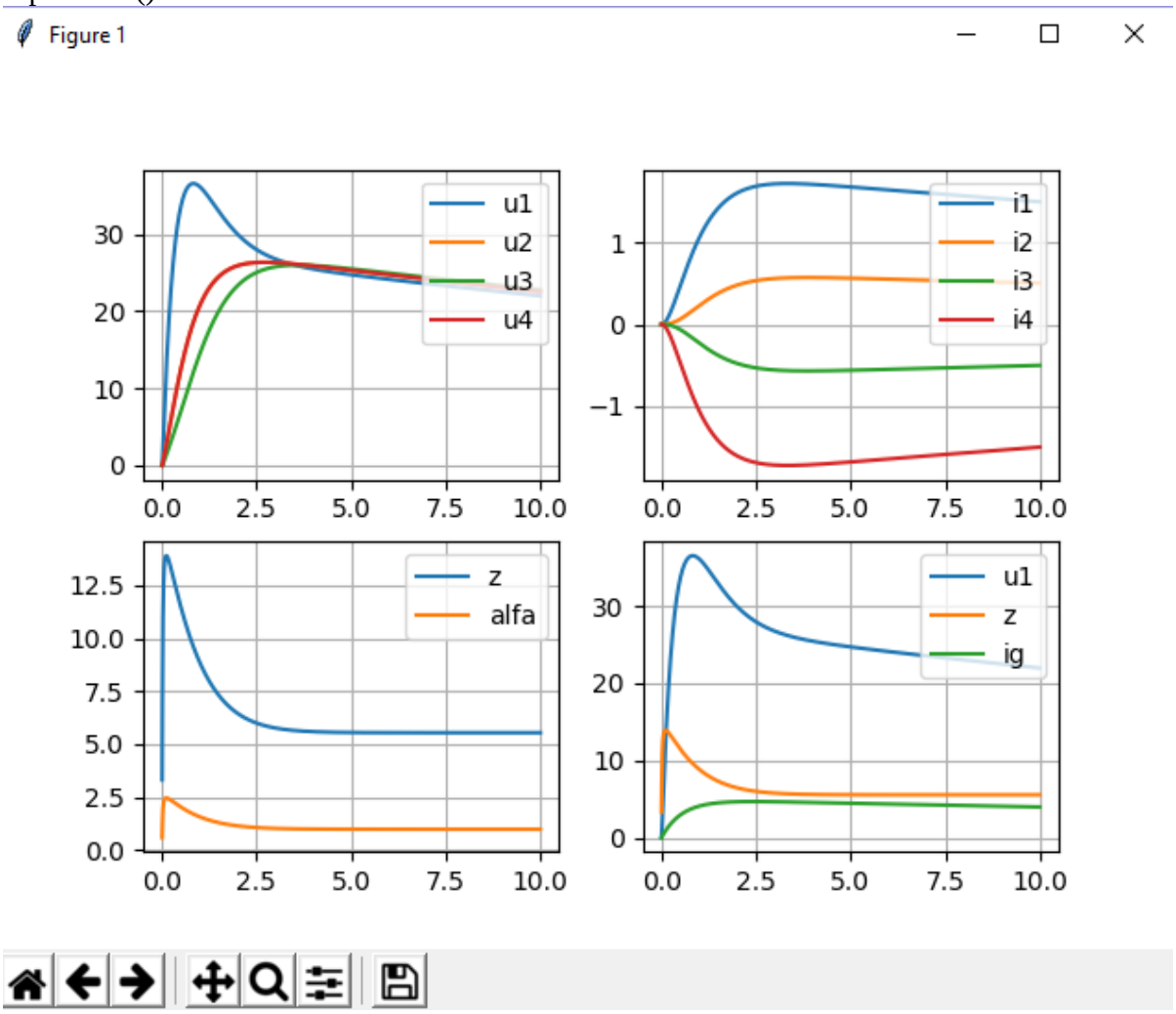
```
import time
import os
import numpy as np
from matplotlib import pyplot as plt
from math import *

def proracun():
  def S(y,t):
    return A_inv.dot(B.dot(y)-ii(t))

  #injektiranje struje u prvi cvor
  def ii(t):
    ii = np.array(np.zeros((2*n, 1)))
    T1 = 1*float(e7.get())
    T2 = 1*float(e8.get())
    I = 1*float(e6.get())
    ii[0][0] = I * (exp((-0.693/T2)* t )-exp((-2.746/T1)* t ))
    return ii

  #Runge-Kutta metoda za resavanje dif jna
  def RK4_step(y, t, dt):
    k1 = S(y,t)
    k2 = S(y+0.5*k1*dt, t+0.5*dt)
    k3 = S(y+0.5*k2*dt, t+0.5*dt)
```

```
    k4 = S(y+k3*dt, t+dt)
    #return dt * S(y,t)
    return dt * (k1 + 2*k2 + 2*k3 + k4) /6

# promenljive

delta_t = 0.001
time = np.arange(0.0, 10.0, delta_t)
n = 4

#r = f(rho,h)
r1 = np.array([
    [15.8, 2.68, 1.48, 2.68],
    [2.68, 15.8, 2.68, 1.48],
    [1.48, 2.68, 15.8, 2.68],
    [2.68, 1.48, 2.6, 15.8]
])

hn = 1*float(e1.get())
hs = 0.5
rhon = 1*float(e2.get()) #rho zemlje
rhos = 100

r = r1*(hs/hn)*(rhon/rhos)

G = np.linalg.inv(r)

#C = f(epislonr,h)
epsilon = 1*float(e5.get())
C = 8.85*(10**(-6))*epsilon*rhon*G

L = np.array([
    [15.7, 0, -0.93, 0],
    [0, 15.7, 0, -0.93],
    [-0.93, 0, 15.7, 0],
    [0, -0.93, 0, 15.7]])

K = np.array([
    [1, 0, 0, -1],
    [-1, 1, 0, 0],
    [0, -1, 1, 0],
    [0, 0, -1, 1]
])

K_p = (-1) * np.transpose(K)

l = 1*float(e4.get())
Scu = 1*float(e3.get())
Ri = l/(55.56e6*Scu*1e-6)
R = np.eye(n, dtype=int)*Ri
```

```
rstac = 1/np.ones(4).T.dot((np.linalg.inv(r)).dot(np.ones(4)))

# pocetne vrednosti
y = np.zeros(8)   # [naponi 0:4, struje 4:8]

A = np.zeros((2*n, 2*n))
for i in range(n):
    for j in range(n):
        A[i][j] = C[i][j]

for i in range(n, 2*n):
    for j in range(n, 2*n):
        A[i][j] = L[i-n][j-n]

B = np.zeros((2*n, 2*n))
for i in range(n):
    for j in range(2*n):
        if j < n:
            B[i][j] = G[i][j]
        else:
            B[i][j] = K[i][j-n]

for i in range(n, 2*n):
    for j in range(2*n):
        if j < n:
            B[i][j] = K_p[i-n][j]
        else:
            B[i][j] = R[i-n][j-n]

A_inv = (-1)*np.linalg.inv(A)

u1 = []
u2 = []
u3 = []
u4 = []

i1 = []
i2 = []
i3 = []
i4 = []

z =[]
# time-stepping resenje

for t in time:
    y = y + RK4_step(y, t, delta_t)
    u1.append(y[0][0])
    u2.append(y[1][0])
    u3.append(y[2][0])
```

```
    u4.append(y[3][0])
    i1.append(y[4][0])
    i2.append(y[5][0])
    i3.append(y[6][0])
    i4.append(y[7][0])

T1 = 1*float(e7.get())
T2 = 1*float(e8.get())
I = 1*float(e6.get())

ig = I* (2.71828**((-0.693/T2)* time )-2.71828**((-2.746/T1)* time ))
u1array = np.asarray(u1)

z = np.divide(u1,ig)
alfa = z/rstac
u1max = np.amax(u1)
igmax = np.amax(ig)
zk = u1max/igmax
zkz = '{:.3f}'.format(zk)#svodjenje na 3 decimala
alfak = zk/rstac
alfakz = '{:.3f}'.format(alfak)

textzk.insert(0.0, zkz)
textalfak.insert(0.0, alfakz)


# grafici
t = [i for i in time]

plt.subplot(221)
plt.plot(t,u1)
plt.plot(t,u2)
plt.plot(t,u3)
plt.plot(t,u4)
plt.legend(['u1', 'u2' ,'u3', 'u4' ], loc='upper right')
plt.grid(True)

plt.subplot(222)
plt.plot(t,i1)
plt.plot(t,i2)
plt.plot(t,i3)
plt.plot(t,i4)
plt.legend(['i1', 'i2' ,'i3', 'i4' ], loc='upper right')
plt.grid(True)


plt.subplot(223)
plt.plot(t,z,t,alfa)
plt.legend(['z', 'alfa' ], loc='upper right')
plt.grid(True)
```

```
plt.subplot(224)
plt.plot(t,u1,t,z,t,ig)
plt.legend(['u1', 'z' ,'ig'], loc='upper right')
plt.grid(True)

plt.show()
```



Slika: Rezultati Python koda