Optimizacija alokacije pasivnih filtera u distributivnim mrežama evolucijskom metodom

Žgela, Matej

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

OPTIMIZACIJA ALOKACIJE PASIVNIH FILTERA U DISTRIBUTIVNIM MREŽAMA EVOLUCIJSKOM METODOM

Diplomski rad

Matej Žgela

Osijek, 2018

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 13.07.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Matej Žgela
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika
Mat. br. studenta, godina upisa:	D 904, 25.09.2017.
OIB studenta:	70346921686
Mentor:	Doc.dr.sc. Marinko Barukčić
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv.prof.dr.sc. Željko Hederić
Član Povjerenstva:	Matej Žnidarec
Naslov diplomskog rada:	Optimizacija alokacije pasivnih filtera u distributivnim mrežama evolucijskom metodom
Znanstvena grana rada:	**Elektroenergetika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	U radu je potrebno opisati pojavu i uzroke viših harmonika u distributivnim mrežama. Također, opisati mjere i uređaje (izvedbu, princip rada...) uređaja (filtera, kondenzatorskih baterija) za smanjenje viših harmonika u mreži. Opisati evolucijske metode kao metode globalne optimizacije. Obaviti par primjera simulacije određivanja optimalne alokacije pasivnih filtera na primjerima mreže iz literature. Za simulacije koristiti gotove alate za analizu mreža i evolucijsku optimizaciju, npr. OpenDSS i PYTHON (preporuka), a može i PSAT i MATLAB ili OpenDSS i PYTHON. Rezervirano: Matej Žgela
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	13.07.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 18.07.2018.

Ime i prezime studenta:	Matej Žgela
Studij:	Diplomski sveučilišni studij Elektrotehnika
Mat. br. studenta, godina upisa:	D 904, 25.09.2017.
Ephorus podudaranje [%]:	26

Ovom izjavom izjavljujem da je rad pod nazivom**: Optimizacija alokacije pasivnih filtera u distributivnim mrežama evolucijskom metodom**

izrađen pod vodstvom mentora Doc.dr.sc. Marinko Barukčić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. Uvod

Osnovna zadaća elektroenergetskog distributivnog sustava je opskrbiti sva trošila kvalitetnim strujama i naponima unutar traženi okvira danih normama. Jedan od glavnih problema koji se javlja pri opskrbi električne energije su nelinearna trošila koja onečišćuju izvorne sinusne struje i napone, odnosno koja stvaraju harmonička izobličenja istih. U suvremenim distribucijskim mrežama iz razloga sve veće prisutnosti nelinearnih uređaja energetske elektronike koji pri svom radu stvaraju više harmonike, stoga na taj način onečišćuju mrežu izražen je navedeni problem. Neki od spomenutih uređaja energetske elektronike su ispravljači, pretvarači, izmjenjivači, nelinearni elementi i sl.

Navedena harmonička izobličenja na elementima distributivne mreže većinom stvaraju probleme koji se očituju kroz nepotrebno dodatno zagrijavanje komponenti distributivne mreže i trošila, te samim time smanjuju kvalitetu i kapacitete prijenosa distribuirane električne energije. Iz razloga što je izričito poželjno izbjeći takve gubitke nastoji se u najvećoj mogućoj mjeri smanjiti razina harmoničkih izobličenja. Budući da trošila koja ih proizvode najčešće nije moguće zamijeniti boljima, preostaje samo mogućnost da se u najvećoj mogućoj mjeri ukloni njihovo daljnje prenošenje od izvora.

Za potiskivanje harmoničkih izobličenja u mreži koriste se hibridni, pasivni i aktivni filtri. Aktivni filtri koriste se kod trošila pri kojima je intenzitet viših harmonika nepoznat ili promjenjiv, stoga ne projektiraju za točno određene harmonike već se njihov princip rada bazira na mjerenju intenziteta viših harmonika i njihovoj odgovarajućoj eliminaciji. Pasivni filtri koriste se kod trošila gdje viši harmonici imaju relativno poznat intenzitet te se projektiraju shodno tim očekivanjima. Značajnu razliku između aktivnih i pasivnih filtra stvara cijena, gdje su aktivni filtri daleko skuplji za ugradnju.

Sa razvojem današnjeg distributivnog sustava javlja se problem mreža sa velikim brojem čvorova gdje sa svakim čvorom raste i broj izvora viših harmonika. Ugradnjom pasivnih filtra u sve čvorove distributivnih mreža postigao bi se jako dobar rezultat u uklanjanju viših harmonika ali taj način rješavanja problema ne bih bio ekonomičan. Kako bi se smanjio udio viših harmonika u distributivnim mrežama s više čvorova ugrađuju se pasivni filtri u samo neke od čvorova, a izbor čvorova koji sadrže pasivne filtre bit će opisan u nastavku.

2. Zadatak diplomskog rada

Zadatak ovog rada je ukazati na mogućnost ugradnje manjeg broja pasivnih filtra nego što ima čvorova u nekoj distributivnoj mreži s visokom učinkovitošću pri uklanjanju viših harmonika. Iz toga razloga će na početku biti objašnjeni uzročnici pojave harmonika u distributivnim mrežama te detaljno objašnjenje njihova nastajanja. Nadalje, detaljnije ćemo opisati pasivne filtre i objasniti način njihova rada. Zatim ćemo opisati način optimizacije pomoću genetskog algoritma, što predstavlja genetski algoritam te što bi predstavljala evolucijska metoda. U konačnici će biti odrađena računalna simulacija distributivne mreže sa više čvorova te simuliranim višim harmonicima pomoću programskog paketa OpenDSS i izrađen genetski algoritam u programskom paketu Python, koji će zajedničkom komunikacijom sa OpenDSS-om prikazati čvor sa najboljim uvjetima za smještaj pasivnog filtra.

3. Definiranje harmonika u izmjeničnim sustavima napajanja

Harmonici su prezentirani kao sinusoide koje imaju frekvenciju cijelog višekratnika bazne frekvencije.

$$f_h = n * f_n \tag{3-1}$$

Gdje je:

f_h- red harmonika

n – cijeli broj

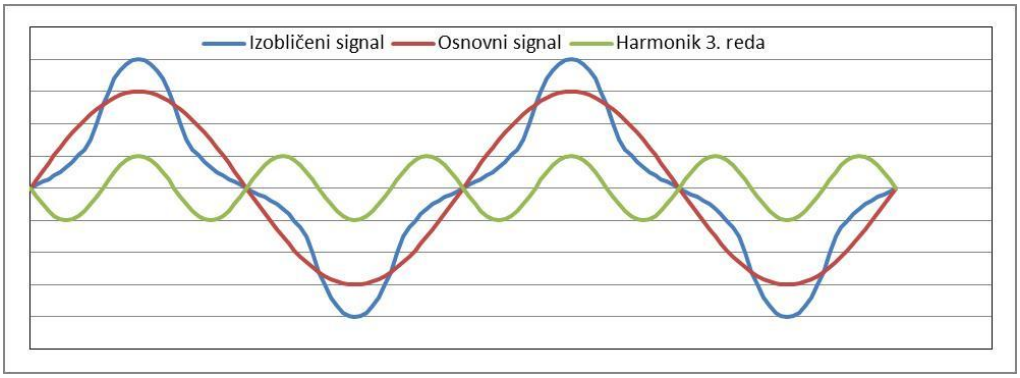
f_n – nazivna frekvencija sustava (50Hz)

Dakle za sustav napajanja na frekvenciji 50Hz, harmonici su komponente frekvencija:

Tablica 3.1. *Prikaz odnosa harmoničkog reda i frekvencija[21]*

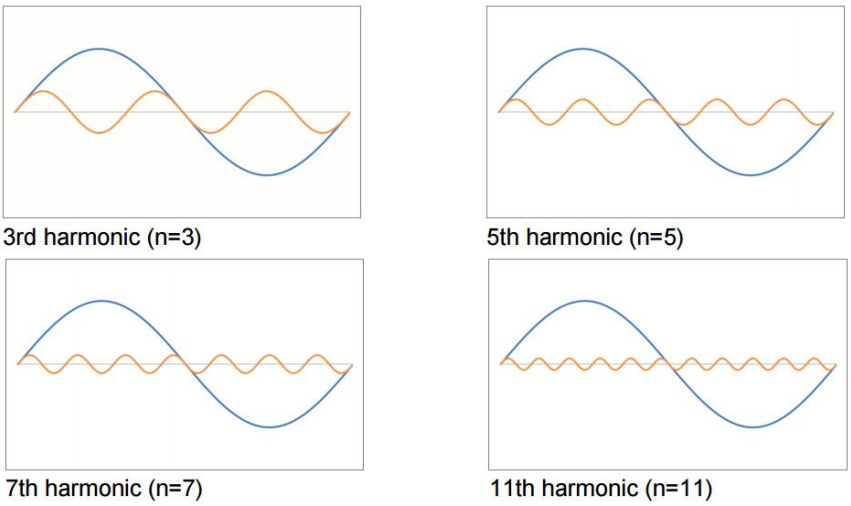
f_h	**Frekvencija (Hz) u 50Hz sustavu napajanja**
1	50
3	150
5	250
7	350
11	550
13	650
n	50*n

Zbog svoje periodičnosti izobličeni signal se može predstaviti harmonicima, tj. sinusoidalnim komponentama. Na primjer, ako je osnovna frekvencija distributivne mreže 50 Hz treći harmonik će biti frekvencije 150 Hz. Primjer koji možemo vidjeti na slici 3.1. predstavlja zbroj osnovnog signala i njegovog trećeg harmonika, taj zbroj rezultira izobličenim signalom.



Slika 3.1. *Valni oblici osnovnog signala struje trošila trećeg harmonika[21]*

Na sljedećoj slici su prikazani idealni valni oblici 50Hz sustava napajanja sa komponentama 3,5,7 i 11 harmonika



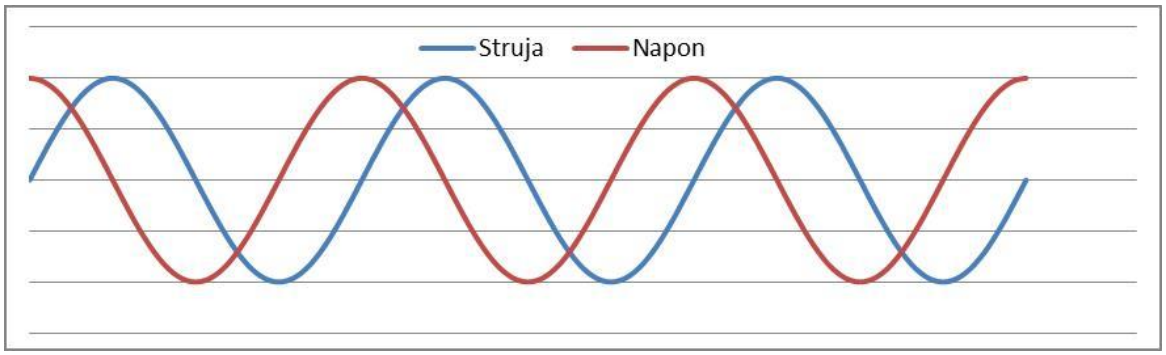
Slika 3.2. *Idealni valni oblici 50 Hz sustava napajanja sa komponentama 3,5,7 i 11 harmonika[21]*

Savršen napon električne mreže bi bio kad bi se njegova amplituda mijenjala iz pozitivnog polariteta u negativni 50 puta u sekundi (50Hz), odnosno kada ne bi bilo utjecaja viših harmonika. Harmonici mogu biti prisutni bilo da se radi o naponu ili struji. Određenim spektrom harmonika u odnosu na referentni sinusoidalni oblik opisuju se sva odstupanja. Svaki harmonik opisan je svojom amplitudom, frekvencijom i početnim kutom. Najčešće se harmoničke amplitude izražavaju kao postotne vrijednosti osnovnog harmonika, a velik postotni udio viših harmonka deformira sinusni oblik signala.[21]

4. Harmonici i njihov utjecaj na distributivnu mreži

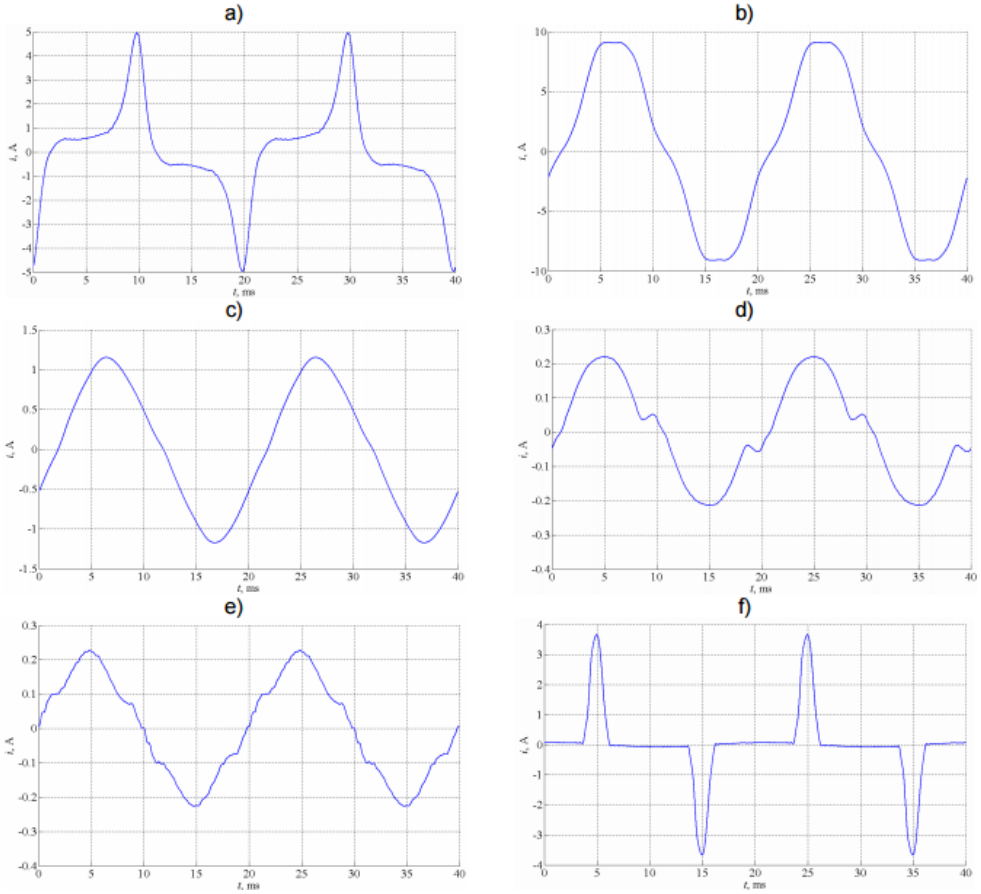
4.1. Valni oblici linearnih i nelinearnih trošila u NN mrežama

Nelinearna trošila u kućanstvima, ali i u industrijskim postrojenjima uzrokuju pojavu harmonika u distributivnim mrežama. U idealnom slučaju valni oblici napona i struje će imati oblik savršene sinusoide. Ukoliko se radi o linearnom trošilu bit će ovakav izgleda struje i napona. Općenito linearno trošilo jest trošilo kod kojeg je oblik struje jednak obliku napona što bi u slučaju sinusnog napona i struje elektroenergetskog sustava značilo da će struja i napon linearnog trošila također imati sinusoidalni oblik. Neka od linearnih trošila su elektromotori, grijači i žarulje s žarnom niti Primjer valnog oblika struje i napona na linearnom trošilu može se vidjeti na slici 4.1.[21]



Slika 4.1. *Valni oblik struje i napona linearnog trošila*

Harmonička izobličenja se pojavljuju ukoliko nelinearna trošila valnog oblika struje ne prate valni oblik napona. Najčešća nelinearna trošila su uređaji napajani elektroničkim sklopovima kao što su klima uređaji i elektromotorni pogoni sa promjenjivom brzinom vrtnje, osobna računala, pisači, fotokopirni uređaji, štedne žarulje itd. Primjeri valnih oblika struje nelinearnih trošila dani su na slici 4.2. [2][21]



Slika 4.2. *Valni oblik struje a) struje magnetiziranja transformatora, b) hladnjaka sa zamrzivačem, c) klima uređaja, d) jednofaznog ispravljača, e) fluorescentne cijevi s elektromagnetskom prigušnicom, f) fluorescentne cijevi s elektroničkom prigušnicom[20]*

Na gornjoj slici prikazano je kako nelinearna trošila valnog oblik postaju izobličeni, stoga se njihovo izobličenje može predstaviti harmonicima, odnosno zbrojem sinusoida određene amplitude i frekvencije.

Izobličeni valni oblik može biti simetričan i nesimetričan. Simetričan je ako mu je pozitivni dio periode identičan negativnom i tada sadrži samo neparne harmonike. Nesimetričan je ukoliko mu pozitivni dio periode nije isti kao negativni i tada sadrži parne i neparne harmonike. Nesimetričan je i u slučaju da je valni oblik pozitivnog i negativnog dijela periode isti, ali ima određenu istosmjernu komponentu (engl. *DC offset*). [21]

4.2. Utjecaj harmonika na distributivni sustav

Opterećenje distributivne mreže sa nelinearnim trošilima u vidu uređaja s elektroničkim napajanjem (Računala, TV uređaji, punjači za mobitele...) drastično se povećao proteklih godina, a tendencija porasta vidljiva je u cijelom svijetu. Izobličenje napona paralelno raste sa povećanjem korištenja ovih uređaja.

Najbolji način za opis utjecaja harmonika na distributivnu mrežu i uređaje je dijeljenje po njihovoj trajnosti na one sa trenutnim i dugotrajnim učinkom.

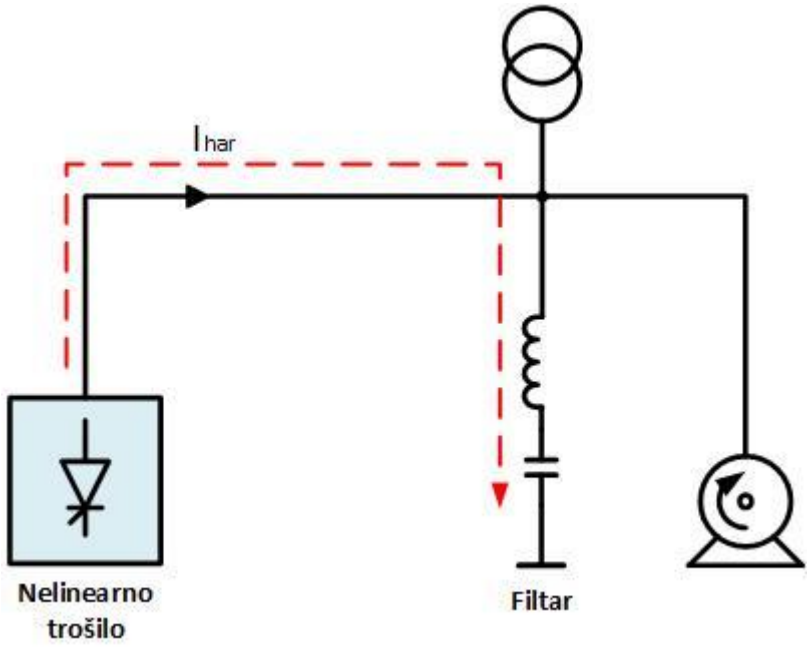
Trenutni učinak harmonika su kvarovi i greške na uređajima izloženim velikim harmoničkim distorzijama. Dugotrajni učinak harmonika je termičke prirode. Harmonici stvaraju povećanje temperature u distributivnim mrežama kao i u uređajima, a visoka temperatura elektroničkih uređaja i električnih motora, transformatora i vodiča, znači veće gubitke i kraći životni vijek uređaja.[21]

Među najznačajnijim problemima utjecaja harmonika su:

- Pregrijavanje nultog vodiča
- Pregrijavanje transformatora
- Pregrijavanje i veliki gubici električnih motora
- Utjecaj na prekidače i zaštitne releje
- Utjecaj na elektronsku opremu
- Utjecaj na telekomunikacije
- Dielektrično naprezanje kondenzatora za kompenzaciju jalove energije
- Distorzija mrežnog napona uslijed pojave harmoničkih struja

5. Pasivni filtri

Princip rada pasivnih filtra odvija se na način da se paralelno nelinearnom trošilu spoji odgovarajući broj LC krugova. Svaki LC krug je ugođen na frekvenciju jednog harmonika koji se želi anulirati i u slučaju pojave harmonika te frekvencije, harmonik biva kratko spojen na nulti potencijal čime je spriječeno njegovo daljnje širenje u mrežu. Svi LC krugovi pasivnog filtra međusobno su spojeni, paralelno trošilu kao što se može vidjeti na slici 5.1. Tipična primjena ovih filtara vidljiva je u industriji gdje snaga nelinearnih trošila prelazi 500 kVA, u slučaju da mreža zahtjeva kompenzaciju jalove snage ili gdje se harmoničko izobličenje struje i napona treba držati u dozvoljenim granicama.[21]



Slika 5.1. *Shema spajanja pasivnog filtra[1]*

Pasivni filtri svoju idealnu primjenu nalaze u postrojenjima gdje nelinearna trošila izazivaju značajna harmonička izobličenja i gdje je potrebno spriječiti širenje tih harmoničkih izobličenja dalje u elektroenergetski sustav. Pasivni filtri su jeftiniji u odnosu na ostale uređaje istih primjena budući da se sastoje samo od pasivnih komponenti. To su kondenzatori, zavojnice i otpornici čije karakteristične vrijednosti ovise o frekvenciji harmonika koji se želi prigušiti.

Frekvencija na koju su ugođeni pasivni filtri trebala bi biti što bliža frekvenciji harmonika koji se želi prigušiti tim filtrom. Također treba imati na umu da rezonantna frekvencija energetskog sustava treba biti udaljena što je moguće više od frekvencije bilo kojeg značajnog

harmonika kako ne bi moglo doći do ozbiljnih oštećenja unutar samog energetskog sustava. Stoga, frekvencija na koju je ugođen pasivni filtar treba biti nešto manja od frekvencije željenog harmonika kako bi sustav bio siguran u slučaju da uslijed vibracija, starenja ili temperaturnih promjena neka od komponenti sustava promjeni svoju vrijednost i time pomakne rezonantnu frekvenciju energetskog sustava bliže frekvenciji nekog od harmonika.

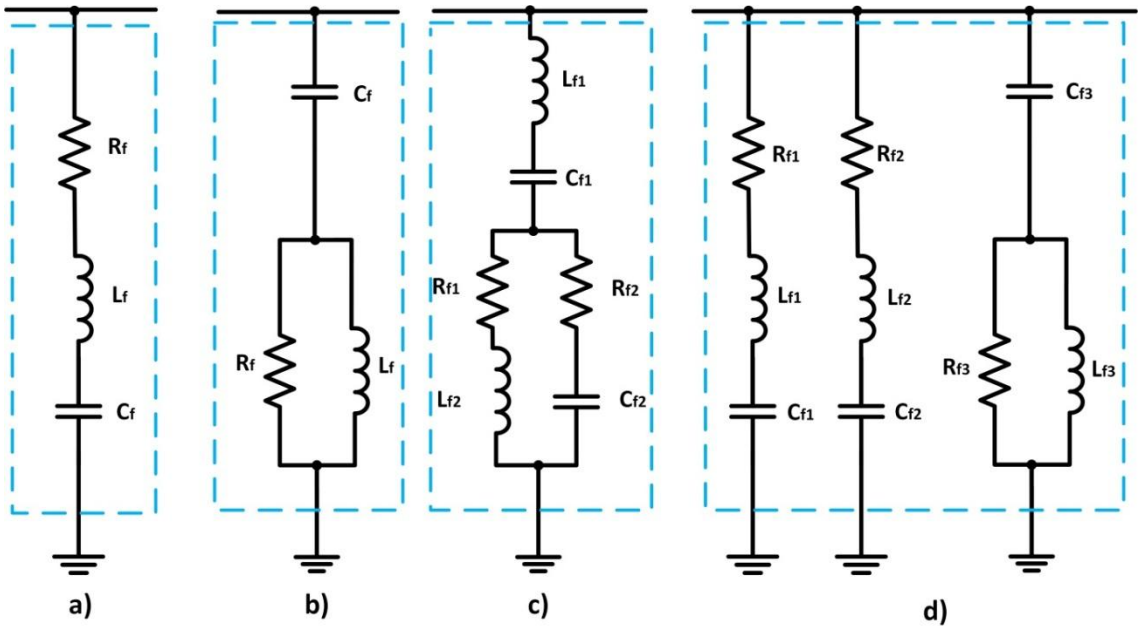
Prilikom dodavanja ili projektiranja pasivnih filtra u praksi se obično kreće od harmonika najnižeg reda, odnosno prvo se dodaje filtar petog, a zatim sedmog harmonika.

Postoje razne izvedbe pasivnih filtara za jednofazne i trofazne sustave u serijskoj ili paralelnoj konfiguraciji. Paralelna izvedba najčešće se koristi i omogućuje niski otpor prilikom protoka strujnih harmonika prema uzemljenju. Prilikom protoka strujnog harmonika kroz pasivni filtar u paralelnoj izvedbi prenosi se iznimno mali dio ukupne struje sustava, što je poželjno. Nasuprot tome, u slučaju serijske izvedbe prenosi se cijela struja sustava. Osim navedenih prednosti paralelne izvedbe pasivnih filtara, dodatna pogodnost je niža cijena pa su zbog toga najčešće korišteni u praksi.

Odabir paralelne ili serijske topologije filtra vrši se ovisno o tome je li dominantni izvor harmonika strujnog ili naponskog karaktera.[21]

5.1. Vrste pasivnih filtara

Konfiguracija filtra koja se odabire prilikom projektiranja ovisi o frekvencijskom spektru i prirodi harmoničkog izobličenja. Na slici 5.2. mogu se vidjeti neki od najčešće korištenih tipova pasivnih filtara.[21]



Slika 5.2. *Najčešći tipovi pasivnih filtara*

Pojasno-propusni filtar drugog reda (slika 5.2.a) najčešće se koristi za prigušenje jednog dominantnog harmonika nižeg reda. No, većina nelinearnih trošila ne stvara samo jedan nego više harmonika i zbog toga jedan filtar nije dovoljan da bi se prigušili svi značajniji harmonici. Postoji nekoliko pristupa rješenju ovog problema, prvi je da se koriste dva pojedinačno ugođena filtra s identičnim karakteristikama kao što je prikazano konfiguracijom na slici 5.2.c, Odnosno da se u slučaju više značajnih harmonika nižeg reda koristi kombinacija više takvih pojedinačno ugođenih filtara tako da se svaki filtar ugodi na frekvenciju određenog harmonika krenuvši on onog najnižeg reda, primjerice petog.
Ovakav pristup nije pogodan ako je broj harmonika nižeg reda velik ili se radi o većem broju harmonika višeg reda.

Drugi pristup ovom problemu je korištenje visoko-propusnog filtra drugog reda (slika 5.2.b) ugođenog na način da je rezonantna frekvencija ispod frekvencije dominantnog harmonika najnižeg reda.

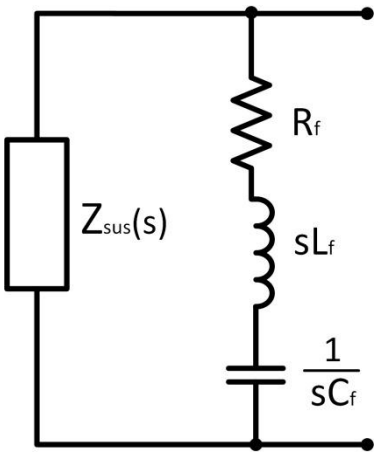
Treći pristup je implementirati složeni filtar (slika 5.2.d) sastavljenih od dvije ili više paralelnih grana koje čine pojasno-propusni filtri ugođeni na frekvencije nižih harmonika te visoko-propusni filtar ugođen na frekvencije harmonika višeg reda.[21]

5.1.1. Pojasno-propusni filtar drugog reda

Predstavlja serijsku kombinaciju kondenzatora C_f, zavojnice L_f i malom prigušnog otpornika R_f. Prigušni otpornik uglavnom se stavlja zbog unutrašnjih otpora kondenzatora i zavojnice. Ovaj filtar ugađa se na frekvenciju jednog harmonika niskog reda. Iz razloga što se inače u energetskom sustavu postrojenja nalazi nekoliko značajnih harmonika nižeg reda postavlja se nekoliko paralelno spojenih pojasno-propusnih filtara drugog reda ugođenih na frekvencije pojedinog harmonika. Dva vrlo važna parametra pojasno-propusnog filtra su: rezonantna frekvencija ω_0 i faktor kvalitete filtra Q prikazani formulama 5.1. i 5.2. Pri čemu X_0 predstavlja reaktivni otpor na rezonantnoj frekvenciji A pojačanje.[21]

$$Z = R + j\omega_0 L + \frac{1}{j\omega_0 C} = R \quad => \omega_0 = \frac{1}{\sqrt{LC}} \tag{5-1}$$

$$X_0 = \omega_0 L = \frac{1}{\omega_0 C} = \sqrt{\frac{L}{C}} => Q = \frac{X_0}{R} = \frac{1}{R}\sqrt{\frac{L}{C}} \tag{5-2}$$



Slika 5.3. *Model sustava sa pojasno-propusnim filtrom drugog reda*

Ako se za shemu sa slike 5.3. pretpostavi da je sustav Z_{sus} čisti induktivni tada za prijenosne funkcije filtra vrijede izrazi 5.3 i 5.4.

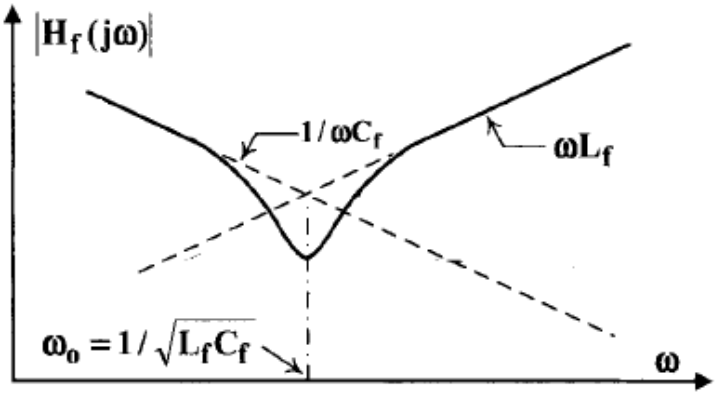
$$H_f(s) = Z_f(s) = \frac{U_f(s)}{I_f(s)} = R_f + sL_f + \frac{1}{sC_f} = \frac{A}{s}\left[\left(\frac{s}{\omega_0}\right)^2 + \frac{s}{\omega_0}\frac{1}{Q} + 1\right] \tag{5-3}$$

$$H_{cds}(s) = \frac{R_f + sL_f + \frac{1}{sC_f}}{R_f + sL_f + \frac{1}{sC_f} + sL_{sus}} \tag{5-4}$$

Prijenosna funkcija impedancije filtra je grafički prikazana na slici 5.4. Na niskim frekvencijama dosta manjim od rezonantne dominira otpor kapaciteta jer vrijedi da je $|1/\omega C_f| \gg |R + j\omega L_{sus}|$. Stoga, impedancija filtra na tim frekvencijama je dosta velika i

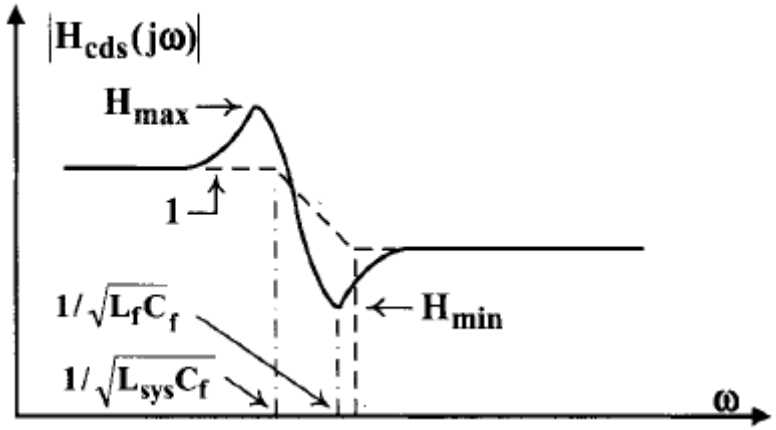
uglavnom ovisi o vrijednosti kapaciteta budući da izraz $H_f(s) \approx 1/\omega C_f$ predstavlja asimptotu funkcije na niskim frekvencijama.

Na visokim frekvencijama puno većim od rezonantne dominira otpor zavojnice jer vrijedi da je $|j\omega L_{sus}| \gg |R + 1/\omega C_f|$. Dakle, prijenosna funkcija impedancije filtra praktički se može aproksimirati izrazom $H_f(s) \approx j\omega L_{sus}$ što ujedno predstavlja asimptotu funkcije na visokim frekvencijama. Na rezonantnoj frekvenciji impedancija sustava poprima vrijednost otpora R_f, dakle, vrijedi da je prijenosna funkcija impedancije $H_f(s) = R_f$. Sukladno tome biti će veće prigušenje harmonika na koji je filtar ugođen što je vrijednost otpora R_f manja.[21]



Slika 5.4. *Prijenosna funkcija impedancije pojasno-propusnog filtra drugog reda[17]*

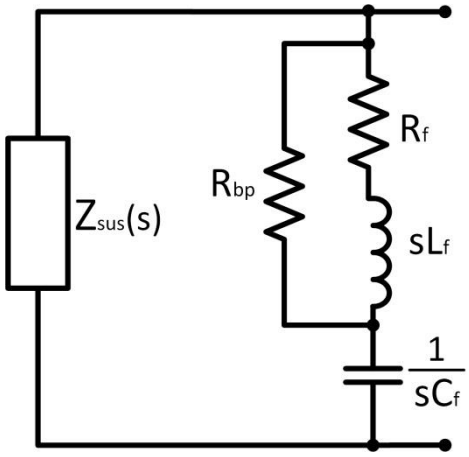
Slična razmatranja vrijede za strujnu prijenosnu funkciju filtra prikazanu na slici 5.5. Važno je uočiti da funkcija poprima minimalnu vrijednost H_{min} vrlo blizu rezonantne frekvencije što znači da pri toj frekvenciji najveći dio struje harmonika prolazi kroz filtar, a ne kroz sustav.[21]



Slika 5.5. *Strujna prijenosna funkcija pojasno-propusnog filtra drugog reda[17]*

5.1.2. Prigušeni pojasno-propusni filtar drugog reda

Najčešće korištena topologija prigušenog pojasno-propusnog filtra drugog reda prikazana je na slici 5.6.



Slika 5.6. *Shema prigušenog pojasno-propusnog filtra drugog reda*

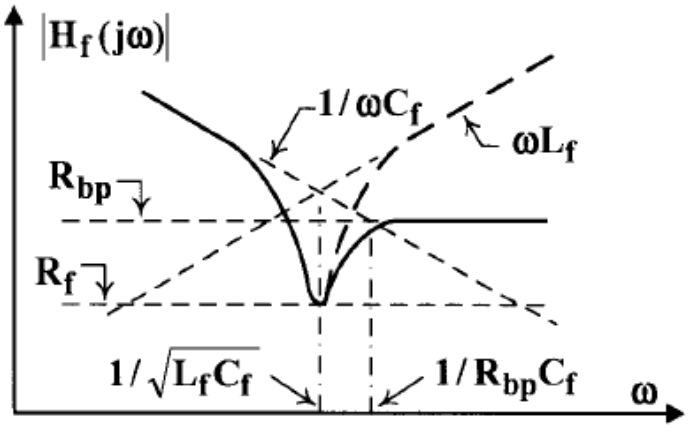
Kratkospojni otpornik R_{bp} spojen paralelno zavojnici osigurava prigušenje harmoničkih komponenti duž šireg frekvencijskog područja. Prijenosna funkcija impedancije filtra prikazana je formulom 5.5.

$$H_f(s) = Z_f(s) = \frac{U_f(s)}{I_f(s)} = \frac{1}{\frac{1}{R_{bp}}+\frac{1}{R_f+sL_f}} + \frac{1}{sC_f} = \frac{A}{s\left(1+\frac{s}{\omega_p}\right)}\left[\left(\frac{s}{\omega_0}\right)^2 + \frac{s}{\omega_0}\frac{1}{Q_{bp}} + 1\right] \qquad (5\text{-}5)$$

U navedenoj formuli A predstavlja pojačanje, ω_0 serijsku rezonantnu frekvenciju, Q_{bp} faktor dobrote filtra i ω_p frekvenciju polova. Navedeni parametri se računaju prema izrazima navedenim u formulama 5.6.

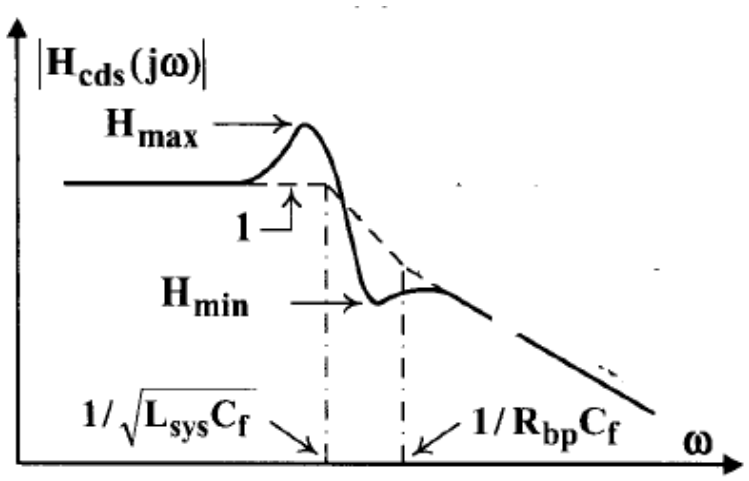
$$A = \frac{1}{C_f}, \omega_0 = \sqrt{\frac{R_f+R_{bp}}{R_{bp}L_fC_f}} \approx \frac{1}{\sqrt{LC}}, Q_{bp} = \frac{R_f+R_{bp}}{\omega_0(R_{bp}L_fC_f+L_f)}, \omega_p = \frac{R_f+R_{bp}}{L_f} \approx \frac{R_{bp}}{L_f} \qquad (5\text{-}6)$$

Vrijednost R_f otpora serijski vezanog zavojnici određuje se prema željenom faktoru kvalitete filtra, a vrijednost R_{bp} prema željenom odzivu, odnosno prigušenju harmonika na visokim frekvencijama.[21]



Slika 5.7. *Prijenosna funkcija impedancije prigušenog pojasno-propusnog filtra drugog reda[17]*

Prijenosna funkcija impedancije prikazana na slici 5.7. uvelike sliči onoj prethodno analiziranoj za neprigušeni pojasno-propusni filtar drugog reda. Glavna razlika koja se može uočiti jest prigušeni filtar koji nema veliku impedanciju na frekvencijama većim od rezonantne već mu je vrijednost na tim frekvencijama konstantna i direktno ovisi o otporu R_{bp}. Time se, za razliku od prethodne verzije neprigušenog filtra, dobiva mogućnost prigušenja harmonika na frekvencijama dosta većim od rezonantne.[21]



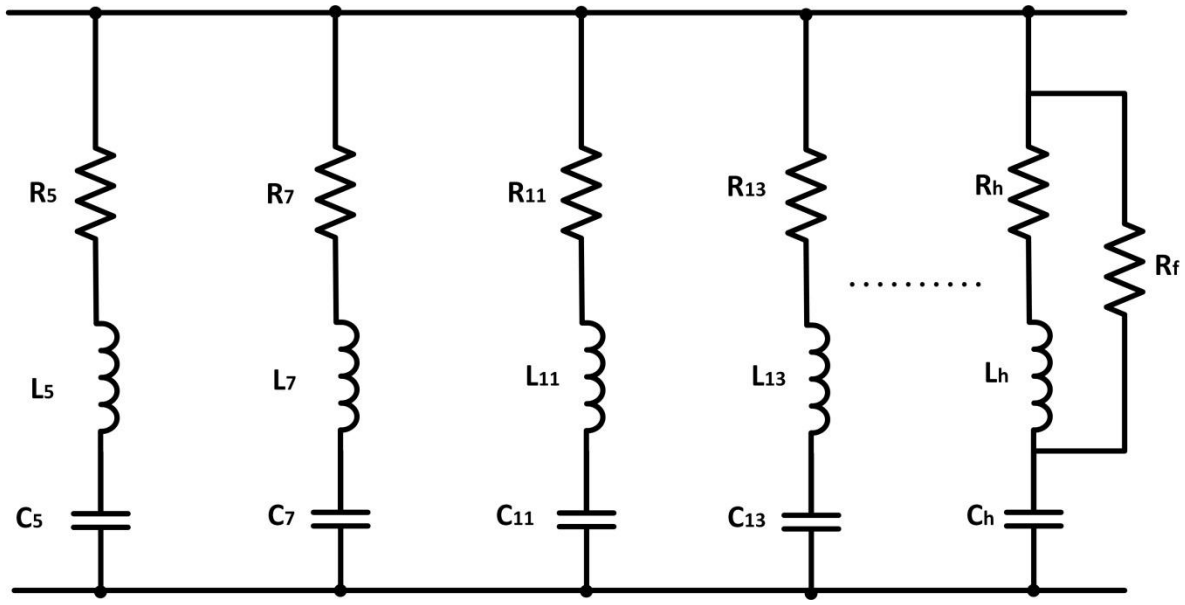
Slika 5.8. *Strujna prijenosna funkcija prigušenog pojasno-propusnog filtra drugog reda[17]*

Strujna prijenosna funkcija prikazana na slici 5.8. također dosta sliči onoj analiziranoj za neprigušeni filtar. Glavna razlika je u području frekvencija većih od rezonantne. Može se zaključiti kako karakteristika opada na višim frekvencijama što znači da određeni dio struje harmonika na tim frekvencijama prolazi kroz filtar i na taj način biva prigušen.

Dakle, glavna razlika,u određenim slučajevima i prednost, prigušenog pojasno-propusnog filtra drugog reda u odnosu na neprigušeni je u mogućnosti prigušenja harmonika na frekvencijama većim od nazivne.[21]

5.2. Složeni filtri

Filtri višeg reda dobiveni su povećavanjem broja spremnika energije koje čine kondenzatori i zavojnice. Međutim, to povećanje reda filtra ima određene posljedice i one se odnose na ekonomsku isplativost, odnosno povećanje cijene filtra te na smanjenje pouzdanosti samog filtra. Opći primjer složenog filtra dan je na slici 5.9.



Slika 5.9. *Primjer složenog filtra n-tog reda*

Može se uočiti kako se paralelno spaja nekoliko pojasno-propusnih filtara ugođenih na frekvencije pojedinog harmonika koji se nastoje prigušiti. Posljednja grana u filtru predstavlja visoko-propusni filtar koji služi prigušenju harmonika visokog reda koji najčešće nastaju uslijed brzih preklapanja učinskih sklopki.

Složeni filtri primjenjivi su samo u situacijama gdje su harmonici parnog reda zanemarivog iznosa. Razlog tomu je što dvije susjedne paralelne grane pojasno propusnih filtara ugođene na dva susjedna neparna harmonika čine rezonantni krug čija je frekvencija na parnom harmoniku između njih, što u slučaju veće amplitude tog parnog harmonika može izazvati ozbiljne probleme.

U simulacijskom modelu koristit ćemo složeni filtar kao na slici 5.9. Za suzbijanje viših harmonika koji su dominantniji u pojednim čvorovima koristit ćemo neprigušene pojasno-propusne filtre drugog reda spojene u paralelu i svaki od njih ćemo parametrirati na pojedini red dominantnog višeg harmonika, a za uklanjanje harmonika viših redova koji nisu toliko dominantni koristit ćemo prigušeni pojasno-propusni filtar drugoga reda. [21]

6. Evolucijski algoritam

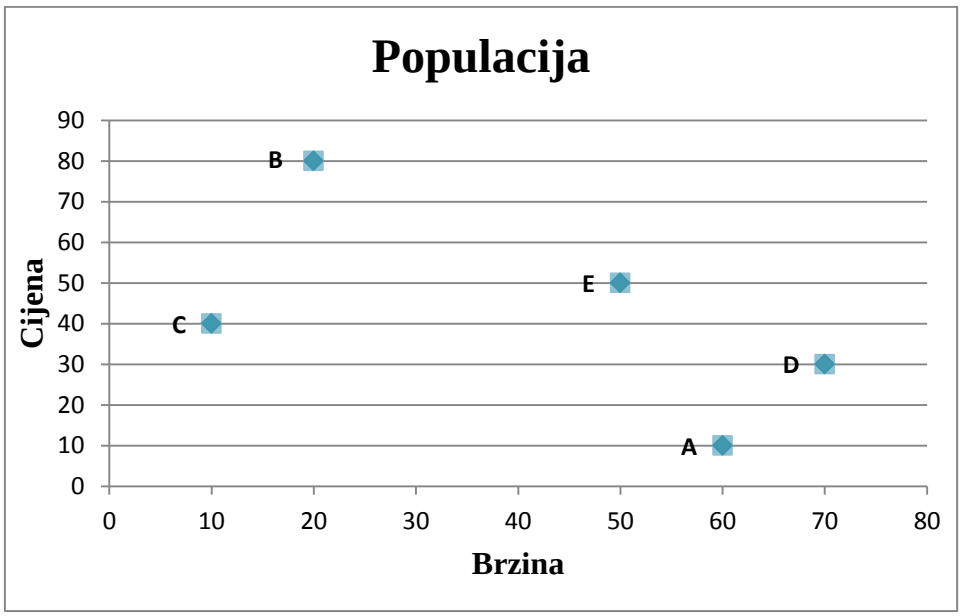
Evolucijski algoritmi su metaheuristički optimizacijski algoritmi koji su temeljeni na biološkim procesima evolucije kao što su mutacija, prirodna selekcija, križanje jedinki, opstanak najjačih i slično. Ovim algoritmom učinkovito se pretražuje područje problema pri čemu stohastičkim metodama dolazi do optimizacije. Jedinke možemo međusobno uspoređivati prema dobroti pri čemu biološkim procesima evolucije svake sljedeće generacije dobivamo jedinke koje su lošije ili bolje od prošle generacije pri čemu se genetski materijal pojedine jedinke prenosi na sljedeću generaciju i selektiraju se prema dobroti. Bolje jedinke su one koje se u konačnici bolje prilagode na okolinu nego njihove prethodne generacije te su im šanse za preživljavanje i stvaranje novih generacija veće, dok lošije jedinke stvaranjem novih generacija „izumiru“. Evolucijski algoritmi često se koriste za rješavanje optimizacijskih problema. Prema količini korištenih objekata razlikujemo dvije vrste optimizacije, a to su jednoobjektna i višeobjektna optimizacija. Jednoobjektna optimizacija uključuje samo jednu skupinu populacije iz koje križa jedinke, dok kod višeobjektne optimizacije postoji više skupina populacije gdje se križanjem i mutacijama traže najbolje jedinke iz više skupina jedinki. [18]

Tablica 6.1. *Jednoobjektna optimizacija[7]*

POPULACIJA	
	Brzina
Automobil 1	60
Automobil 2	20
Automobil 3	10
Automobil 4	70
Automobil 5	50

Tablica 6.2. *Višeobjektna optimizacija[7]*

	POPULACIJA	
	Brzina	Cijena
Automobil 1 (A)	60	10
Automobil 2 (B)	20	80
Automobil 3 (C)	10	40
Automobil 4 (D)	70	30
Automobil 5 (E)	50	50



Slika 6.1. *Omjer brzine i cijene automobila[7]*

Kao jednostavan primjer koristit ćemo automobil kod kojeg se za primjer jednoobjektog problema optimizacije predstavlja brzina automobila(Tablica 6.1.), a za višeobjektni problem su uzeti brzina i cijena automobila(Tablica 6.2.). U ovome primjeru uzeli smo mali uzorak automobila kako bi se lakše dočarala problematika optimizacije, u tablici 6.1 vidimo da je samo jedna vrsta funkcije cilja, a to je brzina pa ako želimo uzeti automobil sa što većom brzinom odabiremo onaj pod „D“ , no ako želimo da imamo što brži i cjenovno što jeftiniji automobil onda se tu javlja problem optimiranja najboljeg rješenja i to predstavlja višeobjektni problem optimizacije (Slika 6.1.) Kao rezultat ove optimizacije predstavljeno je rješenje „A“ jer se za puno manje novca može kupiti brži automobil. [7]

6.1. Podjela evolucijskih algoritama

Evolucijski algoritmi se zbog obuhvaćanja širokog područja primjene dijele na četiri vrste:

- genetski algoritmi
- genetsko programiranje
- evolucijsko programiranje
- evolucijska strategija

6.1.1. Genetski algoritam

Genetski algoritmi su prezentirani kao jedan od četiri tipa evolucijskih algoritama. Ideju koja se koristi za rješavanje problema optimiranja dobivena je na temelju evolucije u prirodi. Pri tome se koriste svi važni čimbenici procesa evolucije kao što su : populacija (jedinke), reprodukcijske metode, funkcija dobrote koja predstavlja sposobnost preživljavanja jedinke u prirodi, te selekcijski mehanizam ili opstanak jedinke. Kako bi usmjerili evoluciju u željenom smjeru te pronašli optimalno rješenje problema i u slučajevima kada ne postoji eksplicitno rješenje ono se ostvaruje uzimanjem u obzir samo željenih osobina jedinke. Primjena genetskih algoritama vidljiva u slučajevima kada se problem može opisati pomoću pretraživanja ili optimizacije proizvoljnih podataka, te ukoliko poznajemo način mogućeg mjerenja uspješnosti svakog pojedinog rješenja. Međutim, iako genetski algoritam pretražuje i optimizira zadane intervale podataka rješenje ne predstavlja uvijek optimum procesa,stoga je potrebno pažljivo i precizno opisati okruženje evolucije kako bi se zaista dobili rezultati bliski rješenju. Primjena genetskih algoritama najrasprostranjenija je na području traženja najkraćeg puta između dvaju ili više točaka, kod rješavanja problema neuronskih mreža, kod problema trgovačkog putnika i kod traženja optimuma nad bazom podataka. Naime, genetski algoritmi su ponovljivi. Ponovljivost(iterativnost) podrazumijeva izvođenje koje se sastoji od nekog broja iteracija, pri čemu se pri svakoj iteraciji dolazi sve bliže željenom rješenju. Postojeći skup jedinki iliti generacija promatra se pri svakoj iteraciji pri čemu se nad jedinkama se primjenjuju genetski operatori i metode selekcije s pomoću čega se stvara nova generacija iz koje se izvlače(selektiraju) najbolje jedinke. [5] [6]

6.1.1.1. Populacija

Populacija predstavlja skup jedinki iste vrste koje su smještene na nekom području. S vremenom jedan dio populacije stari i umire, a prije toga dolazi do razmnožavanja, odnosno stvaranja novih potomaka, ali veličina populacije u svakoj generaciji ostaje konstantna. Kod genetskog algoritma svaka jedinka unutar skupa populacije je potencijalno rješenje postavljenog problema. Glede odabira početne populacije, on se vrši slučajnim odabirom ili nekim drugim optimizacijskim postupkom. Uporabom funkcije dobrote provodi se opstanak jedinki koje su se uspjele prilagoditi novim životnim uvjetima u genetskim algoritmima, ali i onih slabijih jedinki koje to nisu uspjele. Operacijom križanja provodi se razmnožavanje jedinki koje su uz pomoć svojih dobrih svojstava opstale. Sa svakom sljedećom generacijom jedinke populacije postaju sve prilagođenije novim uvjetima, tj. njihova dobrota se povećava i približava rješenju problema Najčešće algoritam završava kad se dosegne zadani broj iteracija ili kada su određeni uvjeti traženog rješenja dosegnuli prihvatljivu razinu. Pri ispunjenu uvjeta bira se najbolja jedinka iz populacije te ona predstavlja rješenje optimizacijskog problema. [6]

6.1.1.2. Dobrota

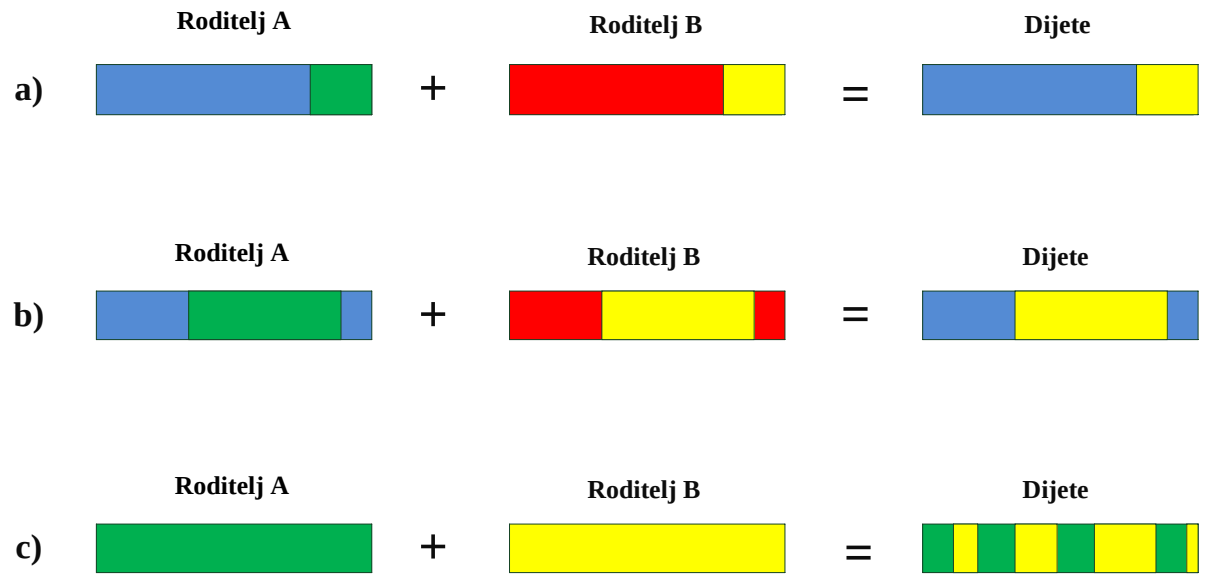
Funkcija dobrote je predstavljena kao prirodno djelovanje okoline na populaciju jedinki nad kojima se obavlja selekcija. Njihov odnos je proporcionalan, dakle, ako je jedinka bolje prilagođena okolini u kojoj živi, odnosno ukoliko joj je veća dobrota, tada su joj i šanse za preživljavanjem veće. Važno je istaknuti kako se jezgra problema ne nalazi u prvoj iteraciji pri odabiru jedinki s najvećom dobrotom iz razloga što jedinke koje su prosječne dobrote nose neka bitna svojstva koja ne bi trebalo izgubiti. Zaključimo, one jedinke koje imaju ekstremno nisku razinu dobrote unutar promatrane populacije bit će izbačene. Kod implementacije genetskog algoritma ključan problem jest odabir adekvatne funkcije dobrote. Funkcija dobrote se najčešće koristi u algoritmu, stoga bi trebala biti što jednostavnija i brža.

6.1.1.3. Križanje

Postupak križanja predstavlja združenost dviju jedinki(roditelja), pri čemu dolazi do zamjene jednog dijela roditeljskog gena onime drugoga te nastaju nove jedinke(djeca). Novonastala jedinka (dijete) sadrži kombinaciju roditeljskih osobina te njihov genom(genetski kod). Postupkom križanja jedinki koje se sastoje od pozitivnih i negativnih osobina očekujemo dijete koje će sadržavati pozitivne osobine, pri čemu je moguće zamjena potencijalnih

negativnih osobina nekog od roditelja genima drugog roditelja čije su osobine bolje na nepovoljnom dijelu.

Glavni cilj je naravno dobiti djecu bolju od svojih roditelja. Jedan od načina prikaza križanja je proizvoljnim brojem prekidnih točaka. Neki od primjera su na sljedećoj slici: pod 6.2. a) nalazi se križanje sa samo jednom točkom prekida, na slici 6.2. b) prikazano je križanje s dvije točke prekida. Na slici 6.2. c) odabrali smo više slučajnih točaka prekida. [5] [6][8]



Slika 6.2. *a) Križanje s jednom prekidnom točkom; b) Križanje s dvije točke prekida; c) Križanje kod kojeg se uzimaju slučajno odabrani dijelovi oba roditelja za zadani broj prekida[8]*

6.1.1.4. Mutacija

Mutacija jest proces slučajne trajne promjene genetskog materijala, pri čemu su najčešći uzrok vanjski čimbenici. Iz razloga što se procesom prirodnog odabira u populaciji nakupljaju mutacije koje pridaju bolju prilagodbu uvjetima okoliša te na taj način poboljšavaju vjerojatnost preživljavanja jedinki koje ih nose i prijenos na sljedeće generacije smatraju se jednim od preduvjeta evolucije. Proces mutacije odvija se nad pojedinom jedinkom, u pravilu nakon postupka križanja svako dijete prolazi navedeni proces mutacije s proizvoljnim parametrom vjerojatnosti. Zahvaljujući mutaciji postoji mogućnost da jedinka poprimi osobnosti koje niti jedan njen prethodnik nije imao.

6.1.1.5. Selekcija jedinki

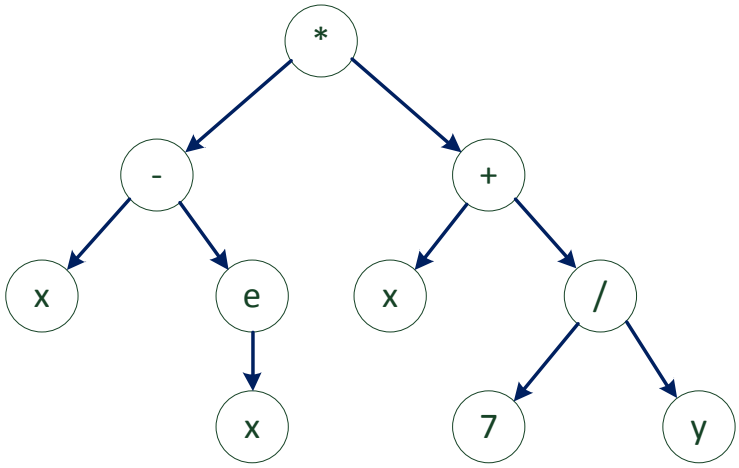
Selekcija je poredak generacije jedinki prema dobroti. Iz razloga što nije moguće očuvati sve jedinke u populaciji treba odrediti način odabira onih koje će opstati, ali i one koje će se ukloniti, odnosno uginuti. Premda je cilj sačuvanje jedinki samo pozitivnih svojstava, prerana konvergencija koja je posljedica selekcije samo najboljih jedinki, dovodi do negativno uspješnog algoritma. Navedeni problem nastoji se riješiti različitim vrstama selekcija. Kako bi proizveli sljedeću generaciju jedinki trebamo pronaći jedinke adekvatne za razmnožavanje. Neke od popularnijih tehnika selekcije su jednostavna, turnirska, eliminacijska i elitizam

1. Jednostavan odabir (eng. *Proportionate reproduction*) – kod ovog načina selekcije se jedinke biraju prema njihovoj dobroti
2. Turnirski odabir (eng. *Tournament selection*) – ovdje se bira slučajan broj jedinki iz generacije, najbolje jedinke postaju roditelji sljedeće generacije.
3. Eliminacijska selekcija (eng. *steady-state selection*) – u ovome slučaju se ne biraju dobre već i loše jedinke. Izborom najboljih jedinki novih generacija se pomoću genetskih operatora stvaraj nove jedinke, te se vrši izbor najlošijih jedinki koje se eliminiraju. Preostale jedinke preživljavaju do sljedeće selekcije.
4. Elitizam – to je mehanizam kojim se najbolja jedinka u populaciji štiti od bilo kakvih izmjena u svrhu očuvanja najkvalitetnijeg materijala za sljedeću generaciju.

6.1.2. Genetsko programiranje

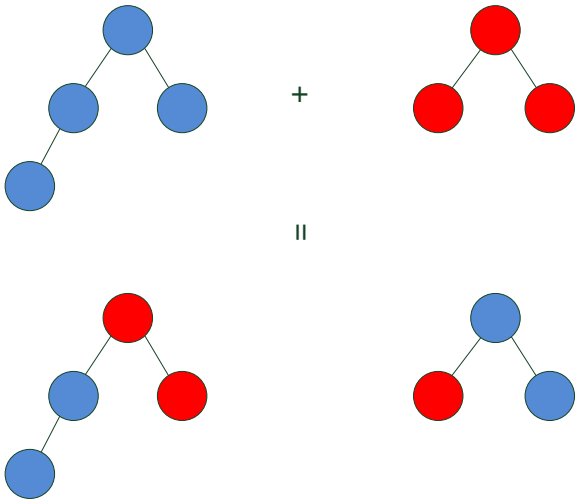
Genetsko programiranje je vrsta evolucijskog algoritma koji je temeljen na biološkoj evoluciji. Primarna ideja je da računalo riješi nekakav problem pri zadanim uvjetima i granicama. Velik je broj kompleksnih problema kod kojih je genetskim programiranjem postignut napredak u odnosu na do tada najbolja poznata rješenja, neki od tih problema vežu se uz područja električnih krugova, molekularne biologije, raspoređivanja itd. Primarna ideja je kreirati populaciju početnih rješenja (programa za rješavanje problema); zatim, kroz određeni broj generacija te s djelovanjem genetskih operatora pronaći optimalno rješenje koje zadovoljava zadane uvjete.Jedinke koje čine generaciju su programi koji predstavljaju potencijalna rješenja problema, a nisu više podatci s konačnim veličinama. Genetsko programiranje zbog jednostavnosti razvija računalne programe koji su u memoriji prikazani pomoću stabala (slika 6.3.). Stabla su preglednija jer se ona mogu rekurzivno vrlo

jednostavno čitati. Svaki čvor stabla sadrži neki matematički operator, dok listovi sadrže operande. Matematički operatori mogu biti aritmetički (+, -, *, /, *sin, cos, tg, ctg, exp*), logički (i, ili, ne, ni, nili) ili specifični ovisno o problemu (min, max).



Slika 6.3. *Prikaz programske jednadžbe „*$(x\text{-}e^{x}) * (x + \frac{7}{y})$*“ pomoću stabla [8]*

Kako kod genetskog algoritma koristimo genetske operatore tako i kod genetskog programiranja isto trebamo koristiti selekcija, križanje i mutaciju. Ova vrsta križanja(križanje stabala) sastoji se od procesa u kome u dva različita stabla odaberemo po jedno podstablo te im supstituiramo mjesta. Proces križanja stabala prikazan je na slici 6.4. Proces mutacije vrši se na način da nasumično odabranom stablu slučajno odabere neki čvor. Novo stablo koje nastaje iz tog čvora se eliminira i na njegovo mjesto dolazi novo slučajno stvoreno stablo i na kraju se vrši selekcija stabala.



Slika 6.4. *Križanje stabala[6]*

6.1.3. Evolucijsko programiranje

Kod evolucijskog programiranja ne postoji razmjena genetskog materijala među jedinkama, odnosno ne koristi se operator križanja nego samo mutacije,a to predstavlja najvažniju razliku između evolucijskog programiranja i ostalih evolucijskih algoritama. Međutim, ukoliko zanemarimo navedeno, evolucijsko programiranje je vrlo slično evolucijskim strategijama. Kako evolucijsko programiranje ne sadrži sve genetske operatore tako se i način selekcije mijenja. Način selekcije kod evolucijskog programiranja vrši se tako da se svaka od „*X*“ jedinki iz pojedine generacije mutira pa dobijemo „*2X*“ jedinki koje se selektiraju prema dobroti i nakon toga bolja polovica populacije se koristi za stvaranje nove generacije dok ostatak izumire. Mutacija mijenja vrijednosti pojedinih gena baš kao i kod ostalih evolucijskih algoritama. Za razliku od ostalih evolucijskih algoritama, kod evolucijskog programiranja intenzitet mutacije predstavlja „snagu“, tj. broj gena pojedine jedinke koje se mutiraju. Najčešća primjena evolucijskog programiranja je pri traženju maksimuma i minimuma funkcija realnih varijabli. Jačina mutacije za ovaj algoritam odvija se prema Gaussovoj jediničnoj normalnoj razdiobi, što znači da algoritam uzima više manjih promjena u odnosu na nekoliko velikih. Procesi evolucijske strategije i evolucijskog programiranja mogu biti gotovo isti, kao i problemi koji se njima rješavaju. Dok evolucijsko programiranje novu generaciju stvara samo mutacijom, kod evolucijske strategije postoji izbor hoće li se koristiti križanje ili ne. [4][8]

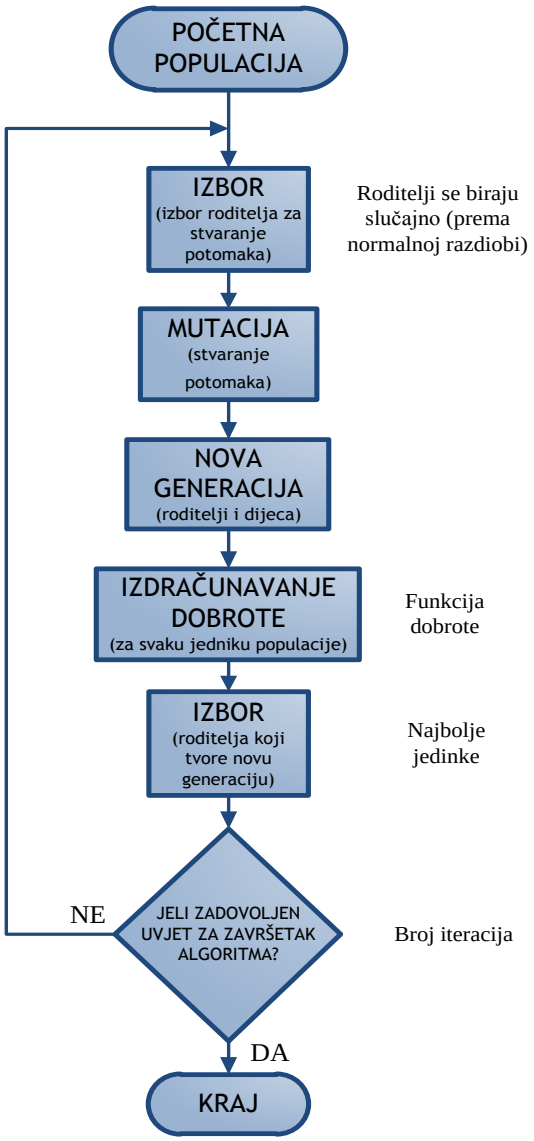
6.1.4. Evolucijske strategije

Evolucijske strategije predstavljaju jednu od tehnika optimizacije iz područja evolucijskih algoritama. Njihova prvenstvena primjena sačinjavala se u rješavanju problema najboljeg aerodinamičkog dizajna, ali danas se primjenjuju pri određivanju parametara na osnovu kojih međudjeluju pojedine galaksije u svemiru. Potencijalna rješenja evolucijske strategije predstavljaju se kao konstantno veliki vektori realnih brojeva, pri čemu broj na određenom mjestu unutar vektora opisuje neku od karakteristika rješenja. Kao i kod svih ostalih evolucijskih metoda iz generacije u generaciju pokušava se pronaći što bolje rješenje korištenjem operatora selekcije i mutacije. Za razliku od genetskih algoritama kod kojih je većinom populacija oko tisuću i više jedinki, kod evolucijske strategije je populacija najčešće jednaka broju dva, od kojih je jedna trenutni roditelj, a druga nova jedinka, koja je rezultat mutacije nad roditeljem. U današnje vrijeme evolucijske strategije koriste više roditelja i

operator križanja kako bi se izbjeglo zaglavljivanje rješenja u području nekog lokalnog optimuma. Mutacija kod evolucijskih strategija je tipična Gaussova mutacija koja je opisana kod Evolucijskog programiranja.

Postupak stvaranja novih generacija opisan koracima i dijagramom toka(Slika 6.5.):

1. Stvaranje populacije
2. Odabir slučajnih roditelja
3. Vršenje mutacije nad genima roditelja te stvaranje „mutanta“
4. Odaberi novu populaciju na temelju funkcije dobrote(selekcija)
5. Ako je zadovoljen rezultat, tada završi algoritam, ako nije vrati se na korak 3.



Slika 6.5. *Dijagram toka evolucijske strategije[8]*

7. OpenDSS simulacijski program

OpenDSS (eng. *The Open Distribution System Simulator*) je opsežan softverski alat za simulaciju distribucije električne energije visokonaponskim sustavima u gotovo svim frekvencijskim domenama. Također, omogućuje vršenje puno analiza koje zadovoljavaju potrebe suvremenih tehnologija. Primjer toga su pametne mreže(eng. Smart Grid), modernizacija energetskih mreža te razna istraživanja u polju obnovljivih izvora energije.

OpenDSS se koristi još od 1997. godine u raznim istraživanjima i projektima koji zahtijevaju analizu energetskih mreža. Većina mogućnosti koje program nudi prvotno su bile namijenjene analizi energetskih distributivnih mreže te je tako ostalo i do danas. Od ostalih mogućnosti uglavnom se koristi za analizu energetske efikasnosti u pri prijenosu električne energije i harmonika koji se pojavljuju. OpenDSS je zamišljen i dizajniran na način da se lako može nadograđivati pa je samim time kvalitetna podloga za srodne analize i izračune koji će se koristiti u budućnosti. Dodatna kvalitetna karakteristika ovog programa je činjenica da se radi o otvorenom softveru, odnosno da se program može besplatno mijenjati ili nadograđivati.

Neki od slučajeva u kojima se koristi OpenDSS su:

- Planiranje i distribucija električne energije
- Općenita analiza višefaznih krugova električne energije
- Studije bazirane na procjeni rizika pri distribuciji električne energije
- Simulacija energetskih postrojenja sa solarnim panelima
- Simulacija vjetroelektrana
- Analiza sustava zaštite distributivnih mreža
- Modeliranje spremišta električne energije
- Analiza harmoničkih izobličenja mrežnih veličina
- Analiza impulsnih opterećenja mreže

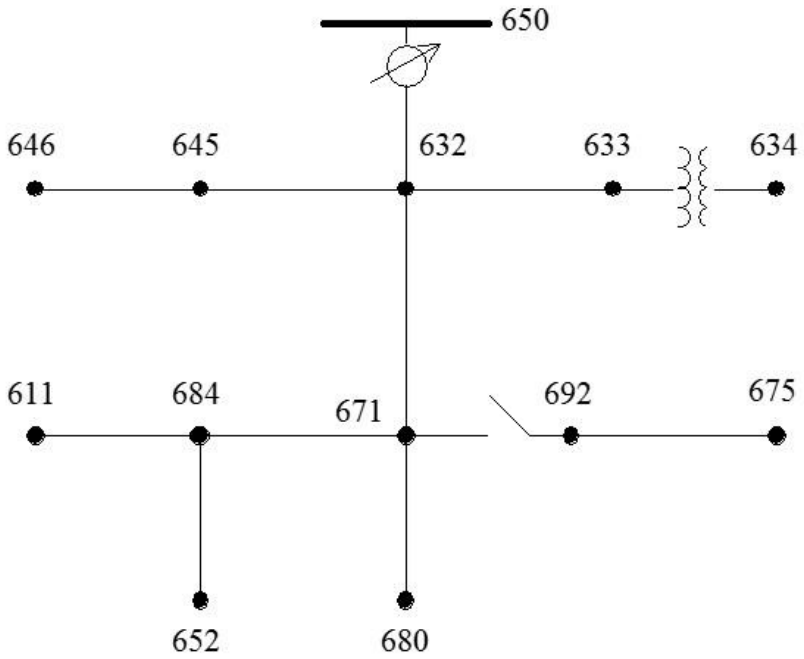
Sukladno namjeri da se postojeći softver može dodatno nadograđivati unutar OpenDSS-a je implementirano COM(eng. *Component Object Model*) sučelje koje se koristi kako bi napredni korisnici mogli iskoristiti mogućnosti ovog programa pri kreiranju novih tipova analiza i studija. Kroz COM sučelje korisniku je omogućeno kreiranje i izvršavanje raznih modela te kontroliranje parametara i funkcija simulatora. Neke od vanjskih aplikacija koji se mogu koristiti za komunikaciju s OpenDSS-om preko COM sučelja su Mathworks MATLAB,

Python, C#. Navedeno omogućuje širok spektar mogućih analiza na modelima unutar OpenDSS-a, kao i izvrsne mogućnosti grafičkog prikaza rezultata.

Unutar ovog rada OpenDSS-om će se upravljati pomoću Python aplikacije i detalji će biti objašnjeni u nastavku rada. [9][10]

7.1. Primjena OpenDSS-a za simulaciju mreže

Za simulaciju nesimetrične visokonaponske mreže uzeli smo primjer mreže sa trinaest čvorova iz priloga koji se dobije pri instalaciji OpenDSS paketa. IEEE mreža s 13 čvorova (Slika 7.1.) je, gledajući na stvarne realne mreže, vrlo mala ali detaljno opisana mreža koja je na baznom naponu od 4,16 kV(650) i ona služi za testiranje i analizu distribucijskog sustava. Mreža se sastoji od regulacijskog transformatora u trafostanici, nadzemnih i podzemnih vodova, dva visokonaponska kondenzatora i 9 nesimetričnih trošila.



Slika 7.1. *Nesimetrična mreža sa 13 čvorova[11]*

Modeliranje u OpenDSS-u vrši se pisanjem naredbi, pa ćemo kao primjer navesti prvo transformator, tj. trofazni transformator u trafostanici ove mreže. Kao što možemo vidjeti naredba „New Transformer.Sub Phases=3 Windins=2 XHL=(8 1000 /)“ predstavlja stvaranje

novog trofaznog transformatora pod nazivom „Sub“ koji sadrži dva namotaja(primarni i sekundarni) pri čemu naredba „XHL“ predstavlja postotak reaktancije(X_{12}) po jedinici snage od 1000kVA. Naredbom „~ wdg=1 bus=SourceBus conn=delta kv=115 kva=5000 %r=(.5 1000 /) “ se opisuje o kojemu se namotaju radi(wdg=1,peimar), „bus=SourceBus“ predstavlja sabirnicu na koju je spojen primarni namotaj transformatora, „conn=delta“ prezentira grupu spoja namotaja(primar je spojen u trokut), „kv=115“ naredba predstavlja naponsku razinu ovog namotaja, „kva=5000“ predstavlja prividnu snagu ovoga namotaja i „%r=(.5 1000 /)“ ova naredba predstavlja otpor namotaja u postotcima po jedinici snage od 1000kVA.

Kod 1:

```
! TRANSFORMER DEFINITION
~ wdg=1 bus=SourceBus   conn=delta  kv=115  kva=5000   %r=(.5 1000 /)
~ wdg=2 bus=650         conn=wye    kv=4.16  kva=5000   %r=(.5 1000 /)
```

Nakon što je opisano modeliranje transformatora sada će se opisat modeliranje trošila. Naredbom „New Load.671 Bus1=671.1.2.3 Phases=3 Conn=Delta Model=1 kV=4.16 kW=1155 kvar=660 “ definiramo novo trošilo „671“ koje ujedno predstavlja čvor u mreži te se nalazi na sabirnici „671.1.2.3“, gdje su sve tri faze spojene na trošilo u trokut spoju(„con=Delta“), trošilo je na naponskoj razini od 4,16 kV gdje je radna snaga trošila je 1155 kW , a jalova snaga 660 kVAr.

Kod 2:

```
!LOAD DEFINITIONS
New Load.671 Bus1=671.1.2.3  Phases=3 Conn=Delta Model=1 kV=4.16   kW=1155 kvar=660
New Load.634a Bus1=634.1     Phases=1 Conn=Wye  Model=1 kV=0.277  kW=160   kvar=110
New Load.634b Bus1=634.2     Phases=1 Conn=Wye  Model=1 kV=0.277  kW=120   kvar=90
New Load.634c Bus1=634.3     Phases=1 Conn=Wye  Model=1 kV=0.277  kW=120   kvar=90
New Load.645 Bus1=645.2      Phases=1 Conn=Wye  Model=1 kV=2.4    kW=170   kvar=125
New Load.646 Bus1=646.2.3    Phases=1 Conn=Delta Model=2 kV=4.16   kW=230   kvar=132
New Load.692 Bus1=692.3.1    Phases=1 Conn=Delta Model=5 kV=4.16   kW=170   kvar=151
New Load.675a Bus1=675.1     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=485   kvar=190
New Load.675b Bus1=675.2     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=68    kvar=60
New Load.675c Bus1=675.3     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=290   kvar=212
New Load.611 Bus1=611.3      Phases=1 Conn=Wye  Model=5 kV=2.4  kW=170   kvar=80
New Load.652 Bus1=652.1      Phases=1 Conn=Wye  Model=2 kV=2.4  kW=128   kvar=86
New Load.670a Bus1=670.1     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=17    kvar=10
New Load.670b Bus1=670.2     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=66    kvar=38
New Load.670c Bus1=670.3     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=117   kvar=68
```

Definiranje vodova u OpenDSS-u se vrši pomoću naredbe „New Line.650632 Phases=3 Bus1=RG60.1.2.3 Bus2=632.1.2.3 LineCode=mtx601 Length=2000 units=ft “ koja predstavlja tri voda pod nazivom „650632“, a oni povezuje čvorove(trošila) „650“ i „632“, na sabirničkim mjestima „RG60.1.2.3 “ i „632.1.2.3“. Naredba „LineCode=mtx601“ predstavlja

matricu koja opisuje svojstva voda, „Length=2000“ predstavlja dužinu voda i „units=ft“ opisuje mjernu jedinicu u kojoj je prikazana duljina voda(u ovom slučaju američka stopa).

Kod 3:

```
!LINE DEFINITIONS
New Line.650632   Phases=3 Bus1=RG60.1.2.3   Bus2=632.1.2.3  LineCode=mtx601 Length=2000 units=ft
New Line.632670   Phases=3 Bus1=632.1.2.3    Bus2=670.1.2.3  LineCode=mtx601 Length=667  units=ft
New Line.670671   Phases=3 Bus1=670.1.2.3    Bus2=671.1.2.3  LineCode=mtx601 Length=1333 units=ft
New Line.671680   Phases=3 Bus1=671.1.2.3    Bus2=680.1.2.3  LineCode=mtx601 Length=1000 units=ft
New Line.632633   Phases=3 Bus1=632.1.2.3    Bus2=633.1.2.3  LineCode=mtx602 Length=500  units=ft
New Line.632645   Phases=2 Bus1=632.3.2      Bus2=645.3.2    LineCode=mtx603 Length=500  units=ft
New Line.645646   Phases=2 Bus1=645.3.2      Bus2=646.3.2    LineCode=mtx603 Length=300  units=ft
New Line.692675   Phases=3 Bus1=692.1.2.3    Bus2=675.1.2.3  LineCode=mtx606 Length=500  units=ft
New Line.671684   Phases=2 Bus1=671.1.3      Bus2=684.1.3    LineCode=mtx604 Length=300  units=ft
New Line.684611   Phases=1 Bus1=684.3        Bus2=611.3      LineCode=mtx605 Length=300  units=ft
New Line.684652   Phases=1 Bus1=684.1        Bus2=652.1      LineCode=mtx607 Length=800  units=ft
```

Modeliranje kondenzatora se vrši pomoću naredbe „New Capacitor.Cap1 Bus1=675 phases=3 kVAR=600 kV=4.16 “ gdje je definiran kondenzator pod imenom „Cap1“ koji se nalazi na sabirnici 675, koji je spojen na sve 3 faze na naponskoj razini od 4,16 kV i jalove snage 600 kVAr.

Kod 4:

```
!CAPACITOR DEFINITIONS
New Capacitor.Cap1 Bus1=675 phases=3 kVAR=600 kV=4.16
New Capacitor.Cap2 Bus1=611.3 phases=1 kVAR=100 kV=2.4
```

Kako bismo izmijenili svojstva nesimetrične mreže, odnosno prilagodili ih normi EN 50160 potrebno je u mreži kreirati filtar. Filtar smo kreirali pomoću naredbe „New Capacitor.FILTER1 Bus1=671.1.2.3 phases=3 kVAR=600 kV=4.16 xl=2.5 R= 0.187 Harm=7 Numsteps=2 conn=wye“ gdje je filtar predstavljen kao kondenzator naziva „FILTER1“ , ali s više svojstava. Dodatna svojstva ovog kondenzatora koja ga čine filtrom su reaktanncija „xl=2.5“, otpor „R= 0.187“ i red harmonika za koji je namijenjen „Harm=7“. „Numsteps=2“ predstavlja broj koraka uključivanja filtra u slučaju kada se koristi nekoliko sekcija kondezatora, a conn=wye“ predstavlja vrstu spoja filtra(ovaj je spojen u zvijezdu) zato što se radi o trofaznom sustavu.

Kod 5:

```
! DEFIE FILTERS
New Capacitor.FILTER1 Bus1=671.1.2.3 phases=3 kVAR=600 kV=4.16 xl=2.5 R= 0.187 Harm=7 Numsteps=2 conn=wye
```

Stvaranje mjerne točke vrši se pomoću naredbe „New monitor.M1 Line.671680“, a ova mjerna točka nazvana je „M1“, a smještena je na vodu između čvorova 671 i 680 i ona mjerene podatke sprema u dokument CSV(Comma Separated Value) formata. Naredbama „Solve“ i „Solve Mode=harmonics“ u OpenDSS-u uključuje se poseban režim rada gdje se na svakom trošilu mjere frekvencije i njihovi višekratnici te amplitude napona i struja koji su na višim frekvencijama od osnovne. I zadnja naredba „Show mon M1“ otvara spremljeni CSV dokument u kojemu su prikazani mjereni podatci u točki mreže „M1“.

Kod 6:

```
New monitor.M1 Line.671680
Solve
Solve Mode=harmonics
Show mon M1
```

Harmonička izobličenja koja stvaraju trošila dana su naredbom „New "Spectrum.defaultload" NumHarm=7 harmonic=(1, 3, 5, 7, 9, 11, 13,) %mag=(100, 1.5, 20, 14, 1, 9, 7,) angle=(0, 180, 180, 180, 180, 180, 180,)“ gdje „NumHarm=7“ predstavlja koliko će se harmonika generirati, „harmonic=(1, 3, 5, 7, 9, 11, 13,)“ koji su to harmonici, „%mag=(100, 1.5, 20, 14, 1, 9, 7,)“ kolika je amplituda pojedinog harmonika i na kraju „angle=(0, 180, 180, 180, 180, 180, 180,)“ kut djelovanja harmonika. Ovo su izvorno zadani harmonici i nisu naknadno mijenjani, a njihove su veličine dane su sljedećom tablicom.[16]

Tablica 7.1. *Harmonička izobličenja kod trošila[16]*

Red harmonika	Amplituda harmonika (%)	Kut harmonika (°)
1	100	0
3	1,5	180
5	20	180
7	14	180
9	1	180
11	9	180
13	7	180

8. Opis optimizacijskog problema

Glavna zadaća ovoga rada je riješiti problem smještaja pojasno-propusnog filtra u nesimetričnu mrežu kako bi se ispunili uvjeti razine THD-a prema normi EN 50160. Optimizacijski problem proizašao je iz potrebe usklađivanja ekonomičnosti i cijena za smanjenje THD-a u distribucijskim mrežama. Kako većina trošila mrežu „prlja“ s jednim ili više neparnih harmonika tada i *THD* prelazi dozvoljeni prag. Da bi se održala kvaliteta električne energije javlja se potreba za suzbijanjem tih harmonika, ali rješenje da se u svaki čvor stavlja filtar nije ekonomično niti cjenovno isplativo pa se zbog toga pomoću genetskog algoritma traži jedan čvor u mreži i parametri filtra(otpor, reaktancija) uz konstantnu snagu filtra. Matematički opis problema predstavit ćemo kao jednoobjektni problem optimizacije. Kao rješenje genetskog algoritma traži se da THD_U mreže bude manji od 8% tako da kombinacijom parametara filtra(varijable odluke): reda harmonika(h) , reaktancije(X_L), otpora(R) i čvora u mreži(N) koji se kreira u nesimetričnoj mreži ispune se zahtjevi EN 50160 norme. Varijable odluke optimizacijskog problema su izlazne varijable optimizacije koje proces optimizacije na kraju da kao rezultat optimizacije. Kako bi zahtjevi norme u potpunosti bili ispunjeni, kao granice navode se i pojedini relativni naponi svakog harmonika. Ograničeni su i intervali reaktancije(X_L) i otpora(R) filtra da bi se ubrzalo rješavanje problema.

$$
\begin{aligned}
&JO(g)=[\,THD_U] \rightarrow min;\\
&g_1: h= 3, 5, 7, 9, 11, 13;\\
&g_2: THD_U < 8\%;\\
&g_3: U_3 \leq 0{,}05 * U_1;\\
&g_4: U_5 \leq 0{,}06 * U_1;\\
&g_5: U_7 \leq 0{,}05 * U_1\ ; \qquad (8\text{-}1)\\
&g_6: U_9 \leq 0{,}015 * U_1;\\
&g_7: U_{11} \leq 0{,}035 * U_1;\\
&g_8: U_{13} \leq 0{,}03 * U_1;\\
&g_9: \ 0 \leq X_L \geq 100000;\\
&g_{10}: \ 0 \leq R \geq 1000;\\
&g_{11}: \ N = 611, 632, 633\ 634, 645, 646, 652, 671, 675, 680, 684, 692, RG60;
\end{aligned}
$$

8.1. Norma EN 50160

Kako bi se sačuvala kvaliteta električne energije u nesimetričnoj mreži potrebno je pridržavati se norme namijenjene za održanje kvalitete električne energije u elektrodistributivnom sustavu. Svaka realna mreža sadrži različite aktivne elemente koji utječu na kvalitetu električne energije(pojava viših harmonika). Norma EN 50160 definirana je tako da maksimalni THD_U bilo koje faze(L1, L2, L3) mreže ne smije biti veći od 8% te u nastavku možemo vidjeti tablicu harmoničkih ograničenja.[12]

Tablica 8.1. *Harmonička ograničenja prema EN 50160*

Red harmonika	Postotak baznog napona (%)
3	5
5	6
7	5
9	1,5
11	3,5
13	3

8.2. Primjena Pythona za rješavanje problema optimizacije

Python je prvenstveno objektno orijentiran, interpreterski programski jezik vrlo visoke razine razvijen 1990. godine . Do 1998. godine Python je već stekao blizu 300 tisuća korisnika, a nakon toga su ga uveliko prihvatili tržišni giganti kao što su: MIT, NASA, IBM, Google, Yahoo i drugi. Python nije inovacijski program, ali je inovativno povezao najbolje ideje i načine rada drugih programa. Također je nadalje dobro optimiziram pa je zbog toga jednostavan za shvaćanje, a ujedno i moćan alat. Kao takav, on dopušta programerima korištenje više stilova programiranja kao što su: objektno orijentirano, strukturno i aspektno orijentirano programiranje. Još jedna važna prednost Pythona je da je on besplatan za akademske ustanove i neprofitnu upotrebu gdje je također dobro pokriven literaturom i dokumentacijom. Kako bi olakšali rad s Pythonom koristi se razvojno sučelje PyCharm koje

je namijenjeno za pristupačnije programiranje u Pythonu, a samim time pruža dodatne mogućnosti, primjerice kontrola pisanja koda(predlaže i dopunjuje kod te podcrtava pogreške), jednostavno kretanje i preglednost koda, dobro usklađena komunikacija s web framework-cima(Django, Jupyter), sadrži Python Debugger pomoću kojeg se lakše otkrivaju pogreške u kodu itd. Pycharm je kompatibilan sa više platformi kao što su: Windows, Linux, MacOS, a da bi se koristio mora biti namijenjen za verziju Pythona koja je instalirana. [13][14]

8.2.1. Opisivane izvornog koda iz PyCharma

Iz razloga da bismo što lakše i bolje mogli programirati u razvojnom sučelju PyCharm uz instaliran Python koristili smo i dodatne pakete koji se instaliraju kao dodatci. Konkretno u ovom primjeru su korišteni slijedeći paketi:

- matplotlib – koristi se za plotanje grafova,
- numpy – koristi se za matematičko računanje
- pywin32 – koristi se za ostvarivanje komunikacije s OpenDss-om preko COM sučelja
- scipy – koristi se za znanstvene eksperimente, a u ovome slučaju za korištenje genetskog algoritma

Da bi kod programa bio pregledniji načinjene su dvije Python skripte. Prva skripta je nazvana „GeneticAlg“, kod nje je uglavnom opisan genetski algoritam i njegove granice, a druga Python skripta se naziva „CreateNetworkInOpenDSS“ i u njoj je opisana komunikacija Pythona i OpenDSS-a te naredbe za prikazivanje i opisivanje rješenja problema, stoga je prvo opisan izvorni kod iz „GeneticAlg“ skripte. Kako možemo vidjeti u Kodu 7. prvo su uveženi(importani) podatci iz skripte „CreateNetworkInOpenDSS“, podatci iz interpretera „numpy“ koji su definirani kao „np“ i uvežena je diferencijalna evolucija(genetski algoritam) iz interpretera „scipy“

Kod 7:

```
import CreateNetworkInOpenDSS
import numpy as np
from scipy.optimize import differential_evolution
```

U kodu 8. definirana je glavna funkcija „main()“ te granice genetskog algoritma(„bounds“). Varijablom „result“ predstavljeno je rješenje funkcije „differential_evolution“ koja je definirana raznim parametrima kao što su broj iteracija, strategija rješenja, koeficijent mutacija, način selekcije itd. Funkcija „scipy.optimize.differential_evolution (func, bounds, args=(), strategy='best1bin', maxiter=None, popsize=15, tol=0.01, mutatio n=(0.5, 1), recombination=0.7, seed=None, callback=None, disp=False, polish=True, init ='latinhypercube') „ generalno služi kako bi pronašla minimum funkcije s više varijabli. Funkcija u svojem radu koristi principe genetičkog algoritam koji je već opisan u sklopu ovog rada. Neki od važnijih parametara sljedeće funkcije koji se koriste u ovom radu su:

- func – predstavlja opis matematičkog problema za koji je potrebno pronaći minimum. Unutar koda ova funkcija predstavlja metodu koja prima parametre potrebne za simulaciju mreže u OpenDSS-u a vraća vrijednost koja se treba minimizirati.
- bounds – predstavlja granice parametara simulacije koji se koriste unutar generičkog algoritma. Unutar koda ovaj parametar predstavlja polje parova koji predstavljaju minimalnu i maksimalnu vrijednost pojedinog parametra.
- strategy – predstavlja strategiju koju algoritam koristi kako bi pronašao minimum date funkcije. Pojedine strategije su više prikladne za pojedine vrste optimizacijskih problema u odnosu na druge. Unutar ovog rada se koristi općenita strategija best1bin i best1exp.
- maxiter – predstavlja parametar generičkog algoritma, odnosno maksimalni broj generacija potomaka unutar kojih je cijela populacija uključena.
- popsize – predstavlja faktor koji se koristi pri računanju veličine populacije generičkog algoritma.
- res – predstavlja rezultat optimizacije odnosno parametre optimizacije koji daju najbolji rezultat. Unutar koda ovaj parametar predstavlja polje parametara koji se mogu smatrati najboljim za dobivanje minimuma funkcije uključujući restrikcije definirane ostalim argumentima ove metode.[15]

Kod 8:

```
Def main():

    bounds = [(0.0, 15.0), (0.0, 100000.0), (0.0, 1000.0), (0.0, 7.0)]

    result = differential_evolution(problemFunction, bounds, (),
```

```
                    'best1bin', 100, 15, 0.01, (0.5, 1),
                    0.7, None, None, False, True,
                    'latinhypercube', 0)
    print(result.x)
    BrojIteracije = np.arange(0, (len(CreateNetworkInOpenDSS.THD1results)))
    print ('Broj iteracija je: ' + str(len(BrojIteracije)-1))
```

Iz razloga što redoslijed viših harmonika ne sadrži parne brojeve potrebno je definirati kako će genetski algoritam pri svakoj iteraciji uzeti neparni harmonik, to smo definirali u kodu 9.

Kod 9:

```
def getHarmFromNum( harmNumber ):
    harmOrder = ['1', '3', '5', '7', '9', '11', '13']
    if harmNumber < 1.0:
        return  harmOrder[0]
    elif harmNumber >= 1.0 and harmNumber < 2.0:
        return  harmOrder[1]
    elif harmNumber >= 2.0 and harmNumber < 3.0:
        return  harmOrder[2]
    elif harmNumber >= 3.0 and harmNumber < 4.0:
        return  harmOrder[3]
    elif harmNumber >= 4.0 and harmNumber < 5.0:
        return  harmOrder[4]
    elif harmNumber >= 5.0 and harmNumber < 6.0:
        return  harmOrder[5]
    elif harmNumber >= 6.0 and harmNumber < 7.0:
        return  harmOrder[6]
```

Kako je definiran odabir harmonika tako je bilo potrebno i definirati odabir pojedinog čvora mreže s obzirom na to da su čvorovi definirani i brojevima i slovima.(Kod 10.)

Kod 10:

```
def getBusFromNum( busNumber ):
    namesOfBuses = ['650.1.2.3', 'RG60.1.2.3', '646.1.2.3', '645.1.2.3',
'632.1.2.3', '633.1.2.3', '634.1.2.3',
                    '670.1.2.3', '611.1.2.3', '684.1.2.3', '671.1.2.3',
'692.1.2.3', '675.1.2.3', '652.1.2.3', '680.1.2.3']
    if busNumber < 1.0:
        return namesOfBuses[0]
    elif busNumber >= 1.0 and busNumber < 2.0:
            return  namesOfBuses[1]
    elif busNumber >= 2.0 and busNumber < 3.0:
            return  namesOfBuses[2]
    elif busNumber >= 3.0 and busNumber < 4.0:
            return  namesOfBuses[3]
    elif busNumber >= 4.0 and busNumber < 5.0:
            return  namesOfBuses[4]
    elif busNumber >= 5.0 and busNumber < 6.0:
            return  namesOfBuses[5]
    elif busNumber >= 6.0 and busNumber < 7.0:
            return  namesOfBuses[6]
```

```
    elif busNumber >= 7.0 and busNumber < 8.0:
            return  namesOfBuses[7]
    elif busNumber >= 8.0 and busNumber < 9.0:
            return  namesOfBuses[8]
    elif busNumber >= 9.0 and busNumber < 10.0:
            return  namesOfBuses[9]
    elif busNumber >= 10.0 and busNumber < 11.0:
            return  namesOfBuses[10]
    elif busNumber >= 11.0 and busNumber < 12.0:
            return  namesOfBuses[11]
    elif busNumber >= 12.0 and busNumber < 13.0:
            return  namesOfBuses[12]
    elif busNumber >= 13.0 and busNumber < 14.0:
            return  namesOfBuses[13]
    elif busNumber >= 14.0:
            return  namesOfBuses[14]
```

U slijedećem kodu 11. je varijabla „problemFunction“ definirana pomoću parametara, odnosno zadan joj je problem pomoću parametara koji će u „CreateNetworkInOpenDSS“ skripti kao filtar biti smješten u OpenDSS simulaciju mreže.

Kod 11:

```
Def problemFunction( parameters, *data ):

    busName = getBusFromNum(parameters[0])
    harmName = getHarmFromNum(parameters[3])

    THDValue = CreateNetworkInOpenDSS.CreateNetwork(busName, parameters[1],
parameters[2], 1000.0, harmName)

    return  THDValue

if __name__ == '__main__':
     main()
```

Sada prelazimo na skriptu „CreateNetworkInOpenDSS“ kod koje se na početku definiraju varijable (Kod 12.) kao što su „THD1results = [ ]“ koje predstavljaju jednodimenzionalno polje koje je trenutno bez podataka, ali će prilikom izvršenja programa naredbom „append“ podatci biti dodani. Također je definirana varijabla „CreateNetwork“ sa svojim parametrima koji imaju sljedeća značenja:

1. filterBusName - mjesto u mreži na kojem se kreira filter
2. filterXl - induktivni dio filtra
3. filterR - otpor filtra
4. filterKVAR - snaga filtra
5. filterHarm - harmonik na koji je filtar ugođen

Nakon toga opet koristimo naredbe uvoza(„import“) gdje uvozimo „win32com.client“ koji je dio „pywin32“ interpretera i služi za ostvarivanje komunikacije sa OpenDSS-om, zatim još uvozimo vrijeme(„time“), numpy i matplotlib koje ćemo naknadno koristiti.

Kod 12:

```
THD1results = []
THD2results = []
THD3results = []
rresult = []
xlresult = []
def CreateNetwork(filterBusName,filterXl, filterR, filterKVAR, filterHarm
):
import win32com.client
import time
import numpy as np
import matplotlib.pyplot as plt
```

U kodu 13. su pretežno naredbe za ostvarivanje komunikacije sa OpenDSS-om te vršenje simulacije. Možemo vidjeti kako se pomoću Pythona kreiraju filtar i mjerno mjesto u OpenDSS-u.

Kod 13:

```
engine = win32com.client.Dispatch(„OpenDSSEngine.DSS“)
engine.Start(„0“)
engine.Text.Command = 'clear'
circuit = engine.ActiveCircuit

filename = 'C:\OpenDSS_DiplomskiProjekt\IEEE13Nodeckt_dilomski.dss'
engine.Text.Command = 'compile ' + filename
CapacitorCommandText = 'New Capacitor.FILTER1 Bus1=' + filterBusName + '
phases=3 kVAR='+str(filterKVAR)+\
                       ' kV=4.16 xl=' + str(filterXl) + ' R= ' +
str(filterR) + ' Harm=' + str(filterHarm)+\
                       ' Numsteps=2 conn=wye'
engine.Text.Command = CapacitorCommandText

engine.Text.Command = 'New monitor.M1 Line.671680'
engine.Text.Command = 'Solve'
engine.Text.Command = 'Solve Mode=harmonics'
engine.Text.Command = 'Show mon M1'
```

Ovdje(Kod 14.) uvozimo interpreter za čitanje CSV datoteka te definiramo varijable napona „listOfHarmonicVoltage1 = [ ]“ koje kopiraju vrijednosti pojedinog napona i pohranjuju u polje varijable.

Kod 14:

```
import csv
listOfHarmonicVoltage1 = []
listOfHarmonicVoltage2 = []
listOfHarmonicVoltage3 = []
with open('C:\OpenDSS_DiplomskiProjekt\IEEE13Nodeckt_Mon_m1_1.csv', 'rb')
as f:
    reader = csv.reader(f)
    for row in reader:
        if row[0].isdigit():
            listOfHarmonicVoltage1.append(float(row[2]))
            listOfHarmonicVoltage2.append(float(row[4]))
            listOfHarmonicVoltage3.append(float(row[6]))
    f.close()
```

Kod sljedećeg izvornog koda prvo uvozimo „math“ kako bismo mogli koristiti matematičke operacije kao što su množenje, dijeljenje, zbrajanje, oduzimanje, korjenovanje, integriranje, deriviranje i slično. Prema formuli (8.2) računamo THD_U za svaku od faza(L1, L2, L3) te se računa ju njihovi relativni naponi po formuli (8.3). Na kraju se u varijablu npr. „THD1results“ spremaju svi rezultati THD-a za fazu L1. Zbog vrlo velike brzine simulacije dolazilo je do pogrešaka pri otvaranju CSV datoteka pa je bilo potrebo staviti vremensko kašnjenje nakon jedne iteracije(„*time delay*“).

$$THD_U = \frac{\sqrt{U_2{}^2 + U_3{}^2 + U_4{}^2 + \cdots + U_N{}^2}}{U_1} \tag{8-2}$$

$$U_{r\%} = \frac{U_h}{U_1} * 100\% \tag{8-3}$$

Kod 15:

```
import math
harmonicsValueSum1 = 0.0
for i in range(1,(len(listOfHarmonicVoltage1)-1)):
    harmonicsValueSum1 +=
(listOfHarmonicVoltage1[i]*listOfHarmonicVoltage1[i])

harmonicsValueSum2 = 0.0
for i in range(1,(len(listOfHarmonicVoltage2)-1)):
    harmonicsValueSum2 += (listOfHarmonicVoltage2[i] *
listOfHarmonicVoltage2[i])

harmonicsValueSum3 = 0.0
for i in range(1,(len(listOfHarmonicVoltage3)-1)):
    harmonicsValueSum3 += (listOfHarmonicVoltage3[i] *
listOfHarmonicVoltage3[i])
```

```
THDphase1 = math.sqrt(harmonicsValueSum1) / listOfHarmonicVoltage1[0] *
100.0
THDphase2 = math.sqrt(harmonicsValueSum2) / listOfHarmonicVoltage2[0] *
100.0
THDphase3 = math.sqrt(harmonicsValueSum3) / listOfHarmonicVoltage3[0] *
100.0

relativV31 = listOfHarmonicVoltage1[1] / listOfHarmonicVoltage1[0] * 100.0
relativV32 = listOfHarmonicVoltage2[1] / listOfHarmonicVoltage2[0] * 100.0
relativV33 = listOfHarmonicVoltage3[1] / listOfHarmonicVoltage3[0] * 100.0
relativV51 = listOfHarmonicVoltage1[2] / listOfHarmonicVoltage1[0] * 100.0
relativV52 = listOfHarmonicVoltage2[2] / listOfHarmonicVoltage2[0] * 100.0
relativV53 = listOfHarmonicVoltage3[2] / listOfHarmonicVoltage3[0] * 100.0
relativV71 = listOfHarmonicVoltage1[3] / listOfHarmonicVoltage1[0] * 100.0
relativV72 = listOfHarmonicVoltage2[3] / listOfHarmonicVoltage2[0] * 100.0
relativV73 = listOfHarmonicVoltage3[3] / listOfHarmonicVoltage3[0] * 100.0
relativV91 = listOfHarmonicVoltage1[4] / listOfHarmonicVoltage1[0] * 100.0
relativV92 = listOfHarmonicVoltage2[4] / listOfHarmonicVoltage2[0] * 100.0
relativV93 = listOfHarmonicVoltage3[4] / listOfHarmonicVoltage3[0] * 100.0
relativV111 = listOfHarmonicVoltage1[5] / listOfHarmonicVoltage1[0] * 100.0
relativV112 = listOfHarmonicVoltage2[5] / listOfHarmonicVoltage2[0] * 100.0
relativV113 = listOfHarmonicVoltage3[5] / listOfHarmonicVoltage3[0] * 100.0
relativV131 = listOfHarmonicVoltage1[6] / listOfHarmonicVoltage1[0] * 100.0
relativV132 = listOfHarmonicVoltage2[6] / listOfHarmonicVoltage2[0] * 100.0
relativV133 = listOfHarmonicVoltage3[6] / listOfHarmonicVoltage3[0] * 100.0
THD1results.append(THDphase1)
THD2results.append(THDphase2)
THD3results.append(THDphase3)
rresult.append(filterR)
xlresult.append(filterXl)
time.sleep(0.1)
```

U posljednjem kodu uvozimo alat za iscrtavanje grafova „matplotlib.pyplot“ koji ćemo naknadno koristiti. Zatim definiramo broj iteracija u ovoj skripti te pomoću „if-else“ naredbi stvaramo ograničavamo genetski algoritam da vrši nove iteracije dok god uvjeti norme EN 50160 ne budu zadovoljeni. U konačnici su zadane naredbe „print()“ za prikazivanje rezultata optimizacije genetskim algoritmom te naredbe za grafički prikaz(plotanje).

Kod 16:

```
import matplotlib.pyplot as plt
if len(listOfHarmonicVoltage1) == 0:
    print('Greska pri trazeju harmonika')
    return  -1.0
else:
    BrojIteracije = np.arange(0, (len(THD1results)))
if (THDphase1 < 9.0) and (THDphase2 < 10.0) and (THDphase3 < 10.0) and
(relativV31 <5) and (relativV32 <5) and \
        (relativV33 <5) and (relativV51 <6) and (relativV52 <6) and
(relativV53 <6) and (relativV71 <5) \
        and (relativV72 <5) and (relativV73 <5) and (relativV91 <1.5) and
(relativV92 <1.5) and (relativV93 <1.5) \
        and (relativV111 <3.5) and (relativV112 <3.5) and (relativV113
<3.5) and (relativV131 <3) and \
        (relativV132 <3) and (relativV133 <3):
    print ('Rezultat s THD vrijednostima pojedine faze ispod 8% je:' +
'THD1:' + str(
```

```
        THDphase1) + ', THD2:' + str(THDphase2) + ', THD3:' + str(
        THDphase3) + ', Bus:' + filterBusName + ', Xl:' + str(filterXl) +
', R:' + str(filterR) + ', HO:' + str(
        filterHarm) + '.')
    print('Relativni napon za fazu L1 su:' + ' 3. harmonik: ' +
str(relativV31) + '% 5. harmonik: '
          + str(relativV51) +'% 7. harmonik: ' + str(relativV71) + '% 9.
harmonik: ' + str(relativV91) +
          '% 11. harmonik: ' + str(relativV111) + '% 13. harmonik: '+
str(relativV131) +'%.')
    print('Relativni napon za fazu L2 su:' + ' 3. harmonik: ' +
str(relativV32) + '% 5. harmonik: '
          + str(relativV52) + '% 7. harmonik: ' + str(relativV72) + '% 9.
harmonik: ' + str(relativV92) +
          '% 11. harmonik: ' + str(relativV112) + '% 13. harmonik: ' +
str(relativV132) + '%.')
    print('Relativni napon za fazu L3 su:' + ' 3. harmonik: ' +
str(relativV33) + '% 5. harmonik: '
          + str(relativV53) + '% 7. harmonik: ' + str(relativV73) + '% 9.
harmonik: ' + str(relativV93) +
          '% 11. harmonik: ' + str(relativV113) + '% 13. harmonik: ' +
str(relativV133) + '%.')

        plt.subplot(3, 1, 1)
        plt.plot(BrojIteracije, THD1results, 'g')
        plt.title('THD/Broj iteracija')

        plt.subplot(3, 1, 1)
        plt.plot(BrojIteracije, THD2results, 'b')

        plt.subplot(3, 1, 1)
        plt.plot(BrojIteracije, THD3results, 'r')

        plt.subplot(3, 1, 2)
        plt.plot(BrojIteracije, rresult, 'r')
        plt.title('R/Broj iteracija')

        plt.subplot(3, 1, 3)
        plt.plot(BrojIteracije,xlresult, 'y')
        plt.title('Xl/Broj iteracija')
        plt.show()
        exit();

    return THDphase1
```

9. Rezultati optimizacije nesimetrične mreže evolucijskom metodom

Nakon izvršene cjelokupne simulacije sa konstantnom jalovom snagom pasivnog filtra od 1000 kVAr pri strategiji vršenja diferencijalne evolucije „best1bin“ dobijemo sljedeće rezultate *THD-a* :

Tablica 9.1. THD_U

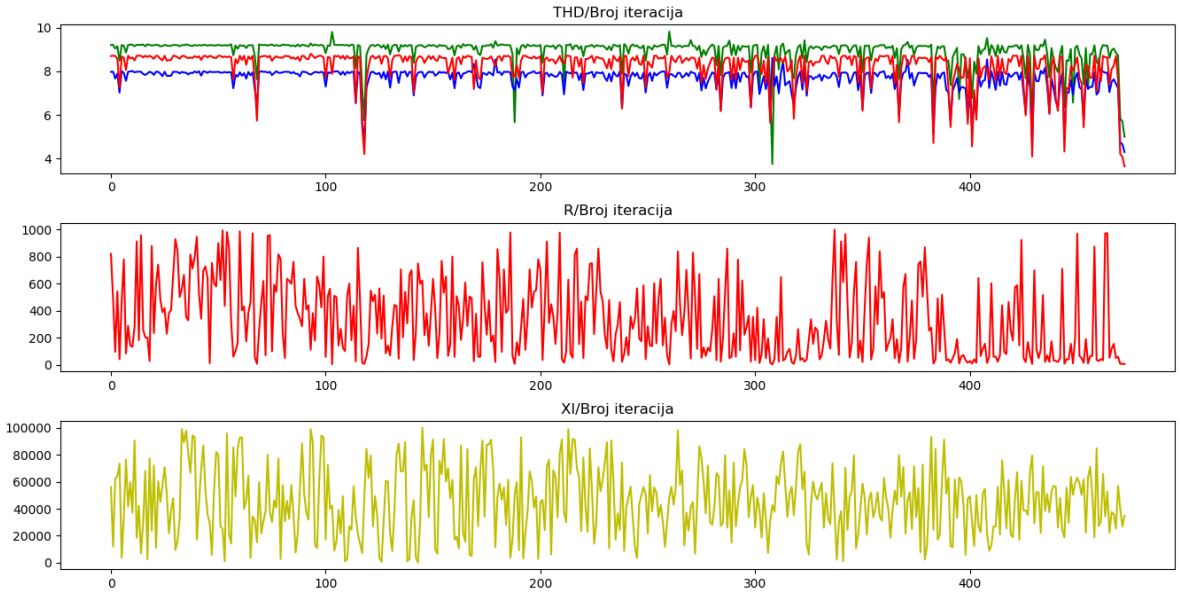
THD_U za fazu L1 (%)	THD_U za fazu L2 (%)	THD_U za fazu L3 (%)
5.00199561335	4.30689257431	3.64574519669

Genetskim algoritmom utvrđeno je najbolje mjesto u mreži za smještaj filtra čvor 675 te da parametri filtra trebaju biti sljedeći:

Tablica 9.2. *Parametri filtra*

Reaktancija filtra X_L (Ω)	Realni otpor filtra R (Ω)	Red harmonika na koji je ugođen filtar
34710.56414919857	5.127717137030117	11

Ovom strategijom rješavanja optimizacijskog problema bilo je dovoljno 472 iteracije kako bi se svi uvjeti norme EN 50160 ispunili. Na sljedećoj slici 9.1. možemo vidjeti kako je genetski algoritam vršio odabir parametara iz iteracije u iteraciju kako bi došao do rješenja optimizacije.



Slika 9.1. *Mijenjanje vrijednosti parametara po iteracijama*

Na prvom dijagramu vidimo kako se kretao *THD* sve tri faze kroz svaku iteraciju sve dok nije došao do krajnjeg rješenja optimizacije. U ostala dva dijagrama možemo vidjeti tijek izbora reaktancije i realnog otpora filtra te način kako je izbor utjecao na smanjenje *THD-a*. U sljedećoj tablici su prikazani su relativni naponi pojedine faze i pojedinog harmonika ugođeni prema normi EN 50160.

Tablica 9.3. *Relativni naponi pojedine faze*

Red harmonika	Relativni napon $U_{r\%}$ faze L1 (%)	Relativni napon $U_{r\%}$ faze L2 (%)	Relativni napon $U_{r\%}$ faze L3 (%)
3.	0.745778459205	0.677745461918	0.852761847014
5.	1.15890408676	3.18130107292	2.53838120984
7.	3.35477608023	2.17416256004	1.39622125094
9.	0.29510872056	0.216464584243	0.151655255468
11.	3.43206996113	1.7875873308	2.03677269397
13.	1.68811219112	1.41702404716	1.07815618605

Promjenom strategije traženja optimalnog rješenja na „best1exp“ povećao se broj iteracija na 796 te smo kao rješenje optimizacije dobili *THD*:

Tablica 9.4. THD_U

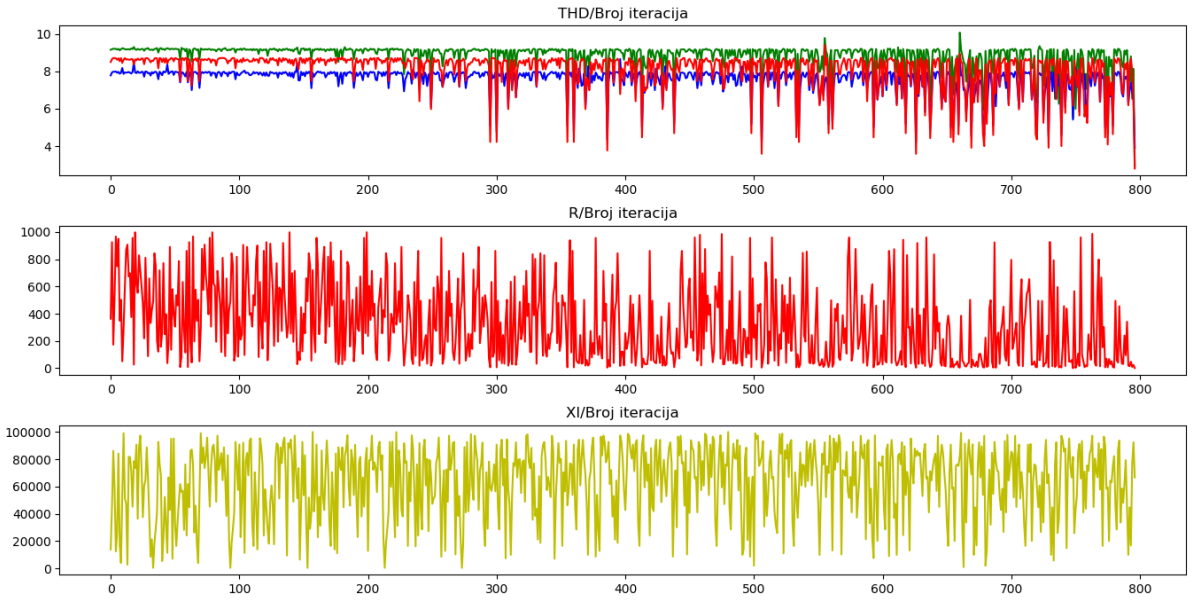
THD_U za fazu L1 (%)	THD_U za fazu L2 (%)	THD_U za fazu L3 (%)
2.93277465913	3.90448027059	2.80005578257

Gdje je za najbolje mjesto u mreži za smještaj filtra ponovno odabran čvor 675 sa parametrima filtra:

Tablica 9.5. *Parametri filtra*

Reaktancija filtra X_L (Ω)	Realni otpor filtra R (Ω)	Red harmonika na koji je ugođen filtar
66628.84155555915	1.1124352352467781	11

Na slici 9.2. možemo vidjeti kako se *THD* mijenjao sa brojem iteracija, kako se vršila promjena veličina parametara filtra te način na koji je to utjecalo na *THD*.



Slika 9.2. *Mijenjanje vrijednosti parametara po iteracijama*

I u konačnici relativni naponi pojedinih faza:

Tablica 9.6. *Relativni naponi pojedine faze*

Red harmonika	Relativni napon $U_{r\%}$ faze L1 (%)	Relativni napon $U_{r\%}$ faze L2 (%)	Relativni napon $U_{r\%}$ faze L3 (%)
3.	1.00140577159	1.01171001119	1.10073242291
5.	1.84567381093	3.55771501367	2.33938831078
7.	1.55388234902	1.04243179841	0.412237161492
9.	0.130842294953	0.0518553473776	0.0576490702239
11.	1.32671466591	0.688999192145	0.991312240117
13.	0.955853766383	0.777544742729	0.995352175106

Kako bi se mogli usporediti rezultati optimizacije napravili smo izračun *THD-a* mreže bez ugrađenog filtra u bilo koji čvor mreže. *THD* pojedine faze dan je u sljedećoj tablici:

Tablica 9.7. THD_U

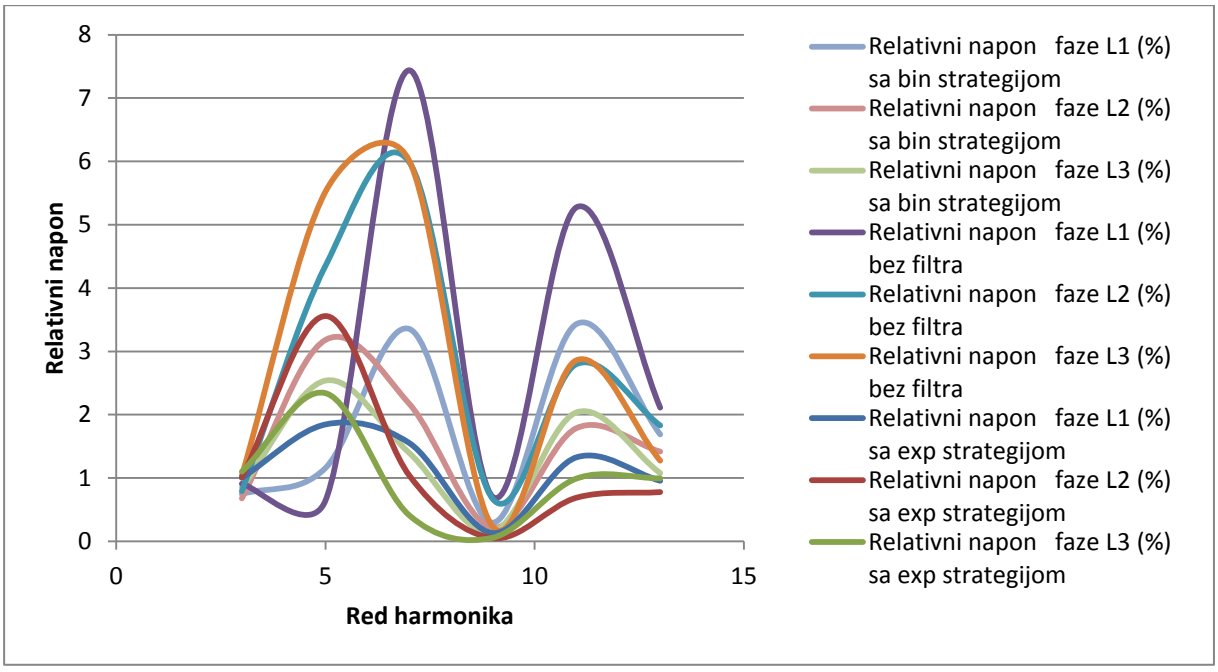
THD_U za fazu L1 (%)	THD_U za fazu L2 (%)	THD_U za fazu L3 (%)
9.21425579359	7.98370308137	8.71603900167

Može se vidjeti kako *THD* prve i treće faze prelazi granicu zadanu normom EN 50160, a iz sljedeće tablice se vidi kako su relativni naponi 7. i 11. harmonika prve faze, 7. harmonika druge faze i 7. harmonika treće faze prelaze zadane vrijednosti norme.

Tablica 9.8. *Relativni naponi pojedine faze*

Red harmonika	Relativni napon $U_{r\%}$ faze L1 (%)	Relativni napon $U_{r\%}$ faze L2 (%)	Relativni napon $U_{r\%}$ faze L3 (%)
3.	0.90905124023	0.79163181256	1.03656189589
5.	0.64617763562	4.35283428924	5.52355185032
7.	7.4387934504	5.99049033224	6.0126260011
9.	0.701628926688	0.667569871643	0.227764688622
11.	5.2754771336	2.79893616595	2.86036747445
13.	2.11063992035	1.82998667206	1.2738107567

Iz sljedećeg grafa prikazanog slikom 9.3. možemo zaključiti kako se relativni naponi u mreži bez ugrađenog filtra daleko više ističu nad onima reduciranim filtrom. Relativni naponi gdje je optimizacija rađena „best1exp“ strategijom pokazuju se kao najkvalitetniji jer su u tom slučaju relativni naponi svih faza niži nego kod ostalih metoda.



Slika 9.3. *Usporedba svih relativnih napona*

10. ZAKLJUČAK

Velik problem današnjih nesimetričnih mreža je „onečišćenje“ istih djelovanjem svakog trošila koje se na mrežu spaja. Većina trošila su nelinearna i aktivna te se zbog toga sve učestalije pojavljuju značajno velike amplitude viših harmonika. Da bi se održala kvaliteta električne energije javlja se potreba za ugradnjom filtra koji će održavati električnu energiju „čistom“(bez velikog intenziteta viših harmonika). Kvaliteta električne energije održava se na način da se poštuju norme namijenjene za očuvanje kvalitete. Europska norma za očuvanje kvalitete električne energije je EN 50160 i ona definira granice harmoničkog utjecaja u mreži.

S obzirom na to da je cijena aktivnih i hibridnih filtara za uklanjanje viših harmonika vrlo velika i u usporedbi s pasivnim filtrima, kao rješenje nudi se ugradnja pasivnih filtara. Međutim, ugradnjom velikog broja pasivnih filtara raste i cijena, a ekonomičnost pada. Zbog toga se postavlja pitanje kako nesimetričnu mrežu koja ne ispunjava uvjete norme prilagoditi granicama EN 50160 norme.

Kao rješenje predlaže se da na osnovu prikupljenih podataka pojedinog trošila mreže evolucijskom metodom(pomoću genetskog algoritma) pronađe jedan filtar koji će sa svojim parametrima smanjiti parametre mreže na one dozvoljene normom. Postoji više metoda traženja najadekvatnijeg rješenja, a neka od njih su opisana i obrazložena u ovome radu.

LITERATURA

[1]Harmonic filtering, 2016, http://www.electrical-installation.org/enwiki/Harmonic_filtering, (Pristupio 10.7.2018.)

[2]Definition of harmonics, 2016, http://www.electricalinstallation.org/enwiki/Definition_of_harmonics, (Pristupio 10.7.2018.)

[3]I. Rep, Pregled evolucijskih algoritama, Fakultet elektrotehnike i računarstva, Zagreb, 2014,http://www.zemris.fer.hr/~golub/ga/studenti/2014_rep/Pregled_evolucijskih_algoritama.html#1.0, (Pristupio 10.7.2018.)

[4]M. Golub, Genetski algoritam, ZEMRIS, 2004, http://www.zemris.fer.hr/~golub/ga/ga_skripta1.pdf, (Pristupio 10.7.2018.)

[5] M. Pielić, Pregled evolucijskih algoritama, Fakultet elektrotehnike i računarstva, Zagreb, 2007,http://www.zemris.fer.hr/~golub/ga/studenti/seminari/2007_pielic/Pregled%20evolucijskih%20algoritama%5B2007%5DPielic_Marko.htm, (Pristupio 10.7.2018.)

[6] N. Mišljenčević, B. Spasojević, Genetski algoritmi, Fakultet elektrotehnike i računarstva, Zagreb,2007,http://www.zemris.fer.hr/~golub/ga/studenti/projekt2007/ga/PR_Misljencevic_Spasojevic.pdf, (Pristupio 10.7.2018.)

[7] Dr. Shahin Rostami, Multi-Objective Problems, Youtube, 2017, https://www.youtube.com/watch?v=56JOMkPvoKs, (Pristupio 10.7.2018.)

[8] J. Petrović, Evolucijski algoritmi, Fakultet elektrotehnike i računarstva, Zagreb, 2007, http://www.zemris.fer.hr/~yeti/studenti/radovi/Petrovic_Juraj_Seminar.pdf, (Pristupio10.7.2018.)

[9] Simulation Tool – OpenDSS, Electric Power Research Institute, California, SAD, http://smartgrid.epri.com/SimulationTool.aspx , (Pristupio 10.7.2018.)

[10]Roger C. Dugan, Davis Montenegro, Electric Power Research Institute, The Open Distribution System Simulator™ (OpenDSS), April 2018

[11] Shammya Saha , Nathan Johnson , IEEE 13 Node Test Case , Arizona State University, https://www.xendee.com/Home/TestCase13Node, (Pristupio 10.7.2018.)

[12] Henryk Markiewicz, Antoni Klajn, Voltage Disturbances Standard EN 50160 - Voltage Characteristics in Public Distribution Systems, Wroclaw University of Technology, 2004,http://www.cdtechnics.be/542-standard-en-50160-voltage-characteristics-in.pdf, (Pristupio 10.7.2018.)

[13] M. Zeman, PyCharm, PRIRODOSLOVNO-MATEMATIČKI FAKULTET, Zagreb, 2014,http://www.phy.pmf.unizg.hr/RUNA/seminari/2014-11%20Marko%20Zeman.pdf , (Pristupio 10.7.2018.)

[14] Python (programski jezik),2016,https://hr.wikipedia.org/wiki/Python_(programski_jezik) (Pristupio 10.7.2018.)

[15] Sphinx,scipi.optimize.differential_evolution,2015,https://docs.scipy.org/doc/scipy0.15.1/reference/generated/scipy.optimize.differential_evolution.html, (Pristupio 10.7.2018.)

[16]rdugan,Spectrum,2014,https://github.com/tshort/OpenDSS/blob/master/Test/indmachtest/Spectrum.DSS, (Pristupio 10.7.2018.)

[17] E.F. Fuchs; M.A.S. Masooum: Power Quality in Power Systems and Electrical Machines, Academic Press, Amsterdam,2008

[18] Dragan Savic, Single-objective vs. Multiobjective Optimisation for Integrated Decision Support, University of Exeter, United Kingdom, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.122.7951&rep=rep1&type=pdf, (Pristupio 10.7.2018.)

[19] T. Patton, J. Wood, D. Woods, IEEE13Node_BusXY, Tennessee Technological University, 2009, OpenDSS\IEEETestCases\13Bus\IEEE13Node_BusXY

[20] Doc.dr.sc. Srđan Žutobradić, Harmonici u elektroenergetskom sustavu, Hrvatska energetska regulatorna agencija, http://www.riteh.uniri.hr/zav_katd_sluz/zee/nastava/svel/pep/download/PREDAVANJA%208.pdf, (Pristupio 10.7.2018.)

[21]M. Žgela, Primjena pasivnih filtera za smanjenje viših harmonika u distributivnim izvodima, Elektrotehnički fakultet Osijek, Osijek, 2015.

SAŽETAK

Optimizacija alokacije pasivnoga filtra u nesimetričnoj distributivnoj mreži sa trinaest čvorova temelji se na dva programska alata OpenDSS i Python gdje se kao programsko razvojno sučelje koristi PyCharm. Nesimetrična distributivna mreža s 13 čvorova uzeta je kao gotov primjer koji je došao kao prilog uz instalaciju OpenDSS-a. Pomoću programskog alata Python kreiran je genetski algoritam koji promjenom parametara filtra u mreži traži najoptimalnije rješenje kako bi se dobio što manji THD mreže. Također je pomoću Pythona u OpenDSS-u kreiran filtar koji svoje parametre mijenja ovisno o genetskom algoritmu. Kao rješenje optimizacije predstavlja se THD mreže koji treba biti unutar ograničenja zadanih normom EN 50160.

Ključne riječi: Optimizacija , OpenDSS, Python, distributivna mreža, filtar, genetski algoritam, THD , EN 50160

ABSTRACT

Optimisation of passive filter of asymmetric distribution network of 13 nodes is based on two software OpensDSS and Python, where development language is used Python. Asymmetric distributive network with 13 nodes is used as prepared example witch is coming with OpenDSS installation. In Python is developed genetic algorithm witch is searching for parameters of passive filter in order to find the best parameters for lowest THD value. Result of software optimisation are parameters and location of passive filter witch satisfy THD value inside limits set according to EN 50160 norm

Key words: Optimisation, OpenDSS, Python, distribution network, filter, genetic algorithm, THD, EN 50160

ŽIVOTOPIS

Rođen 13. travnja 1992. godine u Novoj Gradiški. Nakon završene Osnovne škole „Markovac“ Vrbova, upisuje srednju Tehničku školu u Slavonskom Brodu. Preddiplomski sveučilišni studij Elektrotehnike na Elektrotehničkom fakultetu u Osijeku upisao je 2011. godine. Titulu prvostupnika stječe 2015. godine, zatim iste godine upisuje diplomski studij elektrotehnike smjer Elektroenergetika, izborni blok Industrijska elektroenergetika.

PRILOZI

Izvorni kod opisivanja mreže u OpenDSS-u[19]:

```
new circuit.IEEE13Nodeckt
~ basekv=115 pu=1.0001 phases=3 bus1=SourceBus
~ Angle=30                                    ! advance angle 30 deg so result agree with published angle
~ MVAsc3=20000 MVASC1=21000    ! stiffen the source to approximate inf source


!SUB TRANSFORMER DEFINITION
! Although this data was given, it does not appear to be used in the test case results
! The published test case starts at 1.0 per unit at Bus 650. To make this happen, we will change the impedance
! on the transformer to something tiny by dividing by 1000 using the DSS in-line RPN math
New Transformer.Sub Phases=3 Windings=2   XHL=(8 1000 /)
~ wdg=1 bus=SourceBus   conn=delta  kv=115  kva=5000   %r=(.5 1000 /)
~ wdg=2 bus=650           conn=wye    kv=4.16  kva=5000   %r=(.5 1000 /)

! FEEDER 1-PHASE VOLTAGE REGULATORS

New Transformer.Reg1 phases=1 bank=reg1 XHL=0.01 kVAs=[1666 1666]
~ Buses=[650.1 RG60.1] kVs=[2.4  2.4] %LoadLoss=0.01
new regcontrol.Reg1  transformer=Reg1 winding=2  vreg=122  band=2  ptratio=20 ctprim=700  R=3   X=9

New Transformer.Reg2 phases=1 bank=reg1 XHL=0.01 kVAs=[1666 1666]
~ Buses=[650.2 RG60.2] kVs=[2.4  2.4] %LoadLoss=0.01
new regcontrol.Reg2  transformer=Reg2 winding=2  vreg=122  band=2  ptratio=20 ctprim=700  R=3   X=9

New Transformer.Reg3 phases=1 bank=reg1 XHL=0.01 kVAs=[1666 1666]
~ Buses=[650.3 RG60.3] kVs=[2.4  2.4] %LoadLoss=0.01
new regcontrol.Reg3  transformer=Reg3 winding=2  vreg=122  band=2  ptratio=20 ctprim=700  R=3   X=9


!TRANSFORMER DEFINITION
New Transformer.XFM1  Phases=3   Windings=2  XHL=2
~ wdg=1 bus=633       conn=Wye kv=4.16    kva=500    %r=.55     XHT=1
~ wdg=2 bus=634       conn=Wye kv=0.480    kva=500    %r=.55   XLT=1


!LINE CODES
redirect IEEELineCodes.DSS

// these are local matrix line codes
// corrected 9-14-2011
New linecode.mtx601 nphases=3 BaseFreq=50
~ rmatrix = (0.3465 | 0.1560 0.3375 | 0.1580 0.1535 0.3414 )
~ xmatrix = (1.0179 | 0.5017 1.0478 | 0.4236 0.3849 1.0348 )
~ units=mi
New linecode.mtx602 nphases=3 BaseFreq=50
~ rmatrix = (0.7526 | 0.1580 0.7475 | 0.1560 0.1535 0.7436 )
~ xmatrix = (1.1814 | 0.4236 1.1983 | 0.5017 0.3849 1.2112 )
~ units=mi
New linecode.mtx603 nphases=2 BaseFreq=50
~ rmatrix = (1.3238 | 0.2066 1.3294 )
~ xmatrix = (1.3569 | 0.4591 1.3471 )
~ units=mi
New linecode.mtx604 nphases=2 BaseFreq=50
~ rmatrix = (1.3238 | 0.2066 1.3294 )
~ xmatrix = (1.3569 | 0.4591 1.3471 )
~ units=mi
New linecode.mtx605 nphases=1 BaseFreq=50
~ rmatrix = (1.3292 )
~ xmatrix = (1.3475 )
~ units=mi

New linecode.mtx606 nphases=3 BaseFreq=50
~ rmatrix = (0.7982 | 0.3192 0.7891 | 0.2849 0.3192 0.7982 )
~ xmatrix = (0.4463 | 0.0328 0.4041 | -0.0143 0.0328 0.4463 )
~ Cmatrix = [257 | 0 257 | 0 0 257]  ! <--- This is too low by 1.5
~ units=mi
```

```
New CNDATA.250_1/3 k=13 DiaStrand=0.064 Rstrand=2.816666667 epsR=2.3
~ InsLayer=0.220 DiaIns=1.06 DiaCable=1.16 Rac=0.076705 GMRac=0.20568 diam=0.573
~ Runits=kft Radunits=in GMRunits=in

New LineGeometry.606 nconds=3 nphases=3 units=ft
~ cond=1 cncable=250_1/3 x=-0.5 h= -4
~ cond=2 cncable=250_1/3 x=0   h= -4
~ cond=3 cncable=250_1/3 x=0.5  h= -4

New Linecode.mtx606 nphases=3  Units=mi
~ Rmatrix=[0.791721  |0.318476  0.781649  |0.28345  0.318476  0.791721  ]
~ Xmatrix=[0.438352  |0.0276838  0.396697  |-0.0184204  0.0276838  0.438352  ]
~ Cmatrix=[383.948  |0  383.948  |0  0  383.948  ]
New linecode.mtx607 nphases=1 BaseFreq=50
~ rmatrix = (1.3425 )
~ xmatrix = (0.5124 )
~ cmatrix = [236]
~ units=mi

!LOAD DEFINITIONS
New Load.671 Bus1=671.1.2.3  Phases=3 Conn=Delta Model=1 kV=4.16   kW=1155 kvar=660
New Load.634a Bus1=634.1     Phases=1 Conn=Wye  Model=1 kV=0.277  kW=160   kvar=110
New Load.634b Bus1=634.2     Phases=1 Conn=Wye  Model=1 kV=0.277  kW=120   kvar=90
New Load.634c Bus1=634.3     Phases=1 Conn=Wye  Model=1 kV=0.277  kW=120   kvar=90
New Load.645 Bus1=645.2      Phases=1 Conn=Wye  Model=1 kV=2.4    kW=170   kvar=125
New Load.646 Bus1=646.2.3    Phases=1 Conn=Delta Model=2 kV=4.16   kW=230   kvar=132
New Load.692 Bus1=692.3.1    Phases=1 Conn=Delta Model=5 kV=4.16   kW=170   kvar=151
New Load.675a Bus1=675.1     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=485   kvar=190
New Load.675b Bus1=675.2     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=68   kvar=60
New Load.675c Bus1=675.3     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=290   kvar=212
New Load.611 Bus1=611.3      Phases=1 Conn=Wye  Model=5 kV=2.4  kW=170   kvar=80
New Load.652 Bus1=652.1      Phases=1 Conn=Wye  Model=2 kV=2.4  kW=128   kvar=86
New Load.670a Bus1=670.1     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=17    kvar=10
New Load.670b Bus1=670.2     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=66    kvar=38
New Load.670c Bus1=670.3     Phases=1 Conn=Wye  Model=1 kV=2.4  kW=117   kvar=68

!CAPACITOR DEFINITIONS
New Capacitor.Cap1 Bus1=675 phases=3 kVAR=600 kV=4.16
New Capacitor.Cap2 Bus1=611.3 phases=1 kVAR=100 kV=2.4

!Bus 670 is the concentrated point load of the distributed load on line 632 to 671 located at 1/3 the distance from node
632

!LINE DEFINITIONS
New Line.650632    Phases=3 Bus1=RG60.1.2.3   Bus2=632.1.2.3  LineCode=mtx601 Length=2000 units=ft
New Line.632670    Phases=3 Bus1=632.1.2.3    Bus2=670.1.2.3  LineCode=mtx601 Length=667  units=ft
New Line.670671    Phases=3 Bus1=670.1.2.3    Bus2=671.1.2.3  LineCode=mtx601 Length=1333 units=ft
New Line.671680    Phases=3 Bus1=671.1.2.3    Bus2=680.1.2.3  LineCode=mtx601 Length=1000 units=ft
New Line.632633    Phases=3 Bus1=632.1.2.3    Bus2=633.1.2.3  LineCode=mtx602 Length=500  units=ft
New Line.632645    Phases=2 Bus1=632.3.2      Bus2=645.3.2    LineCode=mtx603 Length=500  units=ft
New Line.645646    Phases=2 Bus1=645.3.2      Bus2=646.3.2    LineCode=mtx603 Length=300  units=ft
New Line.692675    Phases=3 Bus1=692.1.2.3    Bus2=675.1.2.3  LineCode=mtx606 Length=500  units=ft
New Line.671684    Phases=2 Bus1=671.1.3      Bus2=684.1.3    LineCode=mtx604 Length=300  units=ft
New Line.684611    Phases=1 Bus1=684.3        Bus2=611.3      LineCode=mtx605 Length=300  units=ft
New Line.684652    Phases=1 Bus1=684.1        Bus2=652.1      LineCode=mtx607 Length=800  units=ft

!SWITCH DEFINITIONS
New Line.671692    Phases=3 Bus1=671   Bus2=692  Switch=y  r1=1e-4 r0=1e-4 x1=0.000 x0=0.000 c1=0.000 c0=0.000

Set Voltagebases=[115, 4.16, .48]
calcv
BusCoords IEEE13Node_BusXY.csv

! DEFIE FILTERS
New Capacitor.FILTER1 Bus1=671.1.2.3 phases=3 kVAR=600 kV=4.16 xl=2.5 R= 0.187 Harm=7 Numsteps=2 conn=wye

New monitor.M1 Line.671680

Solve
Solve Mode=harmonics
Show mon M1
```

Izvorni kod GeneticAlg python datoteke:

```
import CreateNetworkInOpenDSS
import numpy as np
from scipy.optimize import differential_evolution

def main():

    bounds = [(0.0, 15.0), (0.0, 100000.0), (0.0, 1000.0), (0.0, 7.0)]
    # scipy.optimize.differential_evolution(func, bounds, args=(),
    #  strategy='best1bin', maxiter=1000, popsize=15,
    #  tol=0.01, mutation=(0.5, 1), recombination=0.7, seed=None,
    # callback=None, disp=False, polish=True,
    #  init='latinhypercube', atol=0)
    # Maksimalni broj iteracija je: (maxiter + 1) * popsize * len(bounds) =
    #  (maxiter + 1) * popsize *4

    result = differential_evolution(problemFunction, bounds, (),
                                    'best1bin', 100, 15, 0.01, (0.5, 1),
                                    0.7, None, None, False, True,
                                    'latinhypercube', 0)
    #print(result.x)
    BrojIteracije = np.arange(0, (len(CreateNetworkInOpenDSS.THD1results)))
    print ('Broj iteracija je: ' + str(len(BrojIteracije)-1))


def getHarmFromNum( harmNumber ):
    harmOrder = ['1', '3', '5', '7', '9', '11', '13']
    if harmNumber < 1.0:
        return  harmOrder[0]
    elif harmNumber >= 1.0 and harmNumber < 2.0:
        return  harmOrder[1]
    elif harmNumber >= 2.0 and harmNumber < 3.0:
        return  harmOrder[2]
    elif harmNumber >= 3.0 and harmNumber < 4.0:
        return  harmOrder[3]
    elif harmNumber >= 4.0 and harmNumber < 5.0:
        return  harmOrder[4]
    elif harmNumber >= 5.0 and harmNumber < 6.0:
        return  harmOrder[5]
    elif harmNumber >= 6.0 and harmNumber < 7.0:
        return  harmOrder[6]


def getBusFromNum( busNumber ):
    namesOfBuses = ['650.1.2.3', 'RG60.1.2.3', '646.1.2.3', '645.1.2.3',
                    '632.1.2.3', '633.1.2.3', '634.1.2.3',
                    '670.1.2.3', '611.1.2.3', '684.1.2.3', '671.1.2.3',
                    '692.1.2.3', '675.1.2.3', '652.1.2.3', '680.1.2.3']
    if busNumber < 1.0:
        return namesOfBuses[0]
    elif busNumber >= 1.0 and busNumber < 2.0:
            return  namesOfBuses[1]
    elif busNumber >= 2.0 and busNumber < 3.0:
```

```
            return  namesOfBuses[2]
    elif busNumber >= 3.0 and busNumber < 4.0:
            return  namesOfBuses[3]
    elif busNumber >= 4.0 and busNumber < 5.0:
            return  namesOfBuses[4]
    elif busNumber >= 5.0 and busNumber < 6.0:
            return  namesOfBuses[5]
    elif busNumber >= 6.0 and busNumber < 7.0:
            return  namesOfBuses[6]
    elif busNumber >= 7.0 and busNumber < 8.0:
            return  namesOfBuses[7]
    elif busNumber >= 8.0 and busNumber < 9.0:
            return  namesOfBuses[8]
    elif busNumber >= 9.0 and busNumber < 10.0:
            return  namesOfBuses[9]
    elif busNumber >= 10.0 and busNumber < 11.0:
            return  namesOfBuses[10]
    elif busNumber >= 11.0 and busNumber < 12.0:
            return  namesOfBuses[11]
    elif busNumber >= 12.0 and busNumber < 13.0:
            return  namesOfBuses[12]
    elif busNumber >= 13.0 and busNumber < 14.0:
            return  namesOfBuses[13]
    elif busNumber >= 14.0:
            return  namesOfBuses[14]

def problemFunction( parameters, *data ):
    """
    Funckija opisuje problem koji se optimizacijom treba rijesiti.
Striktura
     je prilagodjena SciPy.
    :param parameters: [ filterBusName, filterXl, filterR:, filterHarm]
    :param filterBusName: Mjesto u mrezi na kojem se kreira filter
    :param filterXl: Induktivni dio filtera
    :param filterR: Otpor filtera
    :param filterHarm: Harmonik na koji je filter ugodjen
    :return: THD vrijednost harmonickog izoblicenja faznog napona. Vraca -
1.0
    ako je doslo do pogreske u izracunu
    """

    # Pretvori ime busa iz broja u string jer differential_evolution daje
samo
    # brojeve kao parametar
    busName = getBusFromNum(parameters[0])
    harmName = getHarmFromNum(parameters[3])
    # Poziv funkcije koja vrsi jednu simulaciju u mreze s zadanim
parametrima
    # Snaga filtera je fiksno postavljena na 5000 kVAr
    THDValue = CreateNetworkInOpenDSS.CreateNetwork(busName,parameters[1],
                                                    parameters[2], 1000.0,
                                                    harmName)
```

```
    return  THDValue

if __name__ == '__main__':
     main()
```

Izvorni kod CreateNetworkInOpenDSS python datoteke:

```
THD1results = []
THD2results = []
THD3results = []
rresult = []
xlresult = []
def CreateNetwork(filterBusName,filterXl, filterR, filterKVAR,
                  filterHarm ):
    """
    Funckija ostvaruje komunikaciju izmedu Pythona i OpenDSS-a.
     Zatim pokrece OpenDSS skriptu u kojoj je definirana
    energetska mreza.
    Nakon sto je kreirana mreza, funkcija kreira filtar prema
     parametrima zadanim u argumentima funkcije.
    Na kraju pokrece simulaciju armonika u mrezi i vraca
     rezultat u .csv fileu.
    :param filterBusName: Mjesto u mrezi na kojem se kreira
     filter
    :param filterXl: Induktivni dio filtera
    :param filterR: Otpor filtera
    :param filterKVAR: Snaga filtera
    :param filterHarm: Harmonik na koji je filter ugodjen
    :return: THD vrijednost harmonickog izoblicenja faznog
    napona. Vraca -1.0 ako je doslo do pogreske u izracunu
    """

    # Uvezi win32com iz pywin32 paketa.
    import win32com.client
    import time
    import numpy as np
    import matplotlib.pyplot as plt

    # Definiraj komuniakciju s OpenDSS-om.
    engine = win32com.client.Dispatch("OpenDSSEngine.DSS")
    engine.Start("0")
    # Posalji komandu za ciscenje radnog prostora u OpenDSS-u
    engine.Text.Command = 'clear'
    # Aktiviraj rad s krugovima
    circuit = engine.ActiveCircuit
    # Definiraj putanju do OpenDSS filea s konfiguracijom mreze
    # bez filtera
    filename = 'C:\OpenDSS_DiplomskiProjekt\IEEE13Nodeckt_dilomski.dss'
    engine.Text.Command = 'compile ' + filename
    # Kreiraj filtar s zadanim parametrima
    CapacitorCommandText = 'New Capacitor.FILTER1 Bus1=' + filterBusName +
\
                           ' phases=3 kVAR='+str(filterKVAR)+\
                           ' kV=4.16 xl=' + str(filterXl) + ' R= '\
                           + str(filterR) + ' Harm=' + str(filterHarm)+\
                           ' Numsteps=2 conn=wye'
    engine.Text.Command = CapacitorCommandText

    # Vrati rezultate simulacije
```

```
engine.Text.Command = 'New monitor.M1 Line.671680'
engine.Text.Command = 'Solve'
engine.Text.Command = 'Solve Mode=harmonics'
engine.Text.Command = 'Show mon M1'

# Procitaj .csv file i izracunaj THD vrijednost
import csv
# Liste koje sadrze fazne vrijednosti napona pojedinog harmonika
#  faza
listOfHarmonicVoltage1 = []
listOfHarmonicVoltage2 = []
listOfHarmonicVoltage3 = []
with open('C:\OpenDSS_DiplomskiProjekt\IEEE13Nodeckt_Mon_m1_1.csv',
          'rb') as f:
    reader = csv.reader(f)
    for row in reader:
        if row[0].isdigit(): # Provjeri dali postoje podaci u tom
            # redu .csv filea
            # if int(float(row[1])) == int(filterHarm):
            #     listOfHarmonicVoltage.append(row[2])
            listOfHarmonicVoltage1.append(float(row[2]))
            listOfHarmonicVoltage2.append(float(row[4]))
            listOfHarmonicVoltage3.append(float(row[6]))
    f.close()



import math
harmonicsValueSum1 = 0.0
for i in range(1,(len(listOfHarmonicVoltage1)-1)):
    harmonicsValueSum1 += (listOfHarmonicVoltage1[i]*
                           listOfHarmonicVoltage1[i])

harmonicsValueSum2 = 0.0
for i in range(1,(len(listOfHarmonicVoltage2)-1)):
    harmonicsValueSum2 += (listOfHarmonicVoltage2[i] *
                           listOfHarmonicVoltage2[i])

harmonicsValueSum3 = 0.0
for i in range(1,(len(listOfHarmonicVoltage3)-1)):
    harmonicsValueSum3 += (listOfHarmonicVoltage3[i] *
                           listOfHarmonicVoltage3[i])


THDphase1 = math.sqrt(harmonicsValueSum1) / \
            listOfHarmonicVoltage1[0] * 100.0
THDphase2 = math.sqrt(harmonicsValueSum2) / \
            listOfHarmonicVoltage2[0] * 100.0
THDphase3 = math.sqrt(harmonicsValueSum3) /\
            listOfHarmonicVoltage3[0] * 100.0
relativV31 = listOfHarmonicVoltage1[1] / \
             listOfHarmonicVoltage1[0] * 100.0
relativV32 = listOfHarmonicVoltage2[1] /\
             listOfHarmonicVoltage2[0] * 100.0
relativV33 = listOfHarmonicVoltage3[1] / \
             listOfHarmonicVoltage3[0] * 100.0
relativV51 = listOfHarmonicVoltage1[2] / \
             listOfHarmonicVoltage1[0] * 100.0
relativV52 = listOfHarmonicVoltage2[2] / \
             listOfHarmonicVoltage2[0] * 100.0
```

```
    relativV53 = listOfHarmonicVoltage3[2] / \
                 listOfHarmonicVoltage3[0] * 100.0
    relativV71 = listOfHarmonicVoltage1[3] / \
                 listOfHarmonicVoltage1[0] * 100.0
    relativV72 = listOfHarmonicVoltage2[3] / \
                 listOfHarmonicVoltage2[0] * 100.0
    relativV73 = listOfHarmonicVoltage3[3] / \
                 listOfHarmonicVoltage3[0] * 100.0
    relativV91 = listOfHarmonicVoltage1[4] / \
                 listOfHarmonicVoltage1[0] * 100.0
    relativV92 = listOfHarmonicVoltage2[4] / \
                 listOfHarmonicVoltage2[0] * 100.0
    relativV93 = listOfHarmonicVoltage3[4] / \
                 listOfHarmonicVoltage3[0] * 100.0
    relativV111 = listOfHarmonicVoltage1[5] / \
                  listOfHarmonicVoltage1[0] * 100.0
    relativV112 = listOfHarmonicVoltage2[5] / \
                  listOfHarmonicVoltage2[0] * 100.0
    relativV113 = listOfHarmonicVoltage3[5] / \
                  listOfHarmonicVoltage3[0] * 100.0
    relativV131 = listOfHarmonicVoltage1[6] / \
                  listOfHarmonicVoltage1[0] * 100.0
    relativV132 = listOfHarmonicVoltage2[6] / \
                  listOfHarmonicVoltage2[0] * 100.0
    relativV133 = listOfHarmonicVoltage3[6] / \
                  listOfHarmonicVoltage3[0] * 100.0
    THD1results.append(THDphase1)
    THD2results.append(THDphase2)
    THD3results.append(THDphase3)
    rresult.append(filterR)
    xlresult.append(filterXl)

    time.sleep(0.1)


    import matplotlib.pyplot as plt
    if len(listOfHarmonicVoltage1) == 0:
        print('Greska pri trazeju harmonika')
        return  -1.0
    else:
        # U slucaju da je THD svih faza pao ispod 8%
        # zaustavi simulaciju
        BrojIteracije = np.arange(0, (len(THD1results)))
        if (THDphase1 < 8.0) and (THDphase2 < 8.0) and \
                (THDphase3 < 8.0) and (relativV31 <5) and (relativV32 <5)\
                and (relativV33 <5) and (relativV51 <6) and (relativV52
<6)\
                and (relativV53 <6) and (relativV71 <5) \
                and (relativV72 <5) and (relativV73 <5) and (relativV91
<1.5)\
                and (relativV92 <1.5) and (relativV93 <1.5) \
                and (relativV111 <3.5) and (relativV112 <3.5) and\
                (relativV113 <3.5) and (relativV131 <3) and \
                (relativV132 <3) and (relativV133 <3):
            print ('Rezultat s THD vrijednostima pojedine faze ispod 8%
je:' +
                   'THD1:' + str(THDphase1) + ', THD2:' + str(THDphase2) +
                   ', THD3:' + str( THDphase3) + ', Bus:' + filterBusName +
                   ', Xl:' + str(filterXl) + ', R:' + str(filterR) + ',
HO:' +
                   str(filterHarm) + '.')
```

```
            print('Relativni napon za fazu L1 su:' + ' 3. harmonik: '
                  + str(relativV31) + '% 5. harmonik: '
                  + str(relativV51) +'% 7. harmonik: ' + str(relativV71) +
                  '% 9. harmonik: ' + str(relativV91) +
                  '% 11. harmonik: ' + str(relativV111) + '% 13. harmonik:
'
                  + str(relativV131) +'%.')
            print('Relativni napon za fazu L2 su:' + ' 3. harmonik: '
                  + str(relativV32) + '% 5. harmonik: '
                  + str(relativV52) + '% 7. harmonik: ' + str(relativV72) +
                  '% 9. harmonik: ' + str(relativV92) +
                  '% 11. harmonik: ' + str(relativV112) + '% 13. harmonik:
'
                  + str(relativV132) + '%.')
            print('Relativni napon za fazu L3 su:' + ' 3. harmonik: ' +
                  str(relativV33) + '% 5. harmonik: '
                  + str(relativV53) + '% 7. harmonik: ' + str(relativV73) +
                  '% 9. harmonik: ' + str(relativV93) +
                  '% 11. harmonik: ' + str(relativV113) + '% 13. harmonik:
'
                  + str(relativV133) + '%.')
            print(BrojIteracije)
            plt.subplot(3, 1, 1)
            plt.plot(BrojIteracije, THD1results, 'g')
            plt.title('THD/Broj iteracija')

            plt.subplot(3, 1, 1)
            plt.plot(BrojIteracije, THD2results, 'b')

            plt.subplot(3, 1, 1)
            plt.plot(BrojIteracije, THD3results, 'r')

            plt.subplot(3, 1, 2)
            plt.plot(BrojIteracije, rresult, 'r')
            plt.title('R/Broj iteracija')

            plt.subplot(3, 1, 3)
            plt.plot(BrojIteracije,xlresult, 'y')
            plt.title('Xl/Broj iteracija')
            plt.show()
            exit();

        return THDphase1
```