Razvoj iOS aplikacije za simulaciju parametara nadzemnog voda

Vrhovac, Manuel

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

RAZVOJ IOS APLIKACIJE ZA SIMULACIJU PARAMETARA NADZEMNOG VODA

Završni rad

Manuel Vrhovac

Osijek, 2019.

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

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 10.09.2019.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Manuel Vrhovac
Studij, smjer:	Preddiplomski sveučilišni studij Elektrotehnika i informacijska tehnologija
Mat. br. studenta, godina upisa:	3797, 23.10.2018.
OIB studenta:	99112082236
Mentor:	Izv. prof. dr. sc. Predrag Marić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Razvoj iOS aplikacije za simulaciju parametara nadzemnog voda
Znanstvena grana rada:	**Elektroenergetika (zn. polje elektrotehnika)**
Predložena ocjena završnog 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: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	10.09.2019.
Datum potvrde ocjene Odbora:	25.09.2019.

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, 10.10.2019.

Ime i prezime studenta:	Manuel Vrhovac
Studij:	Preddiplomski sveučilišni studij Elektrotehnika i informacijska tehnologija
Mat. br. studenta, godina upisa:	3797, 23.10.2018.
Ephorus podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Razvoj iOS aplikacije za simulaciju parametara nadzemnog voda**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Predrag Marić

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:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

Ja, Manuel Vrhovac, OIB: 99112082236, student/ica na studiju: Preddiplomski sveučilišni studij Elektrotehnika i informacijska tehnologija, dajem suglasnost Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek da pohrani i javno objavi moj **završni rad**:

Razvoj iOS aplikacije za simulaciju parametara nadzemnog voda

u javno dostupnom fakultetskom, sveučilišnom i nacionalnom repozitoriju.

Osijek, 10.10.2019.

_____
potpis

Sadržaj

1. Uvod

Elektroenergetski sustav je složeni sustav koji dostavlja električnu energiju u domove, tvornice i svugdje gdje je ona potrebna [1]. Elektroenergetski sustav najčešće se sastoji od četiri jasno odvojene cjeline [1]:

1. Proizvodnja električne energije
2. Prijenos električne energije
3. Distribucija električne energije
4. Potrošnja električne energije

Za prijenos električne energije od elektrane do distribucijskih stanica koristi se nadzemni vod. On prenosi val napona i struje sa jednog kraja na drugi [1]. Nadzemni vod sastoji se od vodiča koji su u svim točkama duljine jednakog presjeka [1]. Kao izolator između vodiča koristi se zrak ili dielektrični medij [2]. Zbog sigurnosnih razloga, razmak između linija i tla je mnogo veći. Vodiče nadzemnog voda nose dalekovodni stupovi građeni od čelika koji pružaju visoku čvrstoću pri zadatku zatezanja vodiča [2].

Performanse nadzemnog voda ovise o njegovim parametrima. Svaki nadzemni vod ima 4 primarna parametra: otpornost, induktivnost, kapacitivnost i vodljivost [13]. Ovi parametri raspoređeni su uniformno po cijeloj dužini voda zbog čega se još nazivaju i raspoređenim (distributiranim) parametrima nadzemnog voda [13]. Induktivnost i otpornost čine serijsku impedanciju a kapacitivnost i vodljivost paralelnu. Ispitivanje performansi voda uključuje računanje struje i napona na kraju voda, faktor snage, gubitak snage u vodu i učinkovitost prijenosa pri stablinom ili prijelaznom stanju voda [13].

U ovom završnom radu modelirat će se nadzemni vod u objektno-orijentiranom programskom jeziku Swift. Pri tom će se definirati sva bitna svojstva nadzemnog voda koja određuju iznose njegovih primarnih parametara.

Zadatak ovog završnog rada je izraditi aplikaciju koja će omogućiti da se za zadanu izvedbu nadzemnog voda (materijal, presjek vodiča, geometrijski raspored) i polazni val napona i struje trenutno prikazuju izlazni valovi i gubitak snage. Ovime bi se pružio instantni uvid u performanse bilo kakvog trofaznog voda.

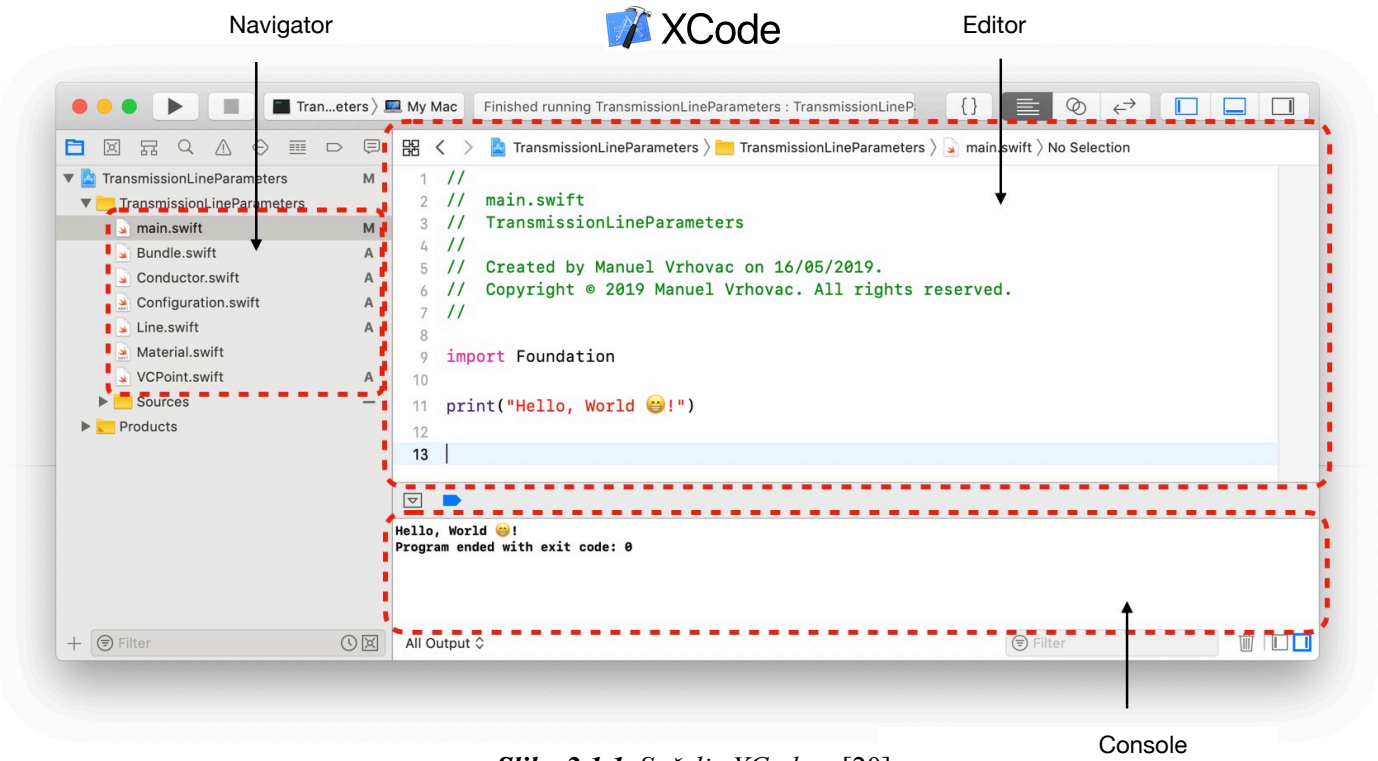
2. Korištene tehnologije

2.1. Razvojno okruženje *XCode*

Xcode je integrirano razvojno okruženje također kreirano od strane Apple-a namijenjeno za izradu aplikacija za Appleove platforme iOS, macOS, tvOS i watchOS. Xcode je program koji sadržava sve potrebno za izradu, testiranje i optimiziranje aplikacija, a naposljetku i distribuciju na App Store-u [3].

Xcode podržava razne jezike kao što su Objective-C, Swift, C++, C, Javascript itd. Sučelje XCode-a obuhvaća uređivanje kôda (Editor), izradu korisničkog sučelja (Interface Builder) i testiranje (simulator iOS, tvOS i watchOS uređaja)[3]. Ovaj rad ograničit će se na pisanje kôda u Editor-u koristeći jezik Swift u projektu namijenjenom za izvršenje u komandnom sučelju (Command Line Interface).

Xcode je podržan isključivo na Apple-ovom operativnom sustavu macOS, a inačica korištena u ovom završnom radu 10.2 službeno je objavljena u ožujku 2019. godine [3].



Slika 2.1.1. *Sučelje XCode-a* [20]

Za ovaj završni rad najbitniji elementi sučelja XCode-a prikazani su na slici 2.1.1, a to su:

- Navigator - područje u kojem se odabire datoteka sa kôdom (eng. *sourcefile*) koja se uređuje
- Editor - područje u kojem se uređuje kôd odabrane datoteke
- Konzola - područje u koje se ispisuje izlazna vrijednost nakon završetka pokretanja programa

2.2. Programski jezik Swift

Swift je objektno orijentiran programski jezik predstavljen od strane tehnološkog diva Apple-a na Worldwide Developers Conference (WWDC) događaju 2014. godine [4]. Osmišljen je kao nasljednik za Objective-C — dotadašnji službeni programski jezik koji se koristio još od davne 1984 za razvoj softwarea za Appleov operativni sustav Mac OS X.

Swift ispravlja glavne mane Objective-C-a i rezultira modernim programskim jezikom koji je intuitivniji, čitkiji, sigurniji i boljih performansi. Od 2015. Od svoje verzije 2.2, u 2015. godini Swift je postao open-source što znači da developeri izvan Apple-a mogu pratiti njegov razvoj, pridonijeti mu i eventualno dovesti ga na druge platforme [4].

Slika 2.2.1. *Službeni logo jezika Swift* [14]

Apple omogućava razvijačima software-a korištenje postojećih razvojnih okvira (eng. Frameworks-a) koji nude podlogu tj. osnovne funkcionalnosti jezika na koje razvijač dodaje svoj kôd. Razvojni okvir koji će se koristiti u ovom završnom radu zove se “Foundation”, a definira (neovisno o platformi) osnovne operacije i rad sa brojevima, podacima, tekstom, datumima, njihovo filtriranje, sortiranje i slično [4].

Verzija Swift-a 4.2 korištena u ovom završnom radu službeno je objavljena u rujnu 2018. godine.

2.3. Kratki uvod programiranja u Swift-u

2.3.1. Varijable i konstante

Varijabla je ime koje se pridodaje jednom dijelu računalne memorije u kojeg je spremljena informacija (vrijednost) koja se kasnije planira koristiti [5]. Stvaranje varijabli naziva se deklariranje varijabli, a u Swift-u se to radi s ključnom riječi `var`. Nakon ključne riječi dolazi ime varijable, zatim njen tip i naposljetku njena vrijednost. Konstante su varijable čije se vrijednosti nakon deklariranja ne mogu mijenjati. One se u kôdu označavaju sa ključnom riječi `let` [6].

```
var numberOfDays: Int = 30
let pi: CGFloat = 3.141
```

2.3.2. Tipovi (vrste) varijabli

Pri deklaraciji varijable potrebno je definirati vrstu (tip) informacije koju ona nosi. Tipovi varijabli koji će se spominjati u ovom završnom radu su:

Tablica 2.3.1: *Popis sastavnih klasa klase Line*

Ključna riječ	Tip	Primjer deklaracije varijable
`Int`	Cjelobrojni broj	`var age: Int = 21`
`CGFloat`	Decimalni broj	`var height: CGFloat = 178.5`
`String`	Tekst (niz znakova)	`var name: String = "Ante"`
`Bool`	true (da) ili false (ne)	`var isOpen: Bool = true`
`Int`	Cjelobrojni broj	`var age: Int = 21`

2.3.3. Metode (funkcije)

Metoda je samostalni dio kôda koji obavlja neku radnju. Često se nazivaju i funkcijama. Svaka metoda ima svoje ime koje opisuje njezin zadatak koje se 'zove', daju joj se potrebni ulazni parametri i očekuje se izlazna vrijednost (rezultat) [6].

Deklaracija metode počinje se ključnom riječi `func`, nastavlja imenom metode, te se u zagradi definiraju vrste njenih ulaznih parametara (ako postoje). Također se može definirati i izlazna vrijednosti tako da se nakon zagrade stavi strelica (->) a zatim tip izlazne vrijednosti [6]. Primjer metode:

```
func ageInMonths(years: Int, months: Int) -> Int {
    let totalMonths: Int = years*12 + months
    return totalMonths
}
```

Primjer pozivanja metode:

```
let myAgeInMonths = ageInMonths(years: 24, months: 3)
print(myAgeInMonths)
```

2.3.4. Klase

Klasa (eng. Class) je sastavni dio kôda pisanog u objektno orijentiranom programskom jeziku. Ona je "nacrt" prema kojemu se stvaraju (*instanciraju*) objekti. U Swift-u klasa može izgledati ovako:

```
class Person {

    var name: String
    var age: Int

    init(name: String, age: Int){
        self.name = name
        self.age = age
    }

    func introduce(){
        print("Hi, I'm \(name) and I am \(age)yrs old!")
    }
}
```

U tijelu klase (unutar vitičastih zagrada) definirane su varijable članice (članovi) te metode**.** Vidi se da će objekti klase imati dvije varijable - `name` i `age`, te jednu metodu - `introduce()`. Također mora se definirati specijalna metoda `init()` koja za ulazne parametre ima početne vrijednosti varijabli članica. Tom metodom stvaraju se objekti klase Person.

2.3.5. Objekti

Nakon što je definirana klasa 'Person', moguće je stvoriti jedan objekt takve klase. Stvaranje objekta određene klase naziva se "instanciranje", a za stvorene objekte se kaže da su instance te klase [6]. U Swiftu se instanciranje obavlja metodom `init` kojoj se prilažu početne vrijednosti svih članova:

```
let person1 = Person.init(name: "Ivan", age: 24) // stvaranje objekta metodom init
let person2 = Person(name: "Ana", age: 25) // skraćeni oblik pozivanja metode init
```

Da bi se pristupilo njihovim varijablama članicama ili metodama, koristi se operator točka kao što je prikazano u tablici 2.3.2 [6]:

Tablica 2.3.2: *Popis sastavnih klasa klase Line*

Kod	Izlaz (ispis na ekranu)
`print(person1.age)`	`24`
`person1.introduce()`	`Hi, i'm Ivan and I am 24yrs old!`

3. Modeliranje dalekovoda u Swift-u

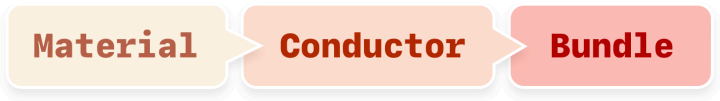
Da bi se definirala klasa koja će računati parametre dalekovoda potrebno je koncipirati klase od kojih će se ona sastojati. One će opisivati svojstva dalekovoda poput materijala od kojeg je izrađen vodič, presjeka vodiča, broja vodiča u snopu i njihovog rasporeda u prostoru.

Popis sastavnih klasa, njihovi kratki opisi, i datoteke u kojem će biti definirane ponuđen je u tablici 3.1:

***Tablica 3.0.1**: Popis sastavnih klasa klase Line*

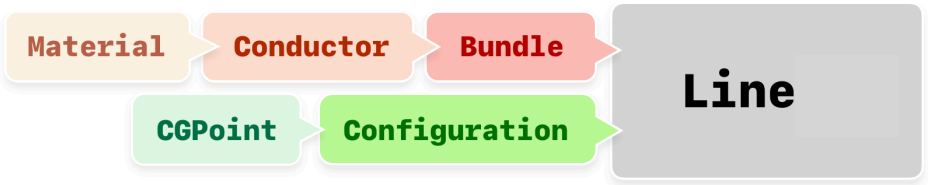
Klasa	Datoteka (sourcefile)	Opis
`Material`	Material.swift	Materijal, tvar od koje je izgrađen vodič
`Conductor`	`Conductor.swift`	Vodič određene površine presjeka i faktora popunjenosti
`Bundle`	`Bundle.swift`	Snop od jednog ili više vodiča.
`CGPoint`	`CGPoint.swift`	Točka u 2D prostoru (x i y koordinate)
`Configuration`	`Configuration.swift`	Prostorni raspored snopova trofaznog voda

U procesu definiranja klasa kreće se od najmanje i najjednostavnije klase - Material (materijal). Ona će među ostalim biti jedna od varijabli članica klase Conductor (vodič), koja će zauzvrat biti varijabla članica klase Bundle (snop vodiča). Ovu hijerarhiju klasa moguće je prikazati slikom 3.0.1:



***Slika 3.0.1:** Hijerarhija klasa Material - Conductor - Bundle* [20]

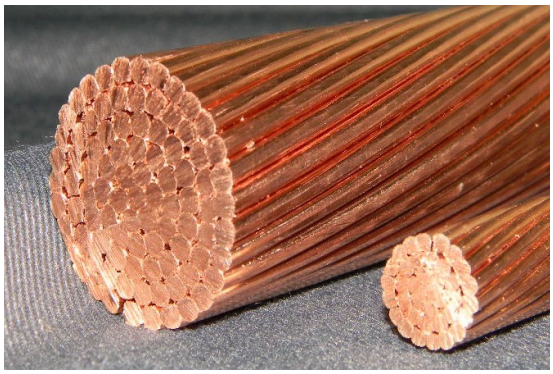
Krajnju, najsloženiju klasu koja će predstavljati trofazni vod nazvat će se `Line`. Ova klasa u svojim varijablama članicama sadržavati će klasu `Bundle`, klasu `Configuration`, frekvenciju sustava i temperaturu okoline. Zaključno s ovom klasom, slika 3.0.2 prikazuje sve klase koje će se javiti u izračunu:



***Slika 3.0.2:** Hijerarhija klase Line i njezinih sastavnih klasa* [20]

3.1. Materijal

Materijali imaju više specifičnih svojstava poput električne vodljivosti, otpornosti na koroziju, elastičnosti itd, a prema vrsti mogu se podijeliti na metale, polimere, keramičke i kompozitne materijale, poluvodiče i biomaterijale [7]. Ovaj rad usmjerit će se samo na metale bakar i aluminij.



Slika 3.1.1. *Bakreni vodič* [15]

Slika 3.1.2. *Aluminijski vodič* [16]

3.1.1. Klasa *Material*

U procesu definicije klasa dalekovoda kreće se od same jezgre, tj. materijala od kojeg je izgrađen vodič. Kao klasu koja opisuje materijal vodiča, definira se klasa `Material` sa dvije varijable članice: `specificResistance` (specifični električni otpor) i `tempResistanceCoeff` (temperaturni koeficijent električnog otpora):

```
class Material {

    var specificResistance: CGFloat
    var tempResistanceCoeff: CGFloat

    init(specificResistance: CGFloat, tempResistanceCoeff: CGFloat){
        self.specificResistance = specificResistance
        self.tempResistanceCoeff = tempResistanceCoeff
    }
}
```

Dakle, svaki objekt klase Material sadržavati te dvije varijable članice. U definiciju klase Material moguće bi bilo dodati još niz različitih varijabli članica kao što su boja, gustoća, čvrstoća i žilavost, no za svrhu ovog izračuna bit će dovoljna samo ona svojstva koja određuju električni otpor.

3.1.2. Specifični električni otpor i temperaturni koeficijent

Osnovno svojstvo materijala u elektrotehnici, specifični električni otpor (ρ) iskazuje kako dobro materijal provodi električnu struju, a temperaturni koeficijent električnog otpora iskazuje koliko se on povećava sa porastom temperature [8]:

$$\rho_t = \rho_0[1 + \alpha(t - t_o)] \quad [\Omega m] \tag{3-1-1}$$

ρ_0 - specifični električni otpor pri 20ºC

α - temperaturni koeficijent eletričnog otpora

t - temperatura

Potrebno je, dakle, definirati metodu `specificResistanceAt(temperature:)` unutar (vitičastih zagrada) klase Material koja koristeći ulazni parametar `temperature` računa i vraća vrijednost specifičnog električnog otpora:

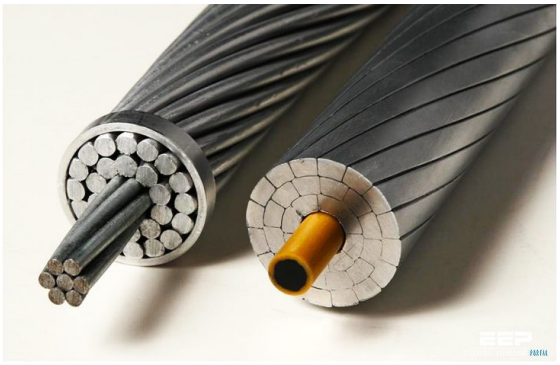
```
func specificResistanceAt(temperature: CGFloat) -> CGFloat {
   return specificResistance * (1 + tempResistanceCoeff*(temperature-20))
}
```

Primjer: u slučaju stvaranja objekta klase `Material` koji će opisivati aluminij, pri inicijalizaciji metodom `init()` treba se dati vrijednosti 2.65E-8 za spec. el. otpor i 3.60E-8 za temperaturni koeficijent:

```
let aluminum = Material(specificResistance: 2.65E-8, tempResistanceCoeff: 3.60E-3)
```

3.2. Vodič

U elektroenergetici i prijenosu električne energije pod ovim nazivom podrazumijevaju se užeta i žice koji provode električnu struju. [9]



Slika 3.2.1. *Vodiči u obliku užeta* [17]

Neki od materijala koji se koriste u izradi vodiča nadzemnog voda su bakar (i legure na bazi bakra), aluminij (i legure na bazi aluminija) i čelik [9]. Postoje različite izvedbe vodiča, npr. pune žice, vodiči u obliku užeta, snopovi, izolirani vodiči i specijalni vodiči [9]. Ipak, ovaj rad bazirat će se isključivo na vodiče izvedbe pune homogene žice i na homogene vodiče u obliku užeta.

3.2.1. Klasa *Conductor*

Za klasu koja opisuje vodič dalekovoda koristit će se klasa `Conductor`. Za svrhu ovog izračuna dovoljno je opisati svojstva vodiča sa samo tri varijable članice: materijalom (`material`, objekt klase `Material`), stvarnom površinom vodičom (`area`, decimalni broj) i faktorom K (`gmrAreaFactor`, decimalni broj).

```
class Conductor {

    var material: Material
    var area: CGFloat
    var gmrAreaFactor: CGFloat

    init(material: Material, area: CGFloat, gmrAreaFactor: CGFloat){
        self.material = material
        self.area = area
        self.gmrAreaFactor = gmrAreaFactor
    }

}
```

3.2.2. Reducirani radijus r'

Pri izračunu induktivnosti i kapacitivnosti javlja se izvedena veličina reducirani radijus [13]. Označava se sa r'. Za homogenu punu žicu vrijedi da je [10]:

$$r' = r \cdot e^{-\frac{1}{4}} = 0.7788r \tag{3-2-1}$$

U različitim izvedbama vodiča (cijev, uže) javljaju se i različite vrijednosti faktora reduciranog radijusa [13]. Zbog ovog uvodi se faktor K koji veže reducirani radijus sa stvarnom površinom vodiča [13]:

$$r' = K\sqrt{A} \tag{3-2-2}$$

K - faktor
A - stvarna površina vodiča

Ako se po (3-2-2) izrazi faktor K i uvrsti r' za slučaj homogene pune žice (3-2-1), vrijedi:

$$K = \frac{r'}{\sqrt{A}} = \frac{0.7788r}{\sqrt{r^2\pi}} = 0.43939 \tag{3-2-3}$$

Dakle, za stvaranje objekta tipa `Conductor` koji će definirati homogenu punu žicu presjeka 10mm², pri inicijalizaciji treba se dati željeni materijal (objekt klase Material), stvarnu površinu, te faktor K koji za slučaj punog okruglog vodiča iznosi 0.43939:

```
let alu = Material(specificResistance: 2.65E-8, tempResistanceCoeff: 3.60E-3)
let solidConductor = Conductor(material: alu, area: 10E-6, gmrAreaFactor: 0.43939)
```

3.2.3. Faktor *skin-effect*-a

Faktor skin-effect-a iskazuje koliko puta će otpor pri izmjeničnoj struji biti veći od onoga pri istosmjernoj [10]. Najšire upotrebljavana formula za aproksimaciju otpora pri izmjeničnoj struji može se pronaći u IEC 60287-1-1 [11]. Ova formula koristi istosmjerni otpor R_{dc} kao polazište pri računanju faktora *skin effect*-a [11]:

$$R_{ac} = R_{dc}(1 + y_s) \quad [\Omega/m] \quad \Longrightarrow \quad f_{\text{skin effect}} = \frac{R_{ac}}{R_{dc}} = 1 + y_s \tag{3-2-4}$$

R_{ac} - električna otpornost pri izmjeničnoj struji

R_{dc} - električna otpornost pri istosmjernoj struji

$$y_s = \frac{x_s^4}{192 + 0.8x_s^4} \tag{3-2-5}$$

$$x_s^2 = \frac{8\pi f K_s}{R_{dc}10^7} \tag{3-2-6}$$

K_s - tubularni faktor koji iznosi 1 za slučaj punog vodiča

Budući da klasa Conductor sadrži informaciju i o materijalu i veličini presjeka, može se definirati sljedeća metoda unutar njezinog tijela (prije zatvaranja vitičaste zagrade):

```
func skinEffectFactor(f: CGFloat, t1: CGFloat) -> CGFloat {
    let rdc = material.specificResistanceAt(temperature: t1) / area
    if rdc == 0 { return 1.0 }
    let tubularFactor: CGFloat = 1.0
    let xs = sqrt(3-14*8*f*tubularFactor/(rdc*1E+07))
    let ys = pow(xs, 4)/(192+0.8*pow(xs,4))
    return 1 + ys
}
```

Ova metoda za ulazne parametre uzima frekvenciju struje i temperaturu vodiča, a za rezultat vraća faktor skin effect-a.

Unutar ove metode koristi se varijabla članica `material`, to jest njezina metoda za izračun specifičnog električnog otpora pri nekoj temperaturi. Rezultat ove metode dijeli se sa površinom i tako se dobije R_{dc} za jedan metar vodiča te se primjenom izraza 3-2-6, 3-2-5 i 3-2-4 računa faktor y_s. Radi jednostavnosti koristi se tubularni faktor iznosa 1.

3.2.4. Specifični električni otpor pri određenoj frekvenciji i temperaturi

Nakon definiranja metode `skinEffectFactor(f:t1:)` moguće je definirati metodu koja računa otpornost vodiča po jedinici duljine pri određenoj frekvenciji i temperaturi:

```
func resistance(f: CGFloat, t1: CGFloat) -> CGFloat {
    let resistanceAtT1 = material.specificResistanceAt(temperature: t1) / area
    return resistanceAtT1 * skinEffectFactor(f: f, t1: t1)
}
```

3.2.5. Radijus i srednji geometrijski radijus

Zbog praktičnosti, definirat će se još dvije metode — jedna koja vraća radijus vodiča dobiven preko stvarne površine (uz pretpostavku da se radi o punom cilindričnom vodiču, izraz 3-2-7), i druga koja vraća srednji geometrijski radijus (eng. geometric mean radius) dobiven preko već spomenutog faktora K i površine vodiča (izraz 3-2-2).

$$r = \sqrt{\frac{A}{\pi}} \tag{3-2-7}$$

A - površina presjeka vodiča

```
func radius() -> CGFloat {
    return sqrt(area / 3.14)
}

func geometricMeanRadius() -> CGFloat {
    return gmrAreaFactor*sqrt(area)
}
```

3.2.6. Instanciranje klase *Conductor* polumjerom

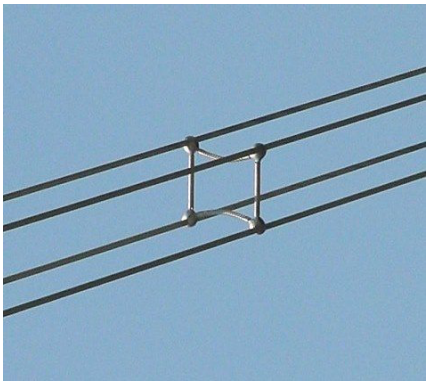
Može se definirati i druga `init` metoda za inicijalizaciju objekta klase Conductor pomoću polumjera:

```
init(material: Material, radius: CGFloat, reducedRadiusFactor: CGFloat)  {
    self.material = material
    self.area = radius * radius * .pi
    self.gmrAreaFactor = reducedRadiusFactor * radius / sqrt(area)
}
```

Ovdje umjesto ulaznog parametra površine i faktora K daje se polumjer i faktor reduciranog radijusa. U tijelu metode zatim se računa površina A po izrazu 3-2-7 i faktor K po izrazu 3-2-2 te se spremaju u varijable članice.

3.3. Grupa vodiča

Za prijenos snage preko većih udaljenosti koristi se i veći napon, a pri većim naponima javlja se efekt korone koji uzrokuje znatan gubitak snage [10]. Da bi se ovaj efekt smanjio često se koristi više od jednog vodiča po fazi — snop vodiča (eng. bundle) [10].



Slika 3.3.1. *Snop od 4 vodiča sa odvojnikom* [18]

3.3.1. Klasa Bundle

Za opis snopa vodiča u izračunu definirat će se klasa `Bundle` (hrv. snop) koja će imati tri varijable članice: broj vodiča, razmak između njih te svojstva vodiča (sadržana u objektu klase `Conductor`). Pri snopu se pretpostavlja geometrija jednakostraničnog poligona.

```
class Bundle {

    var conductor: Conductor
    var count: Int
    var spacing: CGFloat

    init(conductor: Conductor, count: Int, spacing: CGFloat){
        self.conductor = conductor
        self.count = count
        self.spacing = spacing
    }

    func resistance(f: CGFloat, t1: CGFloat) -> CGFloat {
        return conductor.resistance(f: f, t1: t1) / CGFloat(count)
    }
}
```

Kao i u prethodnim klasama Conductor i Material, u klasi Bundle definirana je metoda `resistance(f:t1:)` koja uz danu frekvenciju i temperaturu vraća električni otpor po jedinici duljine cijeloga snopa. On se računa po izrazu:

$$\rho_b = \frac{\rho_c}{n} \tag{3-3-1}$$

ρ_c - specifični električni otpor jednog vodiča

n - broj vodiča u snopu

3.3.2. Srednji geometrijski radijus za induktivnost

U računu induktiviteta snopa vodiča potreban je njegov srednji geometrijski radijus [13]. Budući da klasa Bundle podrazumjeva snop vodiča prostorno raspoređen u jednakostranični poligon, njegov srednji geometrijski radijus (SGR) može se računati po sljedećoj formuli [13]:

$$SGR_{\text{snopa}} = \sqrt[n]{n \cdot SGR \cdot R^{n-1}} \tag{3-3-2}$$

n - broj vodiča u snopu

SGR - srednji geometrijski radijus vodiča

R - polumjer opisane kružnice poligona

U tijelo klase Bundle dodaje se metoda za izračun SGR-a snopa zvana 'bundleGeometricMeanRadius':

```
func bundleGeometricMeanRadius() -> CGFloat {
    let gmr = conductor.geometricMeanRadius()
    let innerRadius = spacing / sqrt(2 - (2 * cos(degrees: 360.0 / count)))
    return pow( n * gmr * pow(innerRadius, count-1) , 1/count)
}
```

3.3.3. Srednji geometrijski radijus za kapacitivnost

U računu dozemnog kapaciteta snopa vodiča koristi se izraz vrlo sličan gore navedenom izrazu sa jednom razlikom — umjesto srednjeg geometrijskog vodiča pod korijenom se nalazi stvarni polumjer vodiča. Razlog tome je da se naboj nalazi na samoj površini vodiča polumjera r [13].

$$SGR_{\text{snopa}} = \sqrt[n]{n \cdot r \cdot R^{n-1}} \tag{3-3-3}$$

n - broj vodiča u snopu

r - stvarni radijus vodiča

R - polumjer opisane kružnice poligona

U ovom slučaju metoda klase Bundle za izračun SGR-a snopa izgleda ovako:

```
func bundleCapacitanceGeometricMeanRadius() -> CGFloat {
    let r = conductor.radius()
    let innerRadius = spacing / sqrt(2 - (2 * cos(degrees: 360.0 / count)))
    return pow( n * r * pow(innerRadius, count-1) , 1/count)
}
```

3.4. Raspored vodiča u prostoru

Prostorni raspored snopova vodiča u prostoru je bitan u modeliranju dalekovoda jer o njemu ovise primarni parametri induktivnost i kapacitivnost [10]. On je određen vrstom korištenih dalekovodnih stupova.

Slika 3.4.1. *Stupovi dalekovoda* [19]

Postoje različiti tipovi stupova, a njihov izbor ovisi o vrijednostima napona te broju krugova kojih može biti jedan (jednostruki vod) ili više (dvostruki vod) [12]. Neki od najčešće korištenih vrsta dalekovodnih stupova za jednostruki vod su: portal, jela, modificirana jela i Y stup. Za dvostruki često se koriste oblici: bačva, dvostruka jela i dunav [10].

U svrhu pojednostavljenja izračuna modelirat će se isključivo stupovi korišteni u jednostrukom vodu, konkretno, tipovi Y-stup i jela prikazani na slici 3.4.1.

3.4.1. Klasa *CGPoint*

Za opisivanje točaka u dvodimenzionalnom sustavu koristit će se jednostavna klasa CGPoint. Ona sadrži dvije varijable članice: x i y koordinatu.

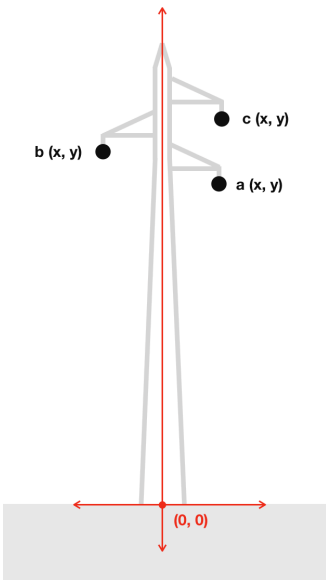
```
class CGPoint {
    var x: CGFloat
    var y: CGFloat

    init(x: CGFloat, y: CGFloat){
        self.x = x
        self.y = y
    }
}
```

3.4.2. Klasa *Configuration*

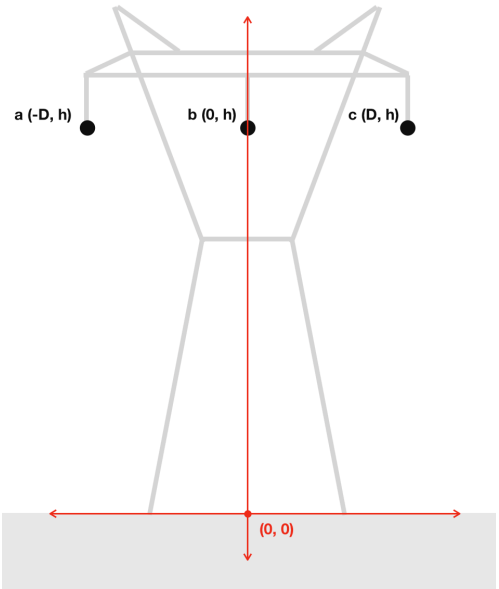
Prostorni raspored snopova vodiča opisat će se klasom `Configuration` koje će sadržavati koordinate sva tri snopa označene slovima a, b i c, te provjes (eng. sag).

```
class Configuration {

    var a: CGPoint
    var b: CGPoint
    var c: CGPoint
    var sag: CGFloat

    init(a: CGPoint, b: CGPoint, c: CGPoint, sag: CGFloat){
        self.a = a
        self.b = b
        self.c = c
        self.sag = sag
    }

    init(distance: CGFloat, height: CGFloat, sag: CGFloat) {
        self.a = CGPoint(x: -distance, y: height)
        self.b = CGPoint(x: 0, y: height)
        self.c = CGPoint(x: distance, y: height)
        self.sag = sag
    }
}
```

Metoda `init(a:b:c:sag)` koristi se pri inicijalizaciji prostornog rasporeda dalekovoda nošenog stupovima "jela" (slika 3.4.2), a metoda `init(distance:height:sag:)` definirana je radi lakše inicijalizacije prostornog rasporeda dalekovoda nošenog stupovima "Y-stup" (slika 3.4.3) sa ulaznim parametrima `distance` (razmak D između vodiča) i `height` (visina sva tri vodiča od zemlje).



***Shema 3.4.2**: Jela* [20]



***Shema 3.4.3**: Y-stup* [20]

3.4.3. Srednja geometrijska udaljenost snopova vodiča

Definirat će se metoda `geometricMeanDistance()` koja vraća međusobnu geometrijsku udaljenost sva tri snopa vodiča:

```
func geometricMeanDistance() -> CGFloat {
    let ab = distance(p1: a, p2: b)
    let bc = distance(p1: b, p2: c)
    let ac = distance(p1: a, p2: c)
    return pow(ab*bc*ac, 0.333)
}
```

Udaljenost točaka dobije se metodom `distance(p1,p2)`, a njihova srednja geometrijska sredina je treći korijen njihovog umnoška. On se računa sa `pow(x,y)` - ugrađenom funkcijom koja vraća decimalni broj x podignut na eksponent y.

3.4.4. Srednja geometrijska visina snopova vodiča

Računanje srednje visine nekog vodiča vrši se po sljedećem izrazu [13]:

$$h_x = H_x - 0.7f \tag{3-4-1}$$

H_x - visina vodiča x

f - provjes

Definirat će se metoda geometricMeanHeight() koja vraća geometrijsku sredinu sve tri srednje visine snopa vodiča:

```
func geometricMeanHeight() -> CGFloat {
    let ha = a.y - 0.7*sag
    let hb = b.y - 0.7*sag
    let hc = c.y - 0.7*sag
    return pow(ha*hb*hc, 0.333)
}
```

3.5. Dalekovod

Svaki dalekovod pokazuje električna svojstva otpornosti, induktivnosti, kapacitivnosti i odvoda. Ovi parametri nazivaju se primarnim konstantama voda [13], a navedeni su u tablici 3.5.1.

Tablica 3.5.1: Primarni parametri voda

Naziv	Oznaka	Jedinica	Topli/Hladni
Jedinični otpor	R_1	[Ω / km]	Topli
Jedinični induktivitet	L_1	[H / km]	Hladni
Jedinični odvod	G_1	[S / km]	Topli
Jedinični kapacitet	C_1	[F / km]	Hladni

Izražene su po jedinici duljine (km), a njihovi iznosi ovise o materijalu vodiča, vanjskim uvjetima te o prostornom rasporedu snopova vodiča. Dijele se na tople i hladne ovisno o tome izazivaju li toplinske gubitke prolaskom struje [10].

3.5.1. Klasa *Line*

Napokon je moguće definirati i klasu `Line` koja će u izračunu predstavljati dalekovod. Ona će sadržavati informaciju o sastavu snopa vodiča (objekt klase Bundle), njihovom prostornom rasporedu (objekt klase Configuration), duljini voda, frekvenciji sistema i temperaturi. Ovaj izračun podrazumjeva trofazni dalekovod koji sadržava tri snopa vodiča jednakih svojstava.

```
class Line {

    var bundle: Bundle
    var configuration: Configuration
    var temperature: CGFloat = 24.0
    var frequency: CGFloat = 50.0
    var length: CGFloat = 500.0

    init(configuration: Line.Configuration, bundle: Bundle, temperature: CGFloat,
         frequency: CGFloat, length: CGFloat){
        self.configuration = configuration
        self.bundle = bundle
        self.temperature = temperature
        self.frequency = frequency
        self.length = length
    }
}
```

3.5.2. Jedinični otpor

Jedinični otpor ovisi o fizičkom sastavu (materijalu) vodiča pri određenoj temperaturi i frekvenciji (skin efekt) [11]. Kao i u prethodnim klasama, i u klasi Line definirat će se metoda koja računa specifični električni otpor, no ovaj put po jedinici duljine u kilometrima što odgovara jediničnom otporu voda:

```
func resistancePerKM() -> CGFloat {
    return bundle.resistance(f: frequency, t1: temperature) * 1000
}
```

Varijable članice `frequency` i `temperature` koriste se kao ulazni parametri za metodu treće varijable članice - `bundle`. Radi se o njezinoj metodi `resistance(f:t1:)` koja vraća specifični el. otpor snopa. Preostaje pomnožiti dobivenu vrijednost sa 1000 za konverziju iz H/m u H/km.

3.5.3. Jedinični odvod

Jedinični odvod uzrokovan je strujom koja teče preko izolatora i zraka [10]. Većina modernih dalekovoda ima zanemariv jedinični odvod pa se on ne uzima u obzir pri analizi voda. Ipak, u svrhe demonstracije uvrstit će se iznos jediničnog odvoda od $5 \cdot 10^{-8}$ S/km. Metoda koja vraća jedinični odvod zvat će se conductancePerKM:

```
func conductancePerKM() -> CGFloat {
    return 5E-8
}
```

3.5.4. Jedinični induktivitet

Jedinični induktivitet i kapacitet uzrokovani su prisustvom magnetskog i električnog polja oko vodiča. Njihov iznos ovisi i o prostornom rasporedu samih vodiča [10].

Za danu međusobnu geometrijsku udaljenost snopova i srednji geometrijski radijus snopa, jedinična induktivnost voda računa se po sljedećem izrazu [13]:

$$L_1 = 2 \cdot 10^{-4} \cdot \ln \frac{\text{MSGU}}{\text{SGR}} \quad [\text{H / km}] \tag{3-5-1}$$

MSGU *(eng. GMD)* - međusobna srednja udaljenost snopova vodiča
SGR *(eng. GMR)* - srednji geometrijski radijus snopa vodiča

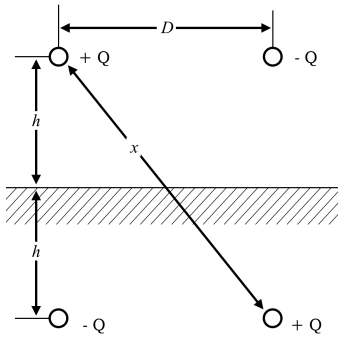
Definirat će se `inductancePerKM` - metoda klase Line koja računa jediničnu induktivnost voda:

```
func inductancePerKM() -> CGFloat {
    let gmd = configuration.geometricMeanDistance()
    let gmr = bundle.bundleGeometricMeanRadius()
    return 2E-4*log(gmd/gmr)
}
```

U ovoj metodi za MSGU snopova koristi se varijabla članica `configuration`, tj. njezina metoda `geometricMeanDistance()`, a za SGR samog snopa varijabla članica `bundle`, tj njezina metodua `bundleGeometricMeanRadius()`. Za prirodni logaritam u jeziku Swift koristi se metoda `log(value:)`.

3.5.5. Jedinični kapacitet

Za proračun jediničnog (pogonskog) kapaciteta koristi se metoda srednjih geometrijskih udaljenosti (SGU) kojom se trofazni nesimetrični prepleteni vod dijeli na dvije grupe vodiča: jednu sa pozitivnim (+Q) i jednu sa negativnim (-Q) nabojem [13]. To se može prikazati sljedećim modelom:



***Slika 3.5.2**: SGU model* [13]

Gdje je:

h - geometrijska sredina visina snopova vodiča

D - srednja udaljenost snopova vodiča

x - međusobna udaljenost dvaju pozitivnih naboja

$$\text{MSGU} = \sqrt[2 \cdot 2]{D \cdot 2h \cdot D \cdot 2h} = \sqrt{D \cdot 2h} \tag{3-5-2}$$

$$\text{SGR*} = \sqrt[2^2]{r \cdot x \cdot r \cdot x} = \sqrt{r \cdot x} \tag{3-5-3}$$

Analogno SGU metodi za proračun induktiviteta, jedinični kapacitet skupine pozitivno nabijenih vodiča i jedinični kapacitet po jednom vodiču računaju se izrazima [13]:

$$C_A = \frac{1}{18 \cdot 10^6 \cdot \ln \frac{\text{MSGU}}{\text{SGR*}}} \tag{3-5-4}$$

$$C_1 = \frac{C_A}{2} = \frac{1}{36 \cdot 10^6 \cdot \ln \frac{\text{MSGU}}{\text{SGR*}}} \tag{3-5-5}$$

Dakle, u tijelo klase `Line` dodaje se:

```
func capacitancePerKM() -> CGFloat {
    let gmd = configuration.geometricMeanDistance()
    let h = configuration.geometricMeanHeight()
    let x = sqrt(D1*D1 + 4*h*h)
    let gmr = bundle.capacitanceGeometricMeanRadius()

    let Dm = sqrt(2 * h * gmd)
    let Ds = sqrt(gmr * x)
    return 1 / (36E6 * log (Dm/Ds))
}
```

U ovoj metodi koristi se varijabla članica `configuration`, tj. njene metode za MSGU (`gmd`) i srednju geometrijsku visinu snopova (`h`) - koje su potrebne za metodu SGU. Zatim se računaju MSGU (`Dm`) i SGR* (`Ds`) modela SGU po izrazima 3-5-2 i 3-5-3, te pogonski kapacitet (C_1) po izrazu 3-5-5.

3.5.6. Jedinična reaktancija i susceptancija

U proračunima izmjenične struje hladni primarni parametri javljaju se u frekvencijski ovisnom obliku koji se nazivaju jedinična reaktancija (X_1) i jedinična susceptancija (B_1). Računaju se po izrazima [13]:

$$X_1 = \omega L_1 = 2\pi f \cdot L_1 \quad [\Omega / \text{km}] \tag{3-5-6}$$

$$B_1 = \omega C_1 = 2\pi f \cdot C_1 \quad [\text{H} / \text{km}] \tag{3-5-7}$$

Oni će se dodati u tijelo klase `Line` sa sljedećim metodama:

```
func reactancePerKM() -> CGFloat {
    return inductancePerKM * 2 * .pi * frequency
}

func susceptancePerKM() -> CGFloat {
    return capacitancePerKM * 2 * .pi * frequency
}
```

3.5.7. Jedinična impedancija i admitancija

Jedinična impedancija $\overline{Z}_1$ i admitancija $\overline{Y}_1$ složene su kompleksne vrijednosti. Računaju se po sljedećim izrazima [13]:

$$\overline{Z}_1 = R_1 + jX_1 \tag{3-5-8}$$

$$\overline{Y}_1 = G_1 + jB_1 \tag{3-5-9}$$

3.5.8. Valna konstanta i karakteristična impedancija

Valna konstanta je sekundarna konstanta voda i još se naziva i konstantom prodiranja. Označava se sa $\overline{\gamma}$ a računa se pomoću kompleksnih vrijednosti jedinične impedancije i jedinične admitancije [13]:

$$\overline{\gamma} = \sqrt{\overline{Z}_1 \cdot \overline{Y}_1} \quad [\text{km}^{-1}] \tag{3-5-10}$$

Karakteristična impedancija je još jedna sekundarna konstanta voda. Označava se sa $\overline{Z}_c$ i također se računa pomoću jedinične impedancije i jedinične admitancije [13]:

$$\overline{Z}_c = \sqrt{\frac{\overline{Y}_1}{\overline{Z}_1}} \quad [\Omega / \text{km}] \tag{3-5-11}$$

3.6. Teorija prijenosa

3.6.1. Matematički modeli dalekovoda

Za dalekovod razvijena su tri matematička modela klasificirana po dužini:

- Dugački - duži od 250 km
- Srednji - između 80 i 250 km
- Kratki - kraći od 80 km

U ovom izračunu koristit će se isključivo dugački model koji je ujedno i najprecizniji.

3.6.2. Prijenosne jednadžbe

U kôdu izračuna koristit će se jednadžbe za slučaj kad su poznati struja i napon na početku voda a treba se izračunati struju i napon u nekoj točki na duljini x. Izvedene su iz telegrafskih jednadžbi, a glase [13]:

$$\overline{V} = \frac{1}{2}(\overline{V}_1 + \overline{Z}_c \cdot \bar{I}_1) \cdot e^{-\overline{\gamma} \cdot x} + \frac{1}{2}(\overline{V}_1 - \overline{Z}_c \cdot \bar{I}_1) \cdot e^{\overline{\gamma} \cdot x} \tag{3-6-1}$$

$$\bar{I} = \frac{1}{2}(\bar{I}_1 + \frac{\overline{V}_1}{\overline{Z}_c}) \cdot e^{-\overline{\gamma} \cdot x} + \frac{1}{2}(\bar{I}_1 - \frac{\overline{V}_1}{\overline{Z}_c}) \cdot e^{\overline{\gamma} \cdot x} \tag{3-6-2}$$

Gdje je:

$\bar{I}_1$ - struja na početku voda

$\overline{V}_1$ - napon na početku voda

x - udaljenost od početka voda

Ako se za x uvrsti krajnja vrijednost - duljinu voda l i ako se gore navedene jednadžbe izraze u njihovom hiperbolnom obliku, tada će izraz za napon na kraju voda glasiti [13]:

$$\overline{V}_2 = \overline{V}_1 \cdot ch(\overline{\gamma} l) - \overline{Z}_c \cdot \bar{I}_1 \cdot sh(\overline{\gamma} l) \tag{3-6-3}$$

$$\bar{I}_2 = \bar{I}_1 \cdot ch(\overline{\gamma} x) - \frac{\overline{V}_1}{\overline{Z}_c} \cdot sh(\overline{\gamma} x) \tag{3-6-4}$$

3.6.3. Klasa *CXCGFloat*

Budući da se u računu javljaju kompleksne vrijednosti poput jedinične impedancije i admitancije, te sekundarnih konstanti voda, mora se koristit kompleksna klasa `CXCGFloat` koja predstavlja kompleksni decimalni broj sa svojom realnom i imaginarnom vrijednosti.

Zbog pojednostavljenja, neće se ulaziti u strukturu klase `CXCGFloat` nego će se samo pregledati njezine varijable članice i metode:

Tablica 3.6.1: Metode klase CXCGFloat

Metode		
Ključna riječ	**Tip rezultata**	**Rezultat**
`angle()`	`CGFloat`	kut (º) ili argument polarnog oblika
`radius()`	`CGFloat`	Amplituda (radijus) polarnog oblika
`conj()`	`CXCGFloat`	Konjugirana vrijednost

Tablica 3.6.2: Varijable članice klase CXCGFloat

Varijable članice		
Ključna riječ	**Tip**	**Opis**
`real`	`CGFloat`	realna vrijednost
`imag`	`CGFloat`	imaginarna vrijednost

3.6.4. Klasa VCPoint

Kako bi se lakše baratalo sa strujno-naponskim prilikama (strujom, naponom, snagom), potrebno je definirati klasu VCPoint koja opisuje strujno-naponske prilike u jednoj točki na vodu. Ona će sadržavati dvije kompleksne vrijednosti, jednu za vrijednost napona a drugu za vrijednost struje:

```
class VCPoint {

    var voltage: CXCGFloat
    var current: CXCGFloat

    init(voltage: CXCGFloat, current: CXCGFloat){
        self.voltage = voltage
        self.current = current
    }

}
```

3.6.5. Računanje struje i napona na kraju voda

Koristeći ranije navedene prijenosne jednadžbe u hiperbolnom obliku za računanje iznosa struje i napona na kraju voda (3-6-3 i 3-6-4), sada je moguće u tijelo klase Line dodati metodu `endVoltageAndCurrent` (hrv. "krajnji napon i struja"):

```
func endVoltageAndCurrent(vcPoint1: VCPoint) -> VCPoint {
    var v1: CXCGFloat = vcPoint1.voltage
    var i1: CXCGFloat = vcPoint1.current

    var Z1 = CXCGFloat(real: resistancePerKM(), imag: reactancePerKM())
    var Y1 = CXCGFloat(real: conductancePerKM(), imag: susceptancePerKM())
    var  y: CXCGFloat = sqrt(Z1*Y1)
    var Zc: CXCGFloat = sqrt(Z1/Y1)

    var v2: CXCGFloat = v1 * cosh(y*length) - i1 * Zc * sinh(y*length)
    var i2: CXCGFloat = i1 * cosh(y*length) - v1 / Zc * sinh(y*length)
    return VCPoint(voltage: v2, current: i2)
}
```

Ulazni parametar `vcPoint1` sadržava kompleksne iznose struje i napona na početku voda, a krajnji rezultat metode je objekt istog tipa (`VCPoint`) koji sadržava iznose struje i napona na kraju voda.

Radi preglednosti, na samom početku metode, vrijednosti varijabli članica `voltage` i `current` od objekta `vcPoint1` spremaju se u varijable `v1` i `i1`.

Popis varijabli koje se dalje javljaju u ovoj metodi, njihove oznake i opisi, te izrazi po kojem se računaju mogu se vidjeti u tablici 3.6.3:

Tablica 3.6.3: Varijable u metodi endVoltageAndCurrent()

Ime varijable	Oznaka i opis		Izraz po kojem se računa
`Z1`	Z_1	Jedinična impedancija	(3-5-8)
`Y1`	Y_1	Jedinična admitancija	(3-5-9)
`y`	$\overline{\gamma}$	Valna konstanta	(3-5-10)
`Zc`	Z_c	Karakteristična impedancija	(3-5-11)

Pri kraju metode računaju se vrijednosti napona i struje na kraju voda (`v2` i `i2`) po izrazima (3-6-3) i (3-6-4). Inicijalizira se novi objekt klase VCPoint (`vcPoint2`) koristeći te dvije vrijednosti. Naposlijetku novonastali objekt `vcPoint2` vraća se kao rezultat ove metode.

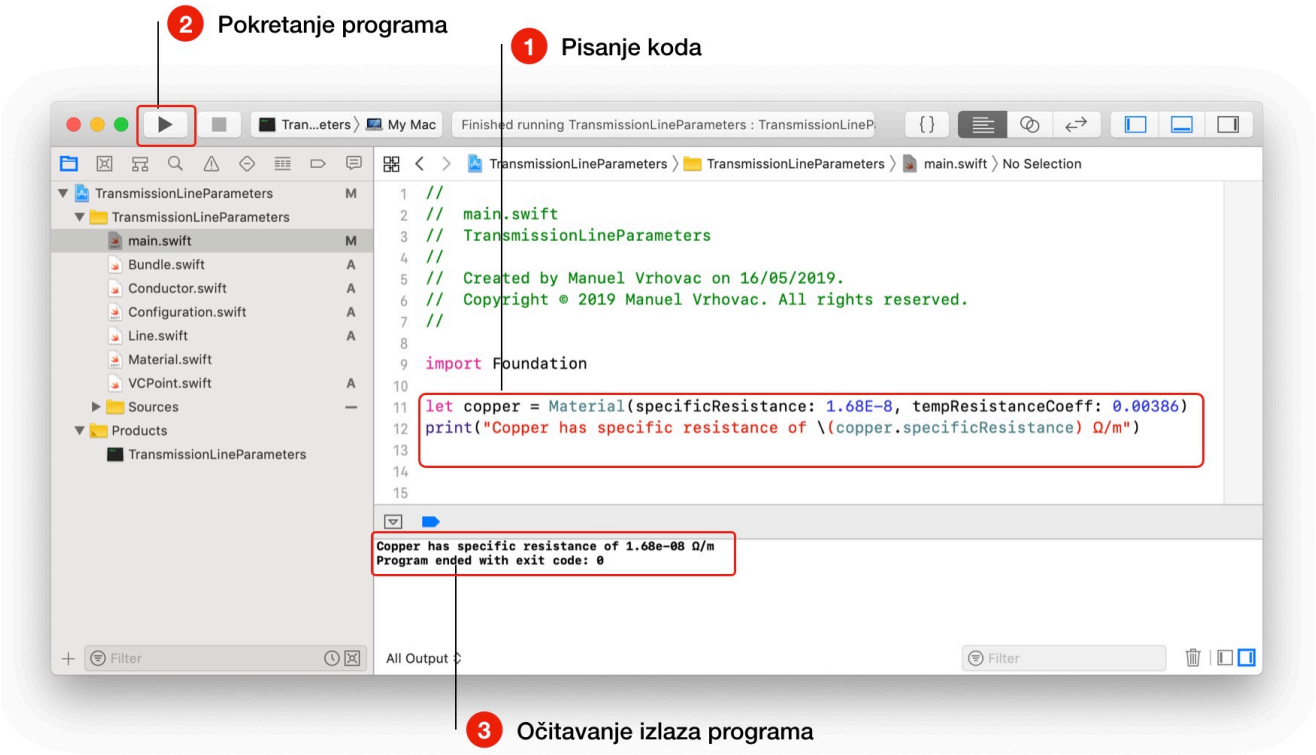
4. Ispitivanje funkcionalnosti

4.1. Testiranje metoda klase *Line*

Za testiranje metoda koje računaju primarne parametre voda ovaj rad će se osloniti na zadatke i rješenja iz zbirke "Elektroenergetske mreže - zbirka riješenih zadataka" [13].

Testiranje će se odvijati u razvojnom okružju XCode prateći tri jednostavna koraka (slika 4.1.1.):

1. Piše se testni kôd u datoteku zvanu "main.swift" slobodno koristeći klase definirane u drugim datotekama (*Material.swift*, *Conductor.swift*, *Line.swift*).
2. Odabire se radnja "Play". XCode će tada kompajlirati kôd svih definicija klasa (provjeriti ima li grešaka u pisanju kôda) i pokrenuti kôd napisan u datoteci "main.swift".
3. Očitava se ispisana vrijednost iz konzole (dno ekrana) i uspoređuje se sa rezultatima iz zbirke. Donosi se zaključak je li dobivena vrijednost točna, tj. funkcionira li klasa onako kako je osmišljena.



Slika 4.1.1: *Testiranje kôda aplikacije u XCode-u* [20]

4.1.1. Ispitivanje metode za jedinični kapacitet

Za ispitivanje metode capacitancePerKM u klasi Line modelirat će se sljedeći zadatak [13]:

> ***Str. 49, zad. 2.4.*** *Položaji ovjesišta vodiča trofaznog nesimetričnog prepletenog voda (s ishodištem koordinatnog sustava u podnožju stupa) su: faza a (-4m, 10m), faza b (4m, 15m), faza c(6m, 9m). Provjes je f = 1.45m. Primjenom SGU metode izračunati jedinični pogonski kapacitet jednog vodiča, ako je polumjer vodiča r = 12mm.* ***Rješenje****: 8.58·10⁻⁹ [F/km]*

1. Kod napisan datoteci "main.swift":
2. Izlaz u području konzole:

```
let aluminum = Material(specificResistance: 2.65E-8,
                        tempResistanceCoeff: 0.0036)
let solidRoundConductor = Conductor(material: aluminum,
                                    radius: 12E-3,
                                    reducedRadiusFactor: 0.7788)
let singleBundle = Bundle(conductor: solidRoundConductor,
                          count: 1,
                          spacing: 0.00)
let lineConfiguration = Configuration(a: CGPoint(x: -4.0, y: 10.0),
                                      b: CGPoint(x: 4.0, y: 15.0),
                                      c: CGPoint(x: 6.0, y: 9.0),
                                      sag: 2.14)
let line = Line(configuration: lineConfiguration,
                bundle: singleBundle,
                frequency: 50.0,
                length: 400.0)
print("C1: \(line.capacitancePerKM*1E9) nF/km")
```

3. Vrijednost 8.5959 nF/ km odgovara rješenju iz zbirke.

```
C1: 8.5957 nF/km
```

4.1.2. Ispitivanje metode za jedinični induktivitet

Za ispitivanje metode inductancePerKM u klasi Line modelirat će se sljedeći zadatak iz zbirke [13]:

> ***Str. 48, zad. 2.3.*** *Metodom SGU izračunati jedinični pogonski induktivitet prepletenog nesimetričnog trofaznog voda kojem se faze sastoje od četiri vodiča u snopu. Vodiči su stvarnog presjeka A = 586 mm², a izrađeni su od 61 žice (d_s = 0.502). Udaljenost faza je D = 12m, a udaljenost između vodiča u snopu a = 40cm.* ***Rješenje****: 0.88 mH / km*

1. Kod napisan datoteci "main.swift":
2. Izlaz u području konzole:

```
let aluminum = Material(specificResistance: 2.65E-8,
                        tempResistanceCoeff: 0.0036)
let solidRoundConductor = Conductor(material: aluminum,
                                    area: 586E-6,
                                    gmrAreaFactor: 0.502)
let quadBundle = Bundle(conductor: solidRoundConductor,
                        count: 4,
                        spacing: 0.40)
let lineConfiguration = Configuration(distance: 12,
                                      height: 20,
                                      sag: 0.0)
let line = Line(configuration: lineConfiguration,
                bundle: quadBundle,
                frequency: 50.0,
                length: 400.0)
print("L1: \(line.inductancePerKM()*1E3 ) mH/km")
```

3. Vrijednost 8.5959 nF/ km odgovaraju rješenju iz zbirke.

```
L1: 0.8832 mH/km
```

4.1.3. Ispitivanje metode za računanje strujno-naponskih prilika na kraju voda

Za ispitivanje metode endVoltageAndCurrentFor klase Line, modelirat će se sljedeći zadatak iz zbirke [13]:

> ***3.6.*** *Za trofazni dalekovod nazivnog napona 220kV, duljine L = 250km, zadani su sljedeći podaci:* $R_1 = 0.085$ */km,* $X_1 = 0.42$ */km,* $C_1 = 8.6 \cdot 10^{-9}$ *F/km,* $G_1 = 0.05 \cdot 10^{-6}$ *S/km. Pogonski napon na početku voda* $U_1 = 245$ *kV i snaga na početku voda* $S_1 = P_1 + jQ_1 = 160+j45$ *MVA (...) , (L1 = 1.3376 mH)*
> ***Rješenje:*** $V_2 = 116.67 - j36.51$ *[kV],* $P_2 = 148.02$ *[MW],* $Q_2 = 27.33$ *[MVAr]*

U svrhu izvedbe ovog testa dodatno smo definirali četiri opcionalne varijable članice (`R1, L1, G1, C1`) u tijelu klase `Line` čije se postojanje provjerava u svakoj odgovarajućoj funkciji koja inače računa otpornost, induktivnost, vodljivost i kapacitivnost po kilometru. Ukoliko ona postoji, funkcija izravno vraća već postavljenu (opcionalnu) vrijednost. Definirana je zatim i metoda `init(R1:L1:G1:C1)` koja ove četiri vrijednosti i duljinu uzima kao parametre te ih postavlja u novodefinirane opcionalne varijable:

```
class Line {

    var configuration: Configuration
    var bundle: Bundle

    var temperature: CGFloat = 24.0
    var frequency: CGFloat = 50.0
    var length: CGFloat = 500.0

    var R1: CGFloat?
    var L1: CGFloat?
    var G1: CGFloat?
    var C1: CGFloat?

    ...

    init(R1: CGFloat, L1: CGFloat, G1: CGFloat, C1: CGFloat, length: CGFloat){
        self.R1 = R1
        self.L1 = L1
        self.G1 = G1
        self.C1 = C1
        self.length = length
    }

    func resistancePerKM() -> CGFloat {
            if let R1 = self.R1 {
                return R1
            }
            ...
    }
    ...
}
```

Kako je zadan linijski napon, dijeli ga se sa korijenom iz tri da bi dobili fazni, a da bi dobili struju na početku voda, konjugirana vrijednost kompleksne snage dijeli sa 3 i sa konjugiranim faznim naponom. Ista operacija obavlja se unatrag da bi dobili kompleksnu snagu na kraju voda.

1. Kod napisan datoteci "main.swift":

```
let line = Line(R1: 0.085, L1: 0.42/314, G1: 0.05E-6, C1: 8.6E-9, length: 250)

let u1 = CXCGFloat(245E3)
let p1 = CXCGFloat(160E6, 45E6)
let v1 = u1 / sqrt(3)
let i1 = (p1.conj / (3 * v1.conj))

let vcPoint1 = VCPoint(voltage: v1, current: i1)
let vcPoint2 = line.endVoltageAndCurrentFor(vcPoint1: vcPoint1)
let p2 = 3 * vcPoint2.voltage * vcPoint2.current.conj

print("v2: \(vcPoint2.voltage*1E-3) [kV]")
print("p2: \(p2*1E-6) [MW]")
```

2. Izlaz u području konzole:

```
v2: (117.66-35.85.i) [kV]
p2: (149.02+29.35.i) [MW]
```

3. Dobivene kompleksne vrijednosti odgovaraju rješenjima iz zbirke (pogreška manja od 1%).

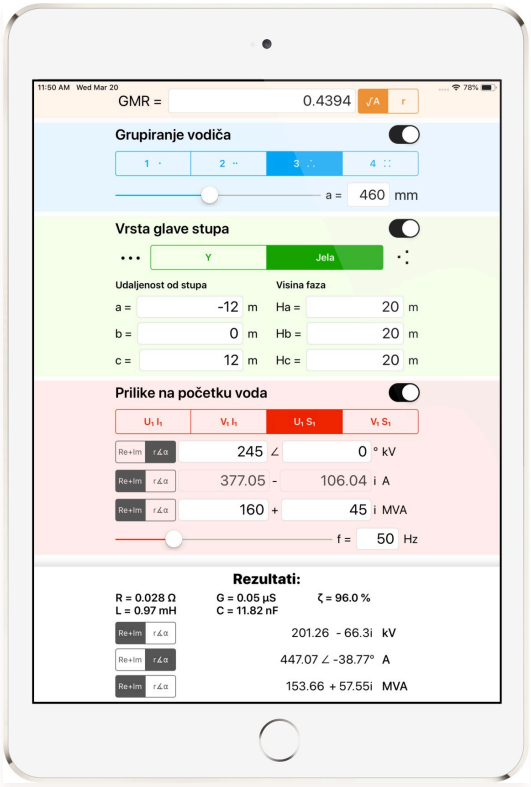
4.2. Testiranje sučelja iOS Aplikacije

U prošlom poglavlju dokazano je kako napisani kôd daje točne rezultate pri računanju primarnih parametara nadzemnog voda i računanju struje i napona na kraju voda uz zadane početne uvjete.

U svrhu demonstracije izrađeno je grafičko sučelje koje će omogućiti da se ovi parametri mijenjaju lako i brzo ali i da se rezultat odmah prikaže na način koji korisniku u tom trenutku najviše odgovara. Izrada grafičkog sučelja u razvojnom okružju XCode i povezivanje elemenata sučelja sa kôdom aplikacije vrlo je složen proces i nadilazi opseg ovog završnog rada. Iz tog razloga neće se ulaziti u sam proces izrade sučelja nego će se objasniti elementi kontrole (upravljanja) parametara i način prikaza rezultata.

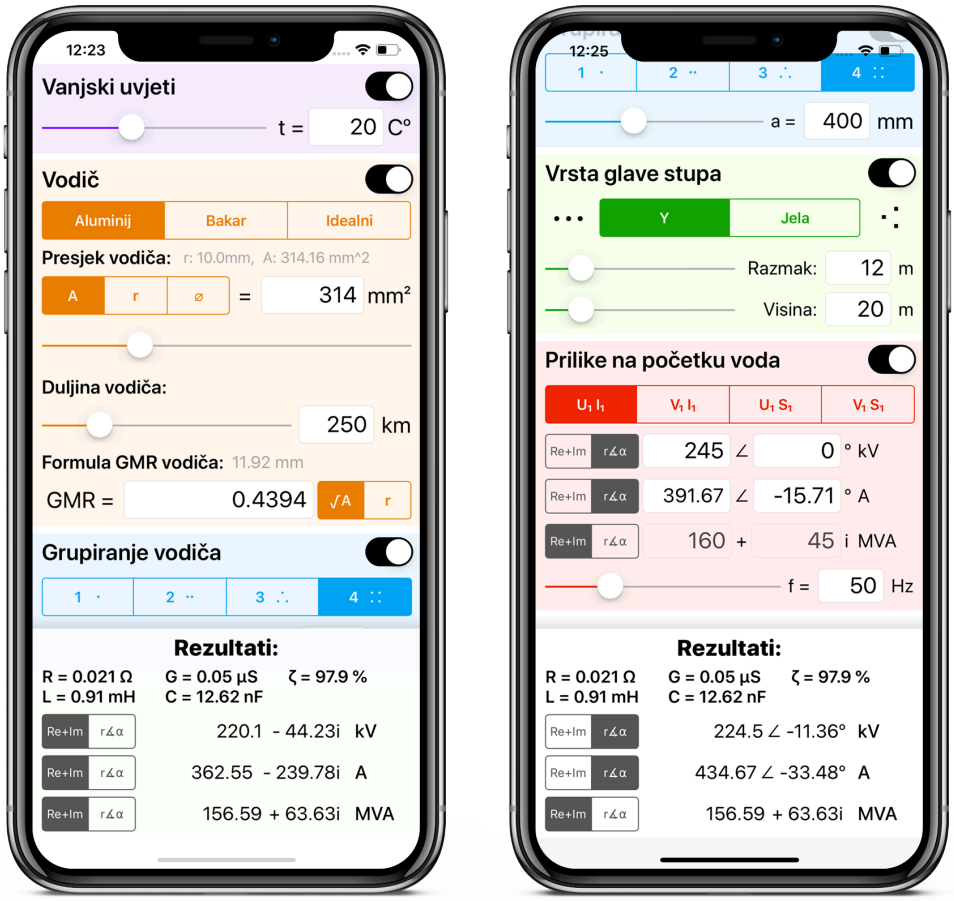
4.2.1. Grafičko sučelje na iOS simulatoru

Aplikacija se pokreće na simulatoru iOS uređaja (iPad ili iPhone). Ulazni parametri izvedbe voda zadaju se koristeći kontrolne elemente: brojčani iznosi pomoću kliznih kontrola ili izravnim unosom iznosa parametra, a odabir između više različitih načina unosa pomoću segmentne kontrole. Nakon unosa svih ulaznih parametara (uključujući i iznos napona i struje na početku voda) instantno se pojavljuje rezultat — sva četiri primarna parametra (konstante) voda, faktor učinkovitosti i krajnji fazori napona i struje prikazani u polarnom ili algebarskom kompleksnom zapisu.



Slika 4.2.1: *Izgled grafičkog sučelja na iPad-u* [20]

4.2.2. Kontrolni elementi



Slika 4.2.2: *Prikaz svih kontrola sučelja na uređaju iPhone X* [20]

Na slici 4.2.2 prikazano je cjelovito grafičko sučelje aplikacije podijeljeno na dva dijela. Vide se 5 grupa kontrolnih elemenata označene različitim bojama:

- **Vanjski uvjeti:** temperatura na kojoj se nalazi materijal vodiča
- **Vodič:** Materijal vodiča i njegov presjek, duljina vodiča (voda) i reducirani radijus vodiča.
- **Grupiranje vodiča:** Broj vodiča u snopu i razmak između njih
- **Vrsta glave stupa:** Geometrijski raspored vodiča - može biti stup ili jela
- **Prilike na početku voda:** Zadani iznosi napona i struje (ili snage) na početku voda i frekvencija

Vidi se da presjek vodiča može biti zadan i pomoću polumjera ili promjera. Reducirani radijus također može biti zadan i u obliku faktora K (izraz 3-2-2). Napon i struja mogu biti zadani u četiri različita oblika ovisno koristi li se napon i struja ili napon i snaga i radi li se o linijskom ili faznom naponu. Moguće je odabrati i oblik zapisa u kojem se unose kompleksne vrijednosti (polarni ili algebarski). Pri prikazu rezultata također postoji izbor prikaza u polarnom ili algebarskom kompleksnom zapisu.

5. Zaključak

U ovom završnom radu obrađen je matematički model dugog trofaznog voda. Parametri poput materijala vodiča, broja vodiča u snopu, geometrijskog rasporeda faza i vanjske temperature po izrazima iz literature uvršteni su u formule i na taj način su definirani primarni parametri (konstante) voda. Zbog mnogobrojnih ulaznih parametara i računa s kompleksnim brojevima izračun ovih parametara je složen i dugotrajan postupak.

Pri definiciji svakog njegovog dijela, matematički model dugog trofaznog voda je preveden u programski model. Sve složene klase, objekti i metode programskog modela zajedno omogućuju gotovo istovremeni izračun primarnih parametara, a u nastavku i strujno-naponskih prilika na jednom kraju voda za definirane prilike na drugom.

Grafičko sučelje iOS aplikacije sa dodirnim kontrolama nadalje omogućuje lakše postavljanje i izmjenu bilo kojeg parametra dalekovoda i ulaznih valova struje i napona, a ispisom rezultata na ekran pravovremeno pruža uvid u nastalu promjenu. Grafičkim prikazom ujedno je omogućen i jasniji prikaz svih ulaznih parametara ali i rezultata - iznosa primarnih parametara, učinkovitosti prijenosa i kompleksnih vrijednosti struje/napona/snage na kraju voda.

6. Literatura

[1] H. Pandžić, I. Kuzle — *Elektroenergetika,*
Neodidacta, Zagreb 2009

[2] Transmission Lines,
https://circuitglobe.com/transmission-lines.html (pristupljeno 02.09.2019.)

[3] Xcode na Mac App Store,
https://itunes.apple.com/us/app/xcode/id497799835 (pristupljeno 15.07.2019.)

[4] A fast look at Swift, Apple's new programming language,
https://arstechnica.com/gadgets/2014/06/a-fast-look-at-swift-apples-new-programming-language/,
(pristupljeno 15.07.2019.)

[5] Names for variables,
http://chortle.ccsu.edu/qbasic/chapter03/bc03_8.html, (pristupljeno 15.07.2019.)

[6] Apple Inc. (2019.). The Swift Programming Language (Swift 5.0).
https://books.apple.com/us/book/the-swift-programming-language-swift-5-0/id881256329
(pristupljeno 15.07.2019.)

[7] F. Ashby, H. Shercliff, D. Cebon - Materials: Engineering, Science, Processing and Design (2010).
https://books.google.hr/books?id=Og2C4bCDSX8C, (pristupljeno 15.07.2019.)

[8] Temperature Dependence of Resistivity,
https://www.askiitians.com/iit-jee-electric-current/temperature-dependence-of-resistivity/,
(pristupljeno 15.07.2019.)

[9] Vodovi visokog i niskog napona - Elektroindustrijska i obrtnička škola Rijeka,
http://mabacic.eios.hr/oo/vodovi.pdf (pristupljeno 15.07.2019.)

[10] Lajos Józsa – *Parametri nadzemnih vodova*,
Sveučilište Josipa Jurja Strossmayera u Osijeku, Elektrotehnički fakultet, Kneza Trpimira 2b, 2006.

[11] Jordi-Roger Riba, Analysis of formulas to calculate the AC resistance of different conductors' configurations, https://core.ac.uk/download/pdf/41822519.pdf (pristupljeno 15.07.2019.)

[12] Single Circuit and Double Circuit Transmission Lines,
https://www.electrotechnik.net/2011/11/single-circuit-and-double-circuit.html
(pristupljeno 15.07.2019.)

[13] S. Nikolovski, D. Šljivac, *Elektroenergetske mreže - zbirka riješenih zadataka (treće izdanje)*,
Elektrotehnički fakultet – Sveučilište Josipa Jurja Strossmayera u Osijeku, Osijek, 2003.

[14] https://commons.wikimedia.org/wiki/File:Swift_logo_with_text.svg (pristupljeno 02.09.2019.)

[15] http://www.owlwire.com (pristupljeno 20.09.2019.)

[16] https://www.vwcable.com/aac-sagebrush-2250mcm-all-aluminum-conductor-aac-cable/ (pristupljeno 20.09.2019.)

[17] http://www.apar.com/aluminium-alloy-conductors.php (pristupljeno 02.09.2019.)

[18] https://www.indiamart.com/proddetail/bundled-conductor-8435537655.html (pristupljeno 02.09.2019.)

[19] https://jooinn.com/high-voltage-towers.html (pristupljeno 02.09.2019.)
https://commons.wikimedia.org/wiki/File:Single-circuit_pylon_70_kV_Opwijk_BE_2016.jpg (pristupljeno 02.09.2019.)

[20] Vlastita zbirka autora završnog rada, izrađeno u programima Adobe Photoshop i Keynote.

7. Sažetak

Cilj ovog završnog rada je što bolje odrediti primarne parametre voda — veličine koje utječu na valove napona i struje prilikom prijenosa na veće udaljenosti. Nakon kratkog obilaska osnovnih značajki razvojne okoline XCode i programskog jezika Swift, matematički model nadzemnog voda preveden je u programski uzimajući pri tome u obzir sve bitne čimbenike navedene u literaturi i izraze po kojem se ti čimbenici računaju. Pri modeliranju obrađena je cjelovita izvedba nadzemnog voda - od najjednostavnije gradivne jedinice materijala, preko geometrijskog rasporeda trofaznog voda pa sve do složenijih kao što je snop vodiča. To je naposljetku omogućilo precizno i pravovremeno simuliranje konstanti voda promjenom bilo kojeg parametra u bilo kojem dijelu njegovog programskog modela. Time je ujedno pružen i trenutni uvid u gubitak snage za različite razine napona.

Abstract

The goal of this final thesis was to define the primary parameters of a transmission line - sizes that affect the current and voltage waves in case of long-distance transmission. After a swift tour of basic features of XCode developing environment and the Swift language, mathematical model of transmission line was translated into a programming one taking into account all the relevant factors referenced in the literature including the expressions they are derived from. Complete design of the transmission line has been worked through during the modeling process - starting from the fundamental build unit material, continuing with geometric distribution of the 3-phase line, up until more complex building parts like conductor bundle. Finally, this enabled precise and real-time simulation of the transmission line primary parameters, changing any parameter from any building part of the entire programming model. Instant insight into power loss for various voltage levels is given as well.