Antenski nizovi zasnovani na Cantorovim krivuljama

Dujak, Andrija

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:676977

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET
Stručni studiji

Antenski nizovi zasnovani na Cantorovim krivuljama

Završni rad

Andrija Dujak

Student: Andrija Dujak
Mentor: Slavko Rupčić
Akademska godina: 2020./2021.

Osijek, 2021. Godina

SADRŽAJ

I.UVOD **2**

II. POTREBNA ZNANJA **3**
2.1.Cantorov skup 3
2.2.Povezane liste i rekurzivne 4

III.RAD PROGRAMA **5**
3.1. Priprema 5
3.2. Početak programa-alfa kut 8
3.3. Beta kut 9
3.4. Povezana lista 9

IV.IZVEDBA PROGRAMA **10**
4.1. Deklaracije 10
4.2. Glavni program 11
4.3. Funkcija CantorSet 12
4.4. Rezultati programa 13

V. ZAKLJUČAK **15**

VI. LITERATURA 17**VII.SAŽETAK**
18**IX. PROGRAM**
20

I.UVOD

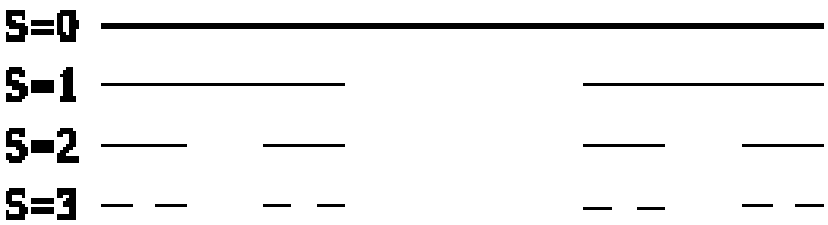
Ovaj završni rad objašnjava kako se izračuvana cantorov skup, kako rade povezane liste ,kako je kreiran program, kako radi program i kako je iskorišten cantorov skup u programu. Program vraća dva podatka niz kuta alpha i matricu kuta beta. Kut alfa označava kuteve u kugli koje su izračunate pomoću cantorovog skupa. A kut beta označava kut koji predstavlja polovicu valne dužine radio signala.

II. CANTOROV SKUP I POVEZANE LISTE

2.1 Cantorov skup

Skup točaka koji se dobije podjelom jednog intervala u tri jednaka podintervala, od tih tri podintervala izbaci se središnji iz skupa a druga dva intervala čine skup. Iz tog skupa uzimaju se prijašnji podintervali i ponovi se proces s njima, tako se može podijeliti do beskonačnosti ali mi zadajemo u koliko koraka će se ponavljati proces dajući nam omeđene granice. Njemački fizičar George Cantor je predstavio teoriju 1883. godine. Slika 2.1. prikazuje primjer cantorovog skupa do trećeg koraka podjele, vidljivo je da broj podintervala u skupu je jednak 2^s, veličina raspona svakog podintervala je jednaka:

$$\square \times (\frac{1}{3})\blacksquare^{\square}$$

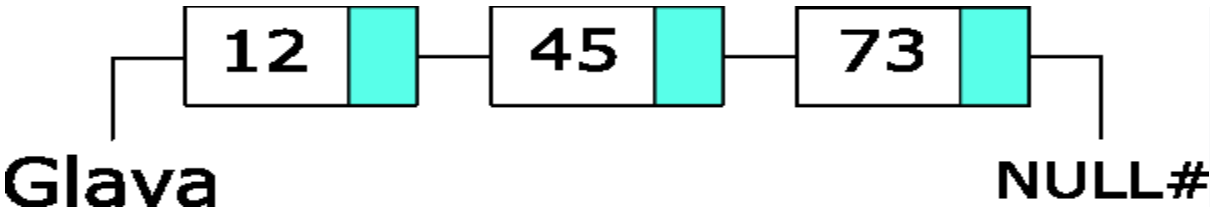


Slika. 2.1. *primjer cantorovog skupa*

Cantorov skup je omeđen, ne proteže se u beskonačnosti nego ima svoje granice. Iako je skup zatvoren u sebi sadrži beskonačno mnogo točaka što ga čini gustim. Ukupna mjera cantorovog skupa je jednaka nuli zbog toga što je skup beskonačni geometriski red. Svakim stupnjem veličina raspona podintervala se smanjuje, mjera se svakim stupnjem približava nuli. Za bilo koju točku u cantorovom skupu i bilo koje proizvoljno malo susjede točke postoji neki drugi broj s ternarni brojem od samo nula i dvojki, kao i brojevi čiji ternarni brojevi sadrže jedinice. Stoga je svaka točka u cantorovom skupu točka akumulacije cantorovog skupa, ali nijedna nije unutarnja točka. Zatvoreni skup u kojem je svaka točka akumulacijska točka u topologiji se naziva i savršenim skupom, dok zatvoreni podskup intervala bez unutarnjih točaka je nigdje gust u intervalu.

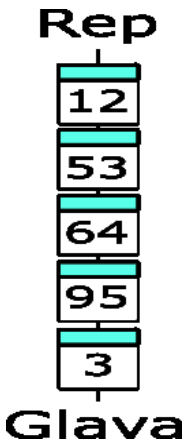
2.2.Povezane liste

Povezana lista je dinamička struktura podataka gdje elementi liste dinamički zauzimaju memorije. Kada dio liste nije više potreban zauzeti dio memorije se oslobodi bez promjene ostatka liste. Svi elementi liste sadrže dva podatka, svoju vrijednost i pokazivač na sljedeći element liste. Tako se dinamički sprema lista. Za korištenje povezane liste potreban je pokazivač koji pokazuje na prvi element liste, taj pokazivač se zove glava. za pristup jednom elementu potrebno je proći kroz sve ostale elemente u nizu dok se ne pristupi zatrazenog elementa. Da bi se dodao novi element u listi treba proći kroz cijelu listu i kod zadnjeg elementa usmjeriti pokazivač na novi element.



Slika. 2.2. *primjer povezane liste*

Stog je struktura podataka s LIFO (last in first out) principom, gdje najstariji element strukture se prvi koristi. Pomoću povezane liste se izvršava stog tako što dodajemo uz glavu pokazivač koji pokazuje na zadnji element koji se zove rep i koristimo funkcije push, pop, is empty i clear. Funkcija push dodaje novi element s zadanom vrijednost na kraju liste pomoću pokazivača rep, pop nam daje prvu vrijednost liste pomoću pokazivača glave i izbriše taj element i pokazivač glava pokazuje na sljedeći element liste.



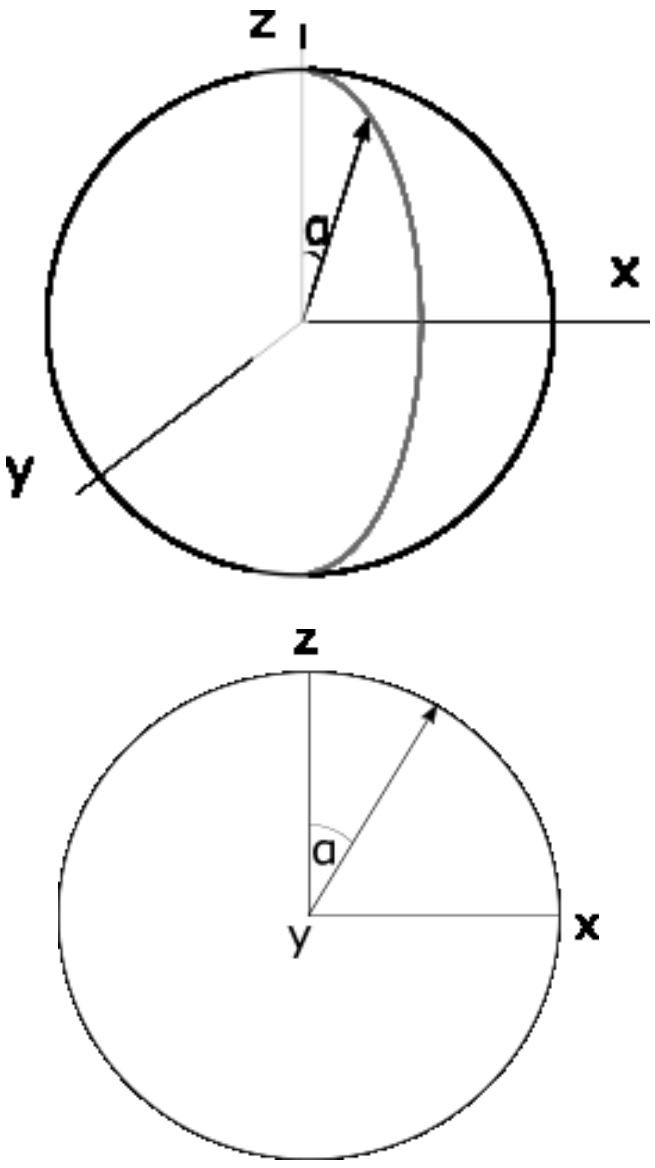
Slika. 2.3. *primjer stoga*

Rekurzivne funkcija je funkcija koja poziva samu sebe unutar svojih naredbi. Za prestanak rada funkcije mora postojati ne rekurzivni put izlaska iz te funkcije, to su trivijalni slučajevi za ulazne parametre funkcije. Ove tri osnovne pojmove programiranja je potrebno znati za pregled računalnog programa jer ih koristi za rad.

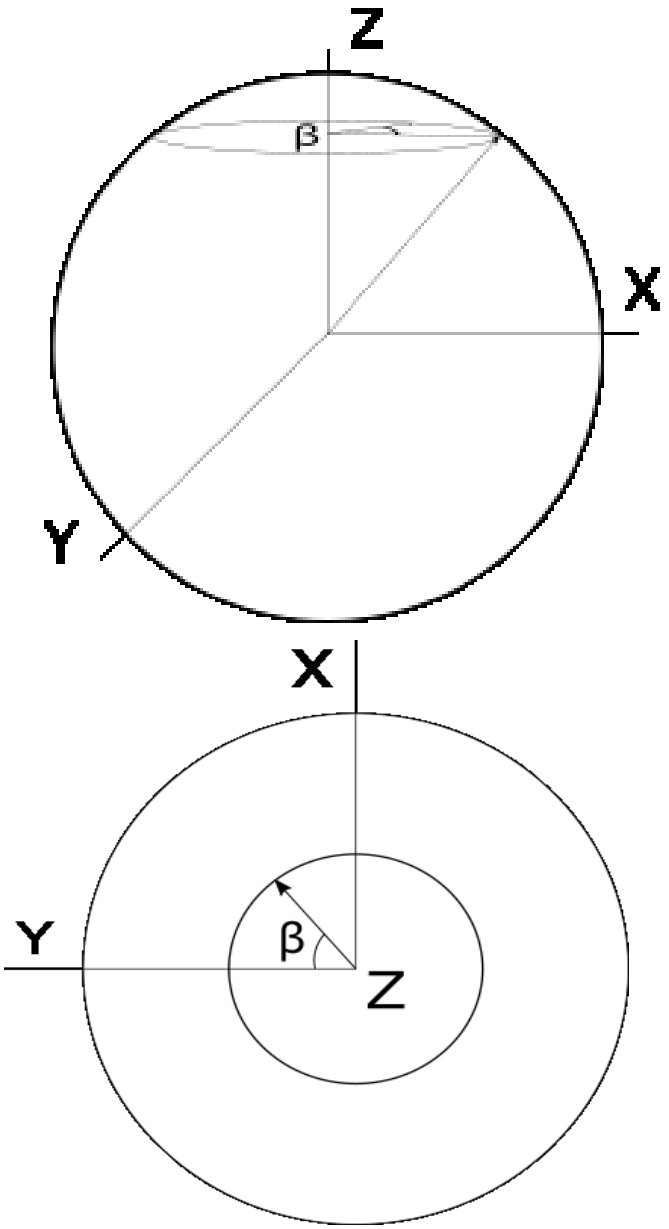
III. IZRADA SIMULACIJE ZA IZRAČUN POZICIJA ANTENA NA SFERI PREMA CANTOROVOM SKUPU

3.1. Priprema

Za početak programa uzimaju se konstante kao što su radijus kružnice sfere, broj cantorovog stupnja i pi, Na sferi postoje dva kuta koja su važna za program, kut alpha koji označuje udaljenost pojedinih cantorovih podskupova i obuhvaća 90° z osi sfere. I kuta beta koji označuje udaljenost pojedinih antena unutar cantorovih podskupova i obuhvaća 360° y osi sfere.



Slika. 3.1. *Prikaz kuta alpha u sferi i pogled sa strane sfere*



Slika. 3.2. *Prikaz kuta beta u sferi i pogled sa visine sfere*

Ti kutovi su izlazi iz programa i koriste se za simulaciju. kuta alfa je niz dok je kut beta matrica. Zbog toga što kut beta treba pozicije kuta alfa za pozicioniranje. sve antene su jednako udaljene od drugih antena s najmanjom vrijednosti:

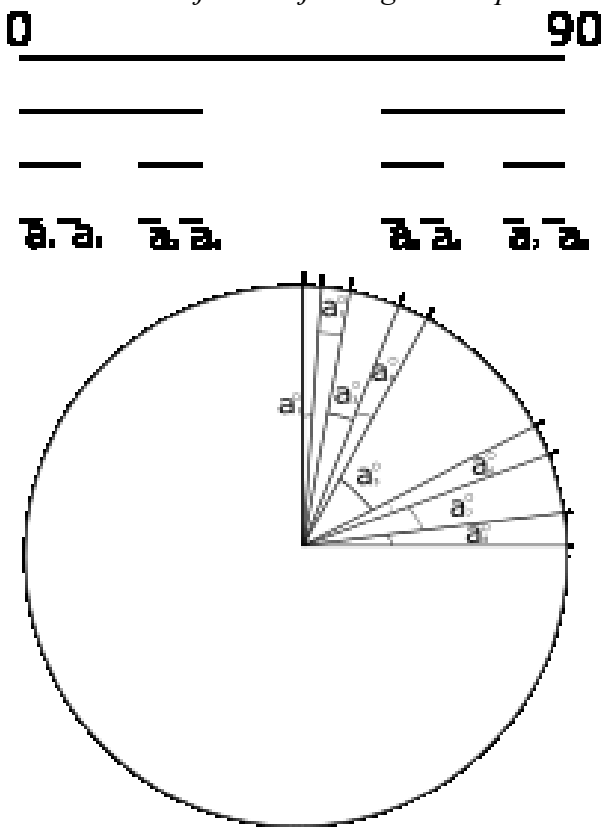
$$\theta = \frac{\square}{\square} \times \frac{1}{2}$$

Formula 3.1. *Izračun duljine sigma*

gdje je θ duljina u metrima koje označava udaljenost antena u opsegu kružnice, C je brzina svjetlosti 299 792 458 m/s i f je postavljena frekvencija 1.75 GHz. Vidljivo je na slici 2.6. da su sve antene udaljene kutom beta koji je određen ovisno o udaljenosti pojedinih antena pomoću duljine θ. Jer ako je duljina kružnice ne daje prirodni broj kada se dijeli s duljinom θ, onda se duljina povećava dok dijeljenjem ne proizvodi prirodni broj:

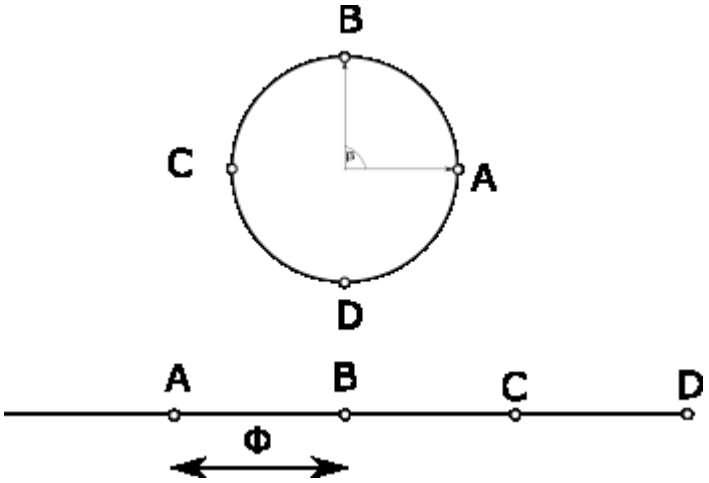
$$\frac{2\square\square}{\square} \neq \square \rightarrow \square = \frac{2\square\square}{\square\square\square(\frac{2\square\square}{\square})}$$

Formula 3.2. *Promjena duljina sigma za optimalan rad*



Slika 3.3. *Prikaz smještaja alpha kuta na sferi pomoću cantorovog niza*

Za spoznaju svih alpha kutova kreira se cantorov skup dužine 90 mjernih jedinica s 3 stupnjeva. Na trećem stupnju cantorovog skupa koristi se gornja granica svakog intervala za alpha kutove. Znači prvi interval od cantorovog skupa je [0,3.3333], 3.3333 se uzme iz intervala i sprema u niz alpha kuta kao a_1 , onda se koristi drugi interval [6.6666,10], 10 se uzme iz intervala i sprema u niz alpha kuta kao a_2 itd. Sve dok se ne koriste svi intervali.



Slika 3.4. *prikaz razmještaj antena pomoću opsega kružnice*

Za cantorov stupanj uzeo sam da ce ih biti 3 jer najbolje prikazuje antene na sferi s malim inicijalnim radijusom, a dužina raspona je od 0 do 90 da predstavlja alfa kut.

3.2. Programiranje – za određivanje kuta alfa

Nakon alokacije svih varijabnih program započinje s rekurzivnom funkcijom CantorSet koji koristi nula, devedeset i nula kao ulazne parametre. nula je donja granica, devedeset je gornja granica i druga nula je trenutni korak za cantor skup. funkcija pregleda ako je trenutni cantor skup jednak stupnju kojeg smo zadali na početku programa, ako jest funkcija će pregledati ako je lista napravljena, ako nije napraviti će se nova lista s vrijednosti gornje granice, u suprotnom dodaje se novi element na listi s vrijednostima gornje granice. Ako trenutni korak nije jednak stupnju onda se zapisuju dvije nove varijable nova gornja granica i nova doljna granica koji se računaju po ovoj formuli:

$$\square\square\square = \square\square + (\square\square - \square\square) * (2/3)$$
$$\square\square\square = \square\square + (\square\square - \square\square) * (1/3)$$

Formula 3.3. *Izračun nove doljnje granice i nove gornje granice*

Nakon toga se dva puta poziva CantorSet funkcija. Prvi s parametrima doljnje granice, nove gornje granice i trenutni korak plus jedan, drugi s parametrima nove doljnje granice, gornje granice i trenutnog koraka plus jedan. I time su se jednostavno izračunali svi alfa kutovi kruga, samo se koristi for petlja da se spreme u niz vrijednosti tako da se izvade iz po redu jer je funkcija tako napravljena da ide od najmanjeg kuta do najvećeg.

3.3. Programiranje – priprema za određivanje kuta beta

Za beta kut potrebno je poznavati alfa kut zbog toga što je potrebno izračunati radijus kružnice kod kuta alfa s sinusom poučkom:

$$\square\blacksquare_{[\square]} = |\square\square\square\blacksquare^{-1}(\square\blacksquare_{[\square]})| \times \square$$

Formula 3.4. *Izračun radijusa za kut beta*

nakon izračuna radijusa možemo dobiti opseg kružnice nad kutom alfa i podijeliti tu vrijednost s dužinom θ da dobijemo broj antena kojih se mogu staviti na toj kružnici. Ako broj nije prirodan onda se povećava dužina θ kao na formuli 3.2. za optimalan rad. Nakon toga pomoću for petlje zapisuju se svi kutovi beta s formulom:

$$\beta\blacksquare_{[\square][\square]} = \frac{\square \times \square \times 360}{2 \times \square \times \square\blacksquare_{[\square]}}$$

Formula 3.4. *Izračun kuta beta*

S time program završava radom i predaje niz alfa i matricu beta.

3.4. Povezana lista

Za povezanu listu modificirao sam javno dostupni modul povezane liste flibs[1] . Flibs model povezane liste ima dvije glavne razlike od povezane liste koje ja koristim za program. Prvo treba dodati pokazivač koji pokazuje na rep liste, s njim možemo dodati elemente liste bez da prolazimo kroz cijelu listu. Druga razlika je uzimanje podataka iz liste, u programu uzima se prvi element iz liste po LIFO principu i odmah se briše, dok flibs kod flibs modela može se uzeti bilo koji element iz liste i pregledati njegove podatke.

.

IV.IZVEDBA PROGRAMA

4.1. Deklaracije

Za početak je potrebno deklarirati sve argumente i module koje se koriste u programu jer fortran zahtijeva deklaraciju svih podataka, funkcija i modula prije početka rada.

```
module MYDATA_MODULE

type MYDATA
    character(len=20) :: string
end type MYDATA

end module

module MYDATA_LISTS
    use MYDATA_MODULE, LIST_DATA => MYDATA

    include "linkedlist.f90"
end module MYDATA_LISTS
```

Slika 4.1. *Deklaracija modula "linkedlist"*

U slici 4.1 je prikazana deklaracija modula "linkedlist" koja je temelj rada povezane liste u programu. Taj modul je od FLIBS dokumentacije rada s fortranom.

```
program CantorCircularProces
    use MYDATA_MODULE
    implicit none
    real :: theta,num_of_cantors,circle_leanght,sinus_alpha,sinus_beta, selected_radius, replacment_angle
    integer :: cantor_step,n,m,radius
    type(LINKED_LIST), pointer :: list
    type(LINKED_LIST), pointer :: head
    real, dimension(:,:), allocatable :: alpha
    real, dimension(:,:), allocatable :: beta
    real, parameter:: pi=3.1415926
    common /coeff/ cantor_step
```

Slika 4.2. *Deklaracija svih podataka u glavnom programu*

Na slici 4.2. vidljive su sve deklaracije svih podataka u programu kao što su realni brojevi theta, broj cantora, dužina kružnice; integeri kao što su cantorov korak, radius i pomoćni integeri n i m za for petlje; pokazivači na povezane liste lista i glava, realne niz alpha i realna matrica beta; parametar pi i deklaracija da je podatak cantorov korak globalan.

```
RECURSIVE SUBRROUTINE CantorSet(dg, gg, i,list)
    integer, intent(in)::i
    type(LINKED_LIST), pointer :: list
    type(LINKED_LIST), pointer :: next
    real, intent(in) :: dg,gg
    real             :: ndg, ngg, cantor_step
    common /coeff/ cantor_step
```

Slika 4.3. *Deklaracija svih podataka u funkciji*

I na kraju su deklaracije kod funkcije kao što su ulazni podaci donja granica, gornja granica, lista i pomoćni podatak “i”.

4.2. Glavni program

U glavnom programu se prvo upisuje broj cantorovog koraka, radijusa kruga, theta i broj intervala u cantorovom skupu kao što je vidljivo u slici 4.4.

```
cantor_step = 3
radius = 30


theta = (299792458/(1.75*(10**9)))/2

theta = theta * 100

num_of_cantors = 2*radius*pi
num_of_cantors = num_of_cantors- MODULO(num_of_cantors,theta)
num_of_cantors = (num_of_cantors/theta)+1
```

Slika 4.4. *Zadavanje vrijednosti podacima.*

Nakon toga se poziva funkcija za kreiranje povezane liste i zadavanja vrijednosti iz cantorovog skupa unutar nje. Objašnjenje rada funkcije se nalazi u sljedećem potpoglavlju. S time gotovo program ide dalje, alocira veličinu niza alpha i matrice beta da se mogu koristiti. Kod niza alpha svi podaci se postavljaju sa podataka direktno iz povezane liste.

```
call CantorSet(0,90,1,list)

allocate(alpha(1:2**cantor_step))

allocate(beta(2**cantor_step:num_of_cantors))

do n=1,2**cantor_step
    alpha(1:n) = CALL list_get_data(list)
    head = list
call list_delete_element(list,head)
end do
```

Slika 4.5 *poziv na “CantorSet” funkciju ,alociranje kuta alpha i beta, i postavljanje kuta alpha*

S time ostane nam još samo jedna stvar u glavnom programu i to je postavljanje matrice beta koja se dobije tako da izračunamo radius kružnice na poziciji kuta alpha. Nakon toga se izračuna koliko točaka se nalaze u toj kružnici na kojim bi mogli postavljati antene. Zadaje se kut koji će zamijeniti kut θ tako da su sve točke jednako udaljene i jedne po jedno se sve točke postavljaju u matrici beta.

```
do n=1,2**cantor_step
    sinus_alpha=sin(alpha(n))
    if (sinus_alpha < 0) then
        sinus_alpha = -sinus_alpha
    end if

    selected_radius = radius * sinus_alpha
    circle_leanght = 2 * selected_radius * pi
    num_of_cantors = circle_leanght/theta
replacment_angle= circle_leanght/int(num_of_cantors)
    do m=1,int(num_of_cantors)
        beta(n)(m) = ((m*replacment_angle*360)/(2*pi*selected_radius)
    end do
end do
```

Slika 4.6. *Postavljanje matrice beta*

4.3. Subrutina CantorSet

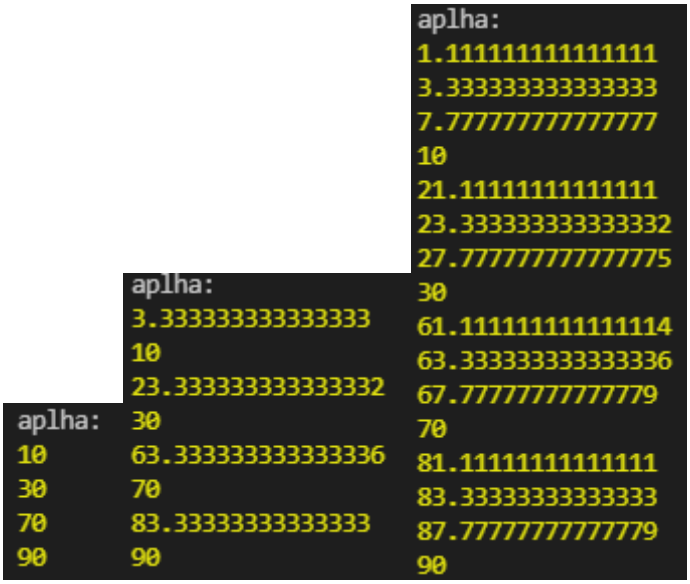
```
RECURSIVE SUBRROUTINE CantorSet(dg, gg, i,list)
    integer, intent(in)::i
    type(LINKED_LIST), pointer :: list
    type(LINKED_LIST), pointer :: next
    real, intent(in) :: dg,gg
    real             :: ndg, ngg, cantor_step
    common /coeff/ cantor_step

    if(i==cantor_step) then
        if(list_count(list)==null) then
                call list_create(list,gg)
        else
                next = list%next
                DO WHILE ( next != null )
                    next = list%next
                END DO
                call list_insert(next,gg)
        end if
    else
        ndg=dg+(gg-dg)*(2/3)
        ngg=dg+(gg-dg)*(1/3)

        CALL CantorSet(dg,ngg,i+1,list)
        CALL CantorSet(ndg,gg,i+1,list)
    end if

end subroutine
```

Slika 4.7. *Funkcija "CantorSet"*

Funkcija "CantorSet" je rekurzivna funkcija koja se ponavlja sve dok ne dodaje sve podatke u povezanoj listi. Zbog toga što je cantorov stup jednak 3 broj podataka je jednak 2^3. Znači sve dok 8 podatka se ne postavi funkcija će dalje raditi. Funkcija prvo pregledava ako je pomoćni podatak "i" jednak cantorovom skupu. Ako jest to znači da je cantorov niz dovoljno puta podijeljen i da podatak gornje granice se treba spremiti u povezanoj listi. zbog redoslijeda pozivanja rekurzivne funkcije podatci će se spremiti od najmanjeg podatka do najvećeg. Ako i nije jednak cantorovom skupu onda se zadaje nova doljnja granica i nova gornja granica za

nove intervale. Ponovno se poziva funkcija “CantorSet” s parametrima stare doljnje granice i nove gornje granice, i poziva se funkcija s parametrima nove doljnje granice i stare gornje granice tako da se kreira dva intervala od jednog. I tako dalje nastavi sve dok i nije jednak cantorovom skupu.

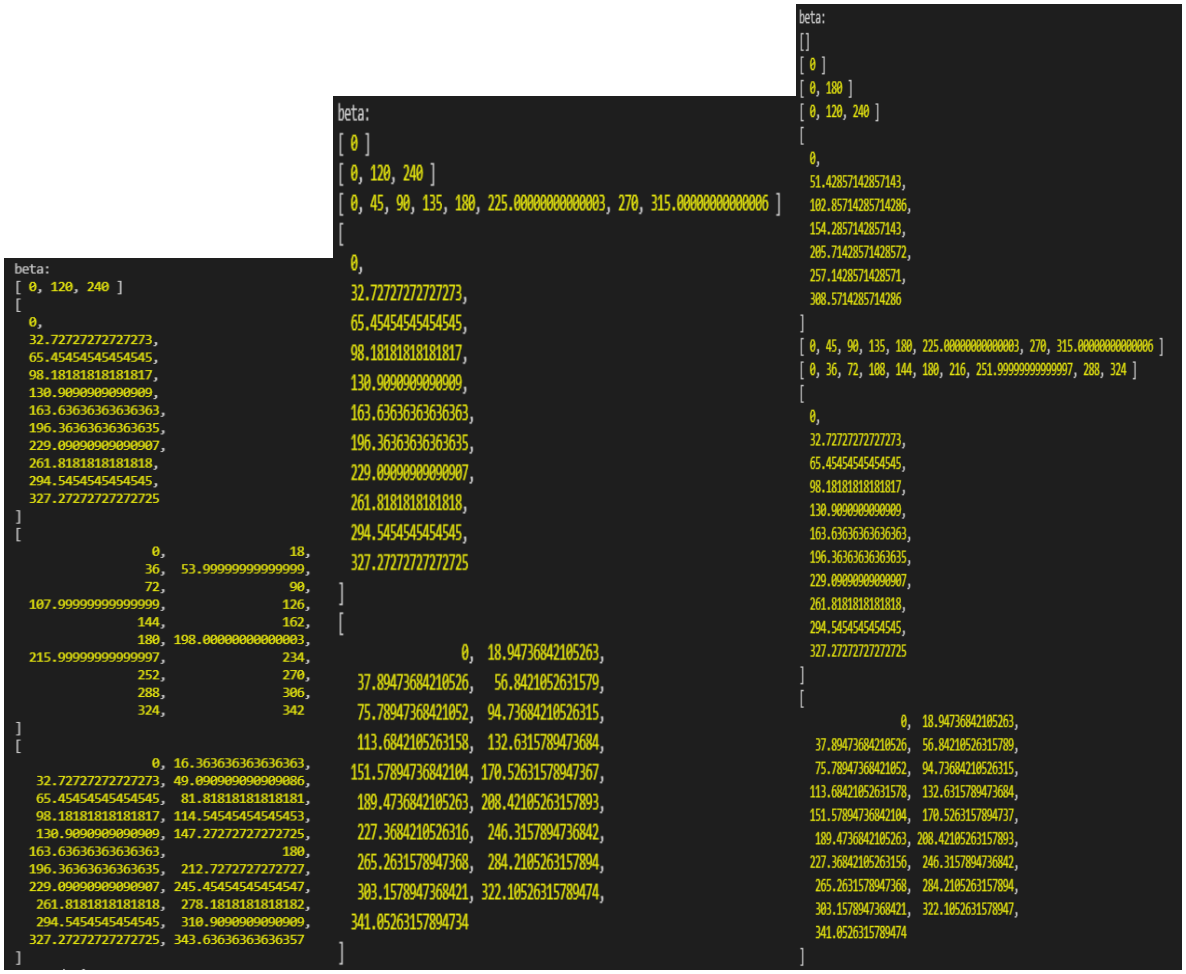
4.4. Rezultati programa



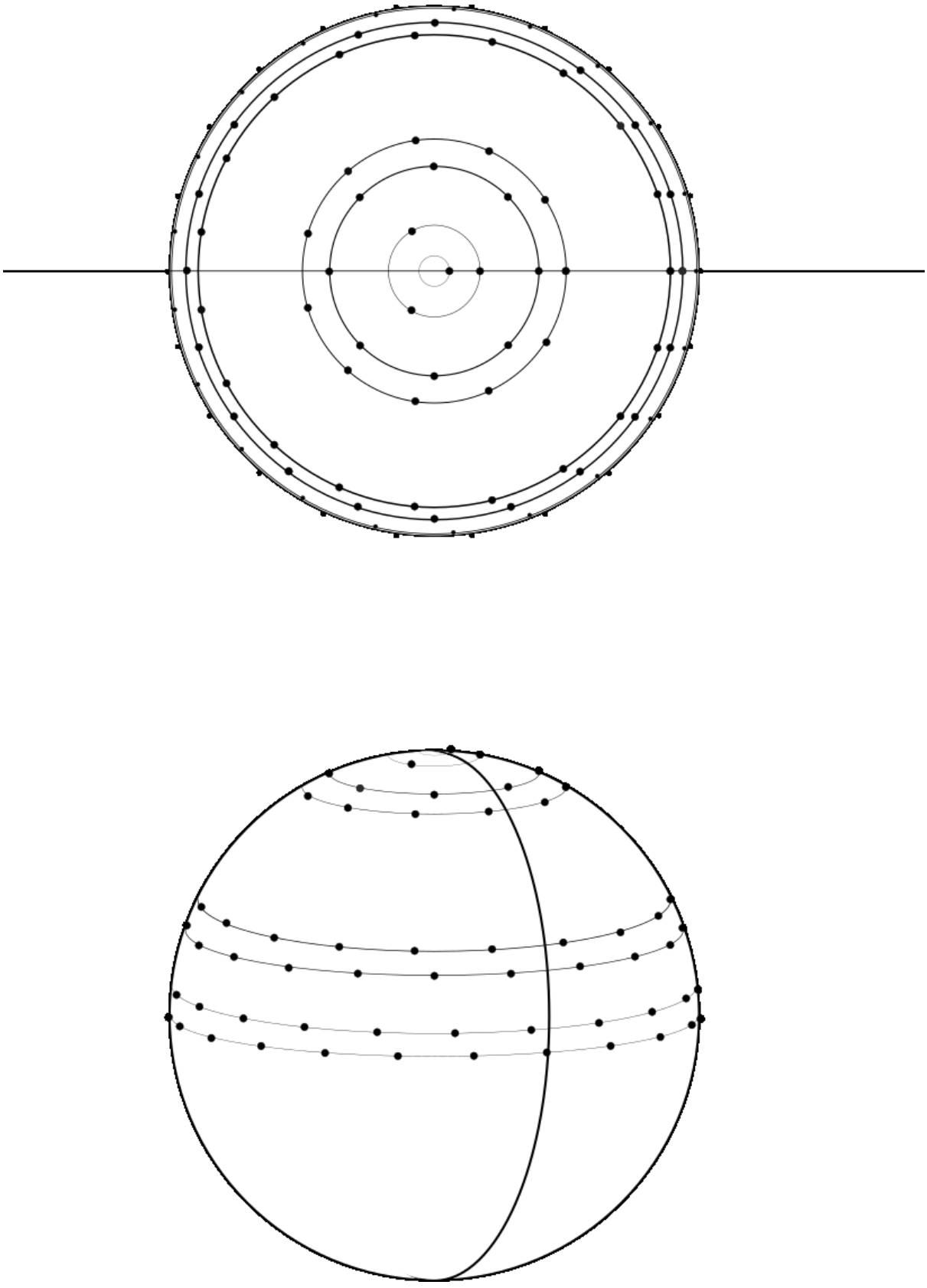
Slika 4.8. *Rezultat alpha kuta za cantorov stupanj: 2,3 i 4.*

Na slikama 4.8 i 4.9 vidljivi su rezultati kuteva alpha i beta koje se spremaju kao nizovi podataka. Zbog velike količine podataka nije moguće ispisati sve rezultate od jednom na stranici. pa sam uzeo prvih 3 do 4 niza u matrici. Na slici 4.10. vidljiv je kako bi izgledala kugla s antenama koji su smješteni prema alpha i beta kutovima koje smo dobili s programa kada je cantor stupanj jednak trojci.

Slika 4.9. *Rezultat beta kuta za cantorov stupanj 2,3 i 4*



Slika 4.9. *Rezultat beta kuta za cantorov stupanj 2,3 i 4*



Slika 4.10. *vizualni prikaz rada programa s cantorovim stupnjem 3*

V. ZAKLJUČAK

Pomoću matematičkih jednađbi,cantorovog niza i računalnim jezikom fortranu uspješno je kreiran program koji olakšali dugotrajni proces izračuna svih kutova koji su potrebni za postavljanje antena na maloj kugli za simulacijske potrebe. I u ovom radu je objašnjeno kako i zašto radi program.

VI. LITERATURA

[1] http://flibs.sourceforge.net/linked_list.html

[2] M. Nilsson, J.W. Schuster, G.Vecchi, IEEE Antennas and Propagation Society International Symposium. 1998 Digest. Antennas: Gateways to the Global Network. Held in conjunction with: USNC/URSI National Radio Science Meeting (Cat. No.98CH36)

[3] F. Diaz, Non-Uniform Cantor Ring Arrays - SUNFEST

VII.SAŽETAK

Za rad je bilo potrebno kreirati program koji bi izračunao kutove unutar kugle. Kut alfa koji predstavlja kut u odsječku kugle s y osi i kut beta koji predstavlja kut u odsječku kugle s x osi. Kut alfa se postavio pomoću cantorovog skupa gdje se uzima gornja granica svih intervala od cantorovog skupa s određenim cantorovim stupnjem, A kut beta je određen tako da se izračuna polovica valne duljine radio signala i podijeli se s kružnicom kugle koji ovisi o kutu alfa. Sve se računa pomoću programa koji je koristio povezane liste, rekurzivne funkcije i matematičke formule za rad. Moralo se prvo izračunat cantorov skup, za to je potrebno odrediti svojevrsno cantorov stupanj koji je prirodan broj. Nakon čega se uzima duljina od dužine devedeset i dijeliti je na 3 jednaka dijela. Srednji dio se izbriše i ostane nam 2 dijela koje nadalje dijelimo na 3 dijela sve dok nismo izvršili taj proces u toliko puta koliko smo zadali cantorov stupanj. Time smo dobili alfa kut s gornjom granicom tih preostalih dužina. Ti kutevi se koriste za izračunavanje kružnice u kugli gdje je kut jednak alfi. Izračunavanje dužine kružnice možemo izračunati broj antena koje možemo postaviti na na njoj tako da podijelimo dužinu s dužinom polovice valne dužine radio signala. Pretvorbom valne dužine u kut i množenjem s rednim brojem antene možemo izračunati sve kutove za betu. I time dobijemo sve podatke.

for the project it was necessary to create a program that would calculate the angles inside a sphere. The angle alpha representing the angle in the section of the sphere with the y axis and the angle beta representing the angle in the section of the sphere with the x axis. The angle alpha is set using a Cantor set where the upper limit of all intervals from the Cantor set with a certain Cantor degree is taken, and the beta angle is determined by calculating half the wavelength of a radio signal and dividing it by the circle length of the sphere depending on the angle alpha. Everything is calculated using a program that used linked lists, recursive functions, and mathematical formulas to work with. It was necessary to first calculate the Cantor set, for this it is necessary to determine the Cantor degree which is a natural number. Then take a length of ninety and divide it into 3 equal parts. The middle part is taken out and we are left with 2 parts which we further divide into 3 parts until we have performed this process as many times as we have given the cantor degree. This gave us alpha angles with an upper limit of these remaining lengths. These angles are used to calculate a circle in a sphere where the angle is equal to alpha. Calculating the length of a circle we can calculate the number of antennas we can place on it by dividing the length by half the wavelength of the radio signal. By converting the wavelength to an angle and multiplying by the ordinal number of the antenna, we can calculate all the beta angles. And with that we get all the data.

VIII.ŽIVOTOPIS

Rođen 27.9.1998. u Novoj Gradišci. Živio sam u Okučanima sve do 4 godine života kada sam se preselio u Staroj Gradišci. Tamo sam otišao u područnoj školi Stara Gradiška, pokazujuči prosječno ili ispod prosječno znanje u svim predmetima osim informatike. Zbog znatiželje i brzog shvaćanja informacijskih tehnologija upisao sam se u tehničkoj školi Požega kao mehatroničar gdje sam kao u osnovnoj školi pokazao prosječne ili ispodprosječne rezultate na svim predmetima koje nisu povezane s radom na računalima ili s informacijskim tehnologijama. Na predmetima kao “programiranje u c++” prikazao sam iznadprosječne rezultate. Htio sam nastaviti s proučavanjem programskih jezika, programskih okvira, načina programiranja itd. zbog toga sam se prijavio na FERIT. Stručnu praksu sam izvršio u tvrtci informatika fortuno d.o.o.

IX. PROGRAM

```
module MYDATA_MODULE

type MYDATA
   character(len=20) :: string
end type MYDATA

end module

module MYDATA_LISTS
   use MYDATA_MODULE, LIST_DATA => MYDATA

   include "linkedlist.f90"
end module MYDATA_LISTS



program CantorCircularProces
   use MYDATA_MODULE
   implicit none
   real :: theta,num_of_cantors,circle_leanght,sinus_alpha,sinus_beta, selected_radius,
replacment_angle
   integer :: cantor_step,n,m,radius
   type(LINKED_LIST), pointer :: list
   type(LINKED_LIST), pointer :: head
   real, dimension(:,:), allocatable :: alpha
   real, dimension(:,:), allocatable :: beta
   real, parameter:: pi=3.1415926
   common /coeff/ cantor_step


   cantor_step = 3
   radius = 30


   theta = (299792458/(1.75*(10**9)))/2

   theta = theta * 100

   num_of_cantors =2*radius*pi
   num_of_cantors = num_of_cantors- MODULO(num_of_cantors,theta)
   num_of_cantors = (num_of_cantors/theta)+1
```

```
   call CantorSet(0,90,1,list)

   allocate(alpha(1:2**cantor_step))

   allocate(beta(2**cantor_step:num_of_cantors))

   do n=1,2**cantor_step
      alpha(1:n) = CALL list_get_data(list)
      head = list
         call list_delete_element(list,head)
   end do

   do n=1,2**cantor_step
      sinus_alpha=sin(alpha(n))
      if (sinus_alpha < 0) then
         sinus_alpha = -sinus_alpha
      end if

      selected_radius = radius * sinus_alpha
      circle_leanght = 2 * selected_radius * pi
      num_of_cantors = circle_leanght/theta
         replacment_angle= circle_leanght/int(num_of_cantors)
      do m=1,int(num_of_cantors)
         beta(n)(m) = ((m*replacment_angle*360)/(2*pi*selected_radius)
      end do
   end do

   call reset()

end program CantorCircularProces


RECURSIVE SUBRROUTINE CantorSet(dg, gg, i,list)
   integer, intent(in)::i
   type(LINKED_LIST), pointer :: list
   type(LINKED_LIST), pointer :: next
   real, intent(in) :: dg,gg
   real            :: ndg, ngg, cantor_step
   common /coeff/ cantor_step
```

```
    if(i==cantor_step) then
       if(list_count(list)==null) then
             call list_create(list,gg)
       else
             next = list%next
             DO WHILE ( next != null )
                next = list%next
             END DO
             call list_insert(next,gg)
       end if
    else
       ndg=dg+(gg-dg)*(2/3)
       ngg=dg+(gg-dg)*(1/3)

       CALL CantorSet(dg,ngg,i+1,list)
       CALL CantorSet(ndg,gg,i+1,list)
    end if

end subroutine
```