

Numeričko rješavanje problema u elektrostatici pomoću programskog jezika Julia

Biršić, Tin

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: <https://urn.nsk.hr/um:nbn:hr:217:409361>

Rights / Prava: [In copyright/Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-03-24**



Repository / Repozitorij:

[Repository of the Faculty of Science - University of Zagreb](#)



SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Tin Biršić

NUMERIČKO RJEŠAVANJE PROBLEMA U
ELEKTROSTATICI POMOĆU PROGRAMSKOG
JEZIKA JULIA

Diplomski rad

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

SMJER: NASTAVNIK FIZIKE I INFORMATIKE

Tin Biršić

Diplomski rad

Numeričko rješavanje problema u
elektrostatici pomoću programskog
jezika julia

Voditelj diplomskog rada: prof. dr. sc. Davor Horvatić

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2024.

Michael Faraday jedan je od najvećih znanstvenika u povijesti. Unatoč tome što nije imao formalno obrazovanje jer mu roditelji nisu mogli pružiti tu priliku, postigao je nevjerojatne stvari. Zamislite samo što bi sve postigao i kako bi svijet danas izgledao samo da su mu roditelji bili Marina i Saša Biršić.
Hvala vam na svemu.

Sažetak

Tradicionalno, problemi iz područja elektrostatike rješavaju se analitičkim metodama. Međutim, s porastom složenosti sustava, analitička rješenja postaju nedostupna ili previše kompleksna za izvođenje. U takvim situacijama, numeričke metode postaju neophodan alat za analizu električnih polja i potencijala. Ovaj rad prikazuje primjenu numeričkih metoda u rješavanju problema iz elektrostatike korištenjem programskog jezika Julia. Korištenjem metode momenata (MoM, eng. *Method of Moments*) i metode konačnih elemenata (FDM, eng. *Finite Difference Method*) prikazali smo mogućnost rješavanja Laplaceove jednadžbe numeričkim metodama koje su nam omogućile da dobijemo aproksimacije potencijala i raspodjele naboja tamo gdje analitička rješenja postaju suviše kompleksna. Implementacija u Juliji pokazala se efikasnom i brzom za izračune te vizualizaciju rezultata, što potvrđuje vrijednost Julije kao alata u znanstvenim i tehničkim područjima.

Ključne riječi: Elektrostatika, numeričke metode, Laplaceova jednadžba, Julia, metoda momenata, metoda konačnih elemenata

Numerical Problem Solving in Electrostatics with the Julia Programming Language

Abstract

Traditionally, problems in the field of electrostatics are solved using analytical methods. However, as the complexity of systems increases, analytical solutions become unavailable or too complex to carry out. In such situations, numerical methods become an essential tool for the analysis of electric fields and potentials. This work presents the application of numerical methods in solving electrostatic problems using the Julia programming language. By utilizing the Method of Moments (MoM) and the Finite Difference Method (FDM), we demonstrated the capability to solve Laplace's equation with numerical methods that allowed us to obtain approximations of potentials and charge distributions where analytical solutions become too complex. The implementation in Julia proved to be efficient and fast for computations and visualization of results, confirming the value of Julia as a tool in scientific and technical fields.

Keywords: Electrostatics, numerical methods, Laplace's equation, Julia, method of moments, finite difference method

Sadržaj

1	Uvod	1
2	Elektrostatika	2
2.1	Električni naboј	2
2.2	Električna sila i Coulombov zakon	3
2.3	Električno polje	4
2.4	Gaussov zakon	6
2.5	Električni potencijal	8
2.6	Poissonova i Laplaceova jednadžba	9
3	Julia	11
3.1	Povijest i razvoj	11
3.2	Paketi	11
3.2.1	Instalacija paketa	12
3.2.2	LinearAlgebra	12
3.2.3	Plots	13
3.3	Specifičnosti jezika	15
3.3.1	Višestruko distribuiranje (eng. <i>Multiple Dispatch</i>)	15
3.3.2	Dinamički tipiziran sustav (eng. <i>Dynamic typed system</i>)	16
3.3.3	Pokretanje koda iz drugih jezika	16
4	Numeričke metode	17
4.1	Metoda Momenata	17
4.1.1	Definiranje elektroda i pod-elektroda	19
4.1.2	Generiranje matrice L	22
4.1.3	Riješavanje sustava	23
4.1.4	Prikaz rezultata	24
4.2	Metoda konačnih elemenata	25
4.2.1	Definiranje mreže	26
4.2.2	Potencijal na čvorovima	26
4.2.3	Sustav jednadžbi	27
4.2.4	Rješavanje sustava	30
4.2.5	Prikaz rezultata	32
4.2.6	Usporedba analitičkog i numeričkog rješenja	33
5	Zaključak	36

1 Uvod

Tradicionalno, problemi iz područja elektrostatike rješavaju se analitičkim metodama. Međutim, s porastom složenosti sustava, analitička rješenja postaju ili nedostupna ili previše komplikirana za izvođenje. Takve situacije gotovo su neizbjježne zbog kompleksnog matematičkog alata koji se koristi u elektrostatici pa numeričke metode postaju neophodan alat za analizu električnih polja i potencijala. Numeričke metode koje koristimo zamijenit će kompleksne jednadžbe s jednostavnijima ali po cijenu povećavanja broja samih jednadžbi koje moramo riješit. Koliko god nam to olakšalo matematički alat, zbog same količine jednadžbi, rješenje nam ne postaje dostupnije. Na sreću, količina jednadžbi koja ljudima predstavljaju sate i dane, računalu predstavljaju stotinke sekunde. Ovaj rad prikazuje primjenu numeričkih metoda u rješavanju problema iz elektrostatike korištenjem programskog jezika Julia.

U prvom dijelu rada predstavljeni su osnovni principi elektrostatike. Električni naboј i njegova svojstva, električna sila, električno polje i potencijal. Također, obradit ćemo ključne jednadžbe, izvesti ih i diskutirati njihov fizikalni smisao. Uvod u elektrostatiku završavamo s Poissonovom i Laplaceovom jednadžbom koje ćemo dalje u radu numerički rješavati.

Drugi dio rada fokusira se na programski jezik Julia. Relativno nov, izuzetno moćan programski jezik, koji je posebno razvijen za visoke performanse u znanstvenim i tehničkim područjima. Julia nudi jedinstvenu kombinaciju brzine, koja je usporediva s tradicionalnim jezicima poput C-a, i jednostavnosti sintakse karakteristične za jezike poput Pythona. Ukratko ćemo pokazati njegovu povijest, razvoj i mogućnosti, s posebnim naglaskom na prednosti koje nudi u znanstvenim izračunima. Pokazat ćemo na primjerima korištenje raznih paketa u kojima leži dio snaga Julije.

Glavni dio rada detaljno se bavi metodama za numeričko rješavanje problema u elektrostatici, pri čemu su naglasak doble metoda momenata (MoM) i metoda konačnih elemenata (FDM). Ove metode predstavljaju moćne alate za analizu elektrostatike, posebice kada se radi o složenim sustavima koji se ne mogu riješiti analitičkim putem. Cilj je pokazati kako se ove numeričke metode mogu koristiti za učinkovitu diskretizaciju prostora i rješavanje problema koji uključuju interakcije električnih naboja i polja. Prvi problem koji ćemo rješavati je distribucija naboja na metalnoj ploči na način da ćemo ploču podijeliti na N malih dijelova i na taj način ćemo problem diskretizirati i svesti na rješavanje sustava linearnih jednadžbi. U drugom problemu tražit ćemo oblik potencijala u prostoru tako što ćemo prostor podijeliti na N točaka u kojima ćemo tražiti potencijal. Rješenje ćemo također proširiti na sve točke prostora i usporediti s analitičkim rješenjem.

2 Elektrostatika

Generalno govoreći, električne i magnetske pojave ne promatramo zasebno jer promjena jedne pojave uzrokuje drugu i obratno. U sustavima u kojima nema vremenskih promjena električnog i magnetskog polja, odnosno kada su sve vremenske derivacije jednake nuli, možemo razdvojiti električne i magnetske pojave u dvije nezavisne kategorije.

$$Uvjet elektrostatike: \nabla \times \vec{E} = \vec{0} \quad (2.1)$$

U tom kontekstu, elektrostatika se bavi proučavanjem svojstava sustava s prostorno odvojenim, statičnim pozitivnim i negativnim električnim nabojima, pri čemu je cijeli sustav neutralan u smislu ukupnog naboja. U ovom radu bavit ćemo se rješavanjem upravo takvih sustava.

2.1 Električni naboј

Električni naboј je fundamentalna fizikalna veličina koja opisuje svojstvo čestica koje uzrokuju električno polje i električnu silu. Električni naboј ima tri osnovna svojstva.

I. Dva tipa naboja

Poznajemo dva tipa naboja, koje nazivamo "pozitivni" i "negativni" jer se njihovi učinci međusobno poništavaju. Naboji zapravo nisu ni pozitivni ni negativni, oni su izvori i ponori električnog polja no mi smo im dodijelili pozitivan i negativan predznak jer nam to matematički lako i intuitivno objašnjava da ako u nekoj točki prostora imamo pozitivan i negativan naboј to je isto kao da naboј i nema.

II. Naboј je očuvan

Sav naboј koji sada postoji, oduvijek je postojao. Dakle, ukupni naboј u svemiru je konstantan. Ovo se naziva **globalna očuvanost naboja**. Također vrijedi i **lokalna očuvanost naboja** koja znači da naboji ne mogu prijeći s jednog mesta na drugo bez nekog kontinuiranog puta što globalna očuvanost naboja dopušta jer se ukupni naboј i dalje nije promijenio. Iz ovog svojstva slijedi **jednadžba kontinuiteta**

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \vec{J} = 0 \quad (2.2)$$

III. Naboј je kvantiziran

Naboј dolazi u cjelobrojnim višekratnicima elementarnog naboja $e \approx 1.602 \times 10^{-19} \text{ C}$ pa je tako naboј elektrona $-e$, protona $+e$ i nikada nismo našli česticu naboja npr. $\frac{1}{2}e$ ili nešto slično. S obzirom na to da je elementarni naboј izrazito mala jedinica, često je u praktičnim slučajevima prihvatljivo zanemariti kvantizaciju u potpunosti i tretirati naboј kao kontinuiranu veličinu kao što je

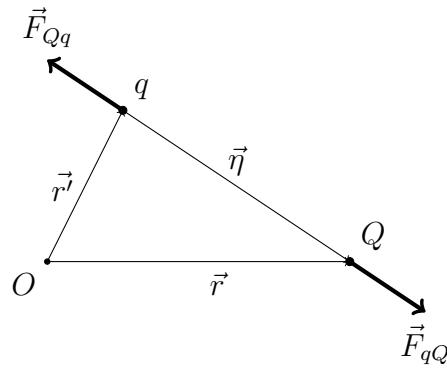
u praktičnim slučajevima prihvatljivo gledati na vodu u cijevi kao kontinuiranu tekućinu iako znamo da je to zapravo skup diskretnih jedinica (molekula).

2.2 Električna sila i Coulombov zakon

Električna sila jedna je od temeljnih sila u prirodi, odgovorna za interakcije između električki nabijenih čestica. Često ju zovemo i "kulonova" ili u širom kontekstu "elektromagnetska" sila. Temeljni zakon koji opisuje električnu силу između dvije točkaste čestice je Coulombov zakon

$$\vec{F}(\vec{r}) = \frac{1}{4\pi\epsilon_0} \frac{qQ}{\eta^2} \hat{\eta} \quad (2.3)$$

Gdje je $\vec{\eta}$ vektor koji pokazuje od naboja q koji se nalazi u točki na koju pokazuje \vec{r}'



Slika 2.1: Skica dva istoimena naboja s vektorima sile

do testnog naboja Q koji se nalazi u točki na koju pokazuje \vec{r} .

$$\vec{\eta} = \vec{r} - \vec{r}' \quad (2.4)$$

Električna sila djeluje u smjeru $\hat{\eta}$ i može biti privlačna ili odbojna ovisno o predznaku naboja

Konstanta ϵ_0 zove se permitivnost vakuuma i iznosi $\epsilon_0 = 8.85 \times 10^{-12} \frac{C^2}{Nm^2}$. Za druge linearne, izotropne i homogene materijale, permitivnost ϵ je dana izrazom $\epsilon = k\epsilon_0$, gdje je k relativna permitivnost materijala.

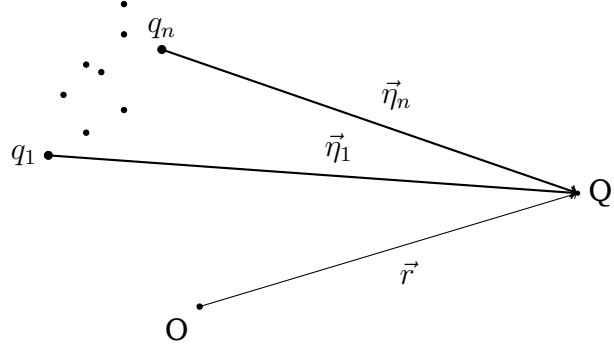
Sila kojom testni nabo Q djeluje na nabo q mora biti, po III. Newtonovom zakonu suprotne orijentacije, ali jednakog iznosa sili kojom nabo q djeluje na testni nabo Q . Iz definicije vektora \vec{r} i \vec{r}' vidimo da će u tom slučaju \vec{r} pokazivati na nabo q jer je to točka gdje je promatrano silu, a vektor \vec{r}' pokazuje na testni nabo Q jer je taj nabo u ovom slučaju "izvor" sile.

$$\vec{\eta} = \vec{r}' - \vec{r} = -\vec{\eta} \quad (2.5)$$

$$\vec{F}(\vec{r}') = -\frac{1}{4\pi\epsilon_0} \frac{qQ}{\eta^2} \hat{\eta} = -\vec{F}(\vec{r}) \quad (2.6)$$

2.3 Električno polje

Za električnu silu vrijedi princip superpozicije pa silu kojom N naboja djeluje na testni naboj Q možemo zapisati kao sumu sila svakog pojedinog naboja q_i na testni naboj Q



Slika 2.2: Skica N naboja koji djeluju na testni naboj Q

$$\vec{F} = \sum_{i=1}^N \vec{F}_i \quad (2.7)$$

$$\vec{F} = \frac{Q}{4\pi\epsilon_0} \left(\frac{q_1}{\eta_1^2} \hat{\eta}_1 + \frac{q_2}{\eta_2^2} \hat{\eta}_2 + \cdots + \frac{q_n}{\eta_n^2} \hat{\eta}_n \right) \quad (2.8)$$

Definiramo novu veličinu koja će ovisiti samo o nabojima q - **električno polje**

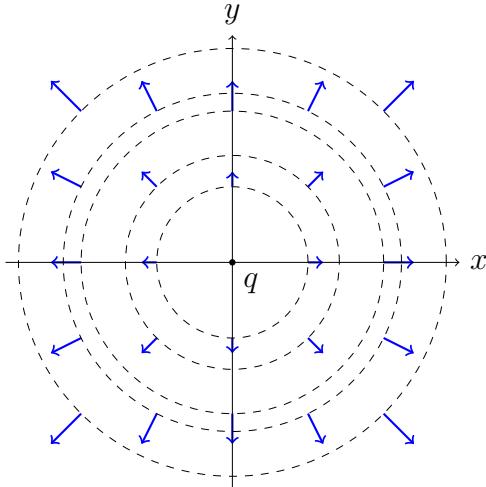
$$\vec{E} = \frac{1}{Q} \vec{F} \quad (2.9)$$

$$\vec{E}(\vec{r}) \equiv \frac{1}{4\pi\epsilon_0} \sum_{i=1}^N \frac{q_i}{\eta_i^2} \hat{\eta}_i \quad (2.10)$$

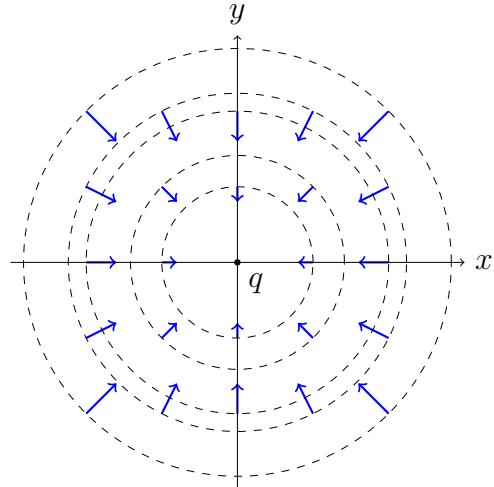
Primjer električnog polja za jedan naboj q u ishodištu $\vec{r}' = \vec{0}$:

$$\vec{r}' = \vec{0} \rightarrow \vec{\eta} = \vec{r} \quad (2.11)$$

$$\vec{E}(\vec{r}) = \frac{1}{4\pi\epsilon_0} \frac{q}{r^2} \hat{r} \quad (2.12)$$



Slika 2.3: $q > 0$

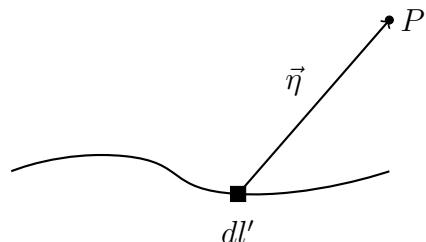


Slika 2.4: $q < 0$

Iznos električnog polja \vec{E} je obrnuto proporcionalan s kvadratom udaljenosti te je konstantan na kružnici radijusa r . Orientaciju određuje predznak naboja q . Nacr-tamo li graf za nekoliko veličina r možemo vidjeti izgled električnog polja oko jednog naboja q . Uočimo da kada je naboј pozitivan tada vektori električnog polja \vec{E} pokazuju od njega, a kada je naboј negativan tada vektori električnog polja \vec{E} pokazuju prema njemu.

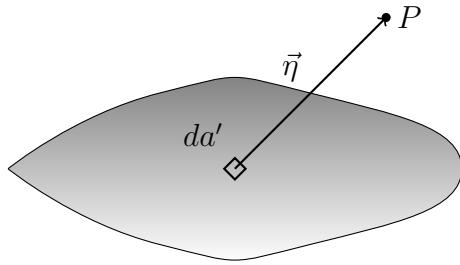
Ovdje uvodimo aproksimaciju da je u nekom volumenu konfiguracija naboja kontinuirana i uvodimo **gustoću naboja**. To nam omogućuje da iz diskretnog oblika, koji je operativno neupotrebljiv zbog velikog broja naboja, pređemo u integralni oblik. Ovisno o problemu, nekada možemo zanemariti neke dimenzije pa definiramo tri gustoće naboja. Linijska λ , površinska σ i volumna ρ gustoća naboja. Pomoću gustoća naboja možemo napisati infinitezimalni dio naboja dq i tako prijeći iz integrala po dq u integral po dl , da ili $d\tau$.

Linijski (1D):



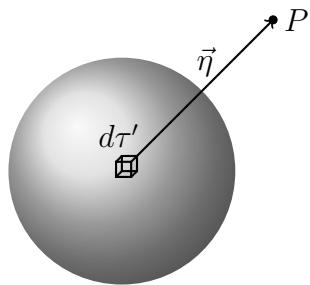
$$\vec{E}(\vec{r}) = \frac{1}{4\pi\epsilon_0} \int \frac{\lambda(\vec{r}')}{\eta^2} \hat{\eta} dl' \quad (2.13)$$

Površinski (2D):



$$\vec{E}(\vec{r}) = \frac{1}{4\pi\epsilon_0} \iint_S \frac{\sigma(\vec{r}')}{\eta^2} \hat{n} da' \quad (2.14)$$

Volumni (3D):



$$\vec{E}(\vec{r}) = \frac{1}{4\pi\epsilon_0} \iiint_V \frac{\rho(\vec{r}')}{\eta^2} \hat{n} d\tau' \quad (2.15)$$

Iz ovih izraza vidimo stvarnu fizikalnu definiciju vrsta naboja iz svojstva 1. Jedna vrsta naboja je **izvor** električnog polja i toj vrsti smo podijelili pozitivan predznak, a druga vrsta **ponor** električnog polja i toj vrsti smo podijelili negativan predznak.

2.4 Gaussov zakon

Gaussov zakon, jedan od temeljnih zakona elektrodinamike, predstavlja ključnu komponentu Maxwellovih jednadžbi i opisuje vezu između toka električnog polja i obuhvaćeno električnog naboja. Krenimo od primjera 2.3 i promotrimo tok električnog polja kroz sferu radijusa R za jedan naboju u ishodištu. Tok vektorskog polja¹ predstavlja ukupnu količinu “protoka” vektorskih veličina kroz danu površinu. Definiramo ga kao integral normalne komponente vektorskog polja kroz zadalu površinu

$$\Phi = \int_S \vec{V} \cdot d\vec{a} \quad (2.16)$$

¹U našem slučaju radi se o električnom polju ali tok i Gaussov teorem, iako su našli na najveću primjenu u elektrodinamici, svakako nisu jedini primjer upotrebe.

Trenutno nas ne zanima bilo kakvo vektorsko polje već električno polje čiji oblik znamo

$$\Phi_E = \oint_S \left(\frac{1}{4\pi\epsilon_0} \frac{q}{R^2} \hat{r} \right) \cdot d\vec{a} \quad (2.17)$$

Uvrstimo standardni izraz za infinitezimalni dio površine $d\vec{a}$ u sfernem koordinatnom sustavu:

$$\Phi_E = \oint_S \left(\frac{1}{4\pi\epsilon_0} \frac{q}{R^2} \hat{r} \right) \cdot (R^2 \sin \theta d\theta d\phi \hat{r}) \quad (2.18)$$

$$\Phi_E = \frac{q}{4\pi\epsilon_0} \oint_S \sin \theta d\theta d\phi = \frac{q}{4\pi\epsilon_0} 4\pi \quad (2.19)$$

$$\oint_S \vec{E} \cdot d\vec{a} = \frac{q}{\epsilon_0} \quad (2.20)$$

Izraz (2.20) zovemo **integralnim oblikom Gaussovog zakona**

Ukupni tok električnog polja kroz zatvorenu površinu S proporcionalan je obuhvaćenom naboju q .

Primjer: Tok električnog polja za N obuhvaćenih naboja

Kao na slici 2.3 zamislimo da imamo N naboja koje smo obuhvatili nekom površinom S .

Za električno polje vrijedi princip superpozicije:

$$\vec{E}_{uk} = \sum_{i=1}^N \vec{E}_i \quad (2.21)$$

Tok ukupnog električnog polja \vec{E}_{uk} glasi:

$$\oint_S \vec{E}_{uk} \cdot d\vec{a} = \sum_{i=1}^N \oint_S \vec{E}_i \cdot d\vec{a} \quad (2.22)$$

$$\oint_S \vec{E}_{uk} \cdot d\vec{a} = \sum_{i=1}^N \frac{q_i}{\epsilon_0} \quad (2.23)$$

$$\oint_S \vec{E}_{uk} \cdot d\vec{a} = \frac{Q_{ENC}}{\epsilon_0} \quad (2.24)$$

Došli smo do pune definicije integralnog oblika Gaussovog zakona. Ukupni tok električnog polja kroz zatvorenu površinu S proporcionalan je ukupnom obuhvaćenom naboju Q_{ENC} .

Ukupni naboј unutar zatvorene površine Q_{ENC} predstavlja ključnu veličinu u Gaussovom zakonu, koji kvantitativno opisuje ponašanje električnog polja. Iako ovaj zakon ne sadrži nove informacije u odnosu na Coulombov zakon i princip superpozicije, njegova je primjena izrazito moćna. Ključna značajka Gaussovog zakona leži u ponašanju električnog polja kao funkcije udaljenosti $\frac{1}{r^2}$ jer bez ovog odnosa ne bi došlo do ključnog kraćenja radiusa, a sam tok električnog polja ovisio bi o specifičnom obliku površine, a ne samo o količini naboja unutar nje. Ovakva analiza ima

široku primjenu, a slične zaključke možemo donijeti i za zakone kao što je Newtonov zakon gravitacije, koji također slijedi vlastiti oblik Gaussovog zakona.

Iz integralnog oblika lagano možemo preći u **diferencijalni oblik** korištenjem Gaussovog teorema koji kaže da je ukupni tok vektorskog polja kroz zatvorenu površinu S jednak zbroju svih izvora i ponora tog polja u volumenu omeđenom površinom S .

$$\oint_S \vec{V} \cdot d\vec{a} = \iiint_V (\nabla \cdot \vec{V}) d\tau \quad (2.25)$$

Koristeći definiciju volumne gustoće naboja ρQ_{ENC} možemo zapisati kao:

$$Q_{ENC} = \iiint_V \rho(\vec{r}') d\tau' \quad (2.26)$$

Uvrštavanjem u integralni oblik Gaussovog zakona dobijemo izraz:

$$\oint_S \vec{E} \cdot d\vec{a} = \frac{1}{\epsilon_0} \iiint_V \rho(\vec{r}') d\tau \quad (2.27)$$

$$\iiint_V (\nabla \cdot \vec{V}) d\tau = \frac{1}{\epsilon_0} \iiint_V \rho(\vec{r}') d\tau \quad (2.28)$$

$$\nabla \cdot \vec{V} = \frac{\rho(\vec{r}')}{\epsilon_0} \quad (2.29)$$

2.5 Električni potencijal

Električno polje \vec{E} nije bilo kakvo vektorsko polje. U elektrostatici električno polje je vektorsko polje kojem je rotacija jednaka nuli².

$$\nabla \times \vec{E} = \vec{0} \quad (2.30)$$

Posljedica toga je da vektorski problem pronalaženja električnog polja \vec{E} možemo pojednostaviti na skalarni problem. U tu svrhu koristit ćemo Helmholtzov teorem koji kaže da svako vektorsko polje, čija je rotacija jednaka nuli, možemo zapisati kao gradijent neke skalarne funkcije.³

$$\nabla \times \vec{E} = \vec{0} \Leftrightarrow \vec{E} = -\nabla V \quad (2.31)$$

Iz tog izraza proizlaze još nekoliko korisnih izraza:

$$\text{Integral } \int_{\vec{a}}^{\vec{b}} \vec{E} \cdot d\vec{l} \text{ je neovisan o putu} \quad (2.32)$$

$$\oint \vec{E} \cdot d\vec{l} = 0 \quad (2.33)$$

$$\text{Skalarna funkcija } V \text{ nije jedinstvena.} \quad (2.34)$$

Definiramo električni potencijal kao skalarnu funkciju oblika:

$$V(\vec{r}) \equiv - \int_o^{\vec{r}} \vec{E}(\vec{r}') \cdot d\vec{l} \quad (2.35)$$

²Rotacija vektorskog polja je vektor pa je preciznije reći nulvektor $\vec{0}$ ali jednostavnosti radi kažemo nula

³Minus predznak ovdje nije definiran matematikom već činjenicom da smo pozitivne naboje definirali kao izvor električnog polja

I. Ekvipotencijalna ploha

Možemo definirati plohu na kojoj je potencijal konstantan. Tu plohu zovemo **ekvipotencijalna ploha** ili samo **ekvipotencijala**

II. Za potencijal vrijedi princip superpozicije

$$V(\vec{r}) = \sum_{i=1}^N V_n(\vec{r}) \quad (2.36)$$

III. Potencijal je neovisan o ishodištu

Iz definicije potencijala vidimo da je to linijski integral od ishodišta do neke točke \vec{r} ali ishodište može biti bilo gdje. Matematički je to vidljivo kada probamo pomaknuti ishodište.

$$V'(\vec{r}) = - \int_{O'}^{\vec{r}} \vec{E} \cdot d\vec{l} \quad (2.37)$$

$$V'(\vec{r}) = - \int_{O'}^O \vec{E} \cdot d\vec{l} - \int_O^{\vec{r}} \vec{E} \cdot d\vec{l} \quad (2.38)$$

$$V'(\vec{r}) = K + V(\vec{r}) \quad (2.39)$$

Zato je potencijal u jednoj točki "besmislen".

Ako je potencijal u jednoj točki "besmislen", ono što nije besmisленo je **razlika potencijala - napon**

$$V(\vec{b}) - V(\vec{a}) = - \int_0^{\vec{b}} \vec{E} \cdot d\vec{l} + \int_0^{\vec{a}} \vec{E} \cdot d\vec{l} \quad (2.40)$$

$$= - \int_0^{\vec{b}} \vec{E} \cdot d\vec{l} - \int_{\vec{a}}^0 \vec{E} \cdot d\vec{l} \quad (2.41)$$

$$V(\vec{b}) - V(\vec{a}) = - \int_{\vec{a}}^{\vec{b}} \vec{E} \cdot d\vec{l} \quad (2.42)$$

Razlika potencijala je linijski integral koji ne ovisi o putu već samo o početnoj i krajnjoj točki.

2.6 Poissonova i Laplaceova jednadžba

Primijenimo li divergenciju na (2.31)⁴ dobit ćemo jednadžbu koju zovemo **Poissonova jednadžba**

$$\nabla \times (-\nabla V) = \frac{\rho}{\epsilon_0} \quad (2.43)$$

$$\nabla^2 V = -\frac{\rho}{\epsilon_0} \quad (2.44)$$

⁴Zapravo uvrštavamo (2.31) u Gaussov zakon

Operator ∇^2 zovemo Laplace operator ili Laplasjan.

Poissonova jednadžba je parcijalna diferencijalna jednadžba če rješenje daje **potencijal za zadanu gustoću naboja**.

$$V(\vec{r}) = \frac{1}{4\pi\epsilon_0} \iiint_V \frac{\rho(\vec{r}')}{\eta} d\tau' \quad (2.45)$$

Ako u području rješavanja jednadžbe nema gustoće naboja, tada govorimo o **Lapla-ceovoj jednadžbi**

$$\nabla^2 V = 0 \quad (2.46)$$

$$\frac{\partial^2}{\partial x^2} V(x, y, z) + \frac{\partial^2}{\partial y^2} V(x, y, z) + \frac{\partial^2}{\partial z^2} V(x, y, z) = 0 \quad (2.47)$$

U 1D slučaju, jednadžba se svodi na:

$$\frac{d^2}{dx^2} V(x) = 0 \quad (2.48)$$

Da bi riješili ovu jednadžbu potencijal V moramo integrirati po x dva puta. Funkcija koju dobijemo bit će oblika:

$$V(x) = C_1 x + C_2 \quad (2.49)$$

C_1 i C_2 su konstante koje su određene rubnim uvjetima. Za primjer možemo uzeti rubne uvijete:

$$V(x_1) = 0, V(x_2) = V_{max} \quad (2.50)$$

$$(2.51)$$

To nam daje sustav dvije jednadžbe s dvije nepoznanice čije rješavanje će nam dati vrijednost C_1 i C_2

$$C_1 x_1 + C_2 = 0 \quad (2.52)$$

$$C_1 x_2 + C_2 = V_{max} \quad (2.53)$$

Međutim, u 2D i 3D sustavima jednadžbe postaju sve složenije za rješavanje do te mjere da, u odnosu na probleme koji se ne mogu analitički riješiti, postoji vrlo ograničen broj problema za koje je analitičko rješenje moguće pronaći. U takvima situacijama, preostaje nam primjena različitih numeričkih metoda.



3 Julia

U ovom radu koristit ćemo programski jezik Julia. Relativno nov, ali izuzetno moćan programski jezik, koji je posebno razvijen za visoke performanse u znanstvenim i tehničkim područjima. Julia nudi jedinstvenu kombinaciju brzine, koja je usporediva s tradicionalnim jezicima poput C-a, i jednostavnosti sintakse karakteristične za jezike poput Pythona. Njena glavna prednost leži u mogućnosti pisanja lako čitljivog koda koji se može izvršavati s visokim performansama, što je od presudne važnosti za rješavanje zahtjevnih numeričkih problema.

3.1 Povijest i razvoj

Julia je prvi put predstavljena 2012. godine od strane tima istraživača s MIT-a, kojeg su činili Jeff Bezanson, Stefan Karpinski, Viral B. Shah i Alan Edelman. Cilj ovog tima bio je stvoriti jezik koji bi eliminirao potrebu za korištenjem više različitih programskih jezika u jednom projektu, što je čest slučaj u znanstvenim istraživanjima. Julia je izvorno razvijena s fokusom na visoke performanse i lakoću upotrebe, što ju je učinilo pogodnom za matematičke izračune, analizu podataka i algoritme strojnog učenja.

3.2 Paketi

U programiranju općenito, paketi su zbirke funkcija, tipova podataka i drugih resursa organiziranih tako da pružaju specifične funkcionalnosti korisnicima. U Jeziku Julia, paketi igraju ključnu ulogu jer omogućuju proširenje osnovnih funkcionalnosti jezika. Julia dolazi s relativno malom standardnom bibliotekom, ali njena snaga leži u bogatom i lako dostupnom *ekosustavu* paketa koji omogućuju rješavanje različitih problema. Od analize podataka preko strojnog učenja pa sve do optimizacije i vizualizacije. Paketi omogućuju korisnicima da koriste već napisane i optimizirane alate, umjesto da svaki put iznova pišu funkcionalnosti za specifične zadatke. Takva praksa se čak smatra lošom i potiče se korištenje paketa gdje god je to moguće. Neke od prednosti paketa su:

- I. **Modularnost** - Umjesto da programski jezik dolazi s velikim brojem ugrađenih funkcionalnosti, koje možda nisu potrebne svakom korisniku, Julia omogućuje korisnicima da biraju (učitaju) samo one funkcionalnosti koje su im potrebne. To omogućuje brže pokretanje programa i smanjuje opterećenje na sustav.

- II. **Recikliranje koda** - Korištenje paketa štedi vrijeme i resurse. Umjesto da korisnici pišu vlastite implementacije za često korištene zadatke (npr. rad s tabličnim podacima, vizualizacija itd.), mogu koristiti već napisane, provjerene i optimizirane funkcije.
- III. **Specijalizacija** - Mnogi paketi u Juliji su specijalizirani za određene zadatke. Na primjer, `DifferentialEquations.jl` je specifično dizajniran za rješavanje diferencijalnih jednadžbi, dok je `Flux.jl` usmjeren na modelе strojnog učenja. Ovakva specijalizacija omogućuje korisnicima da se fokusiraju na svoj problem, bez potrebe za razumijevanjem svih detalja algoritma koje koriste kao alat u rješavanju problema.
- IV. **Zajednica i podrška** - Julia ima aktivnu zajednicu koja konstantno razvija nove pakete. Ovi paketi često dolaze s bogatom dokumentacijom, primjerima i podrškom, što korisnicima olakšava učenje i implementaciju novih funkcionalnosti.

3.2.1 Instalacija paketa

Instalacija paketa u Juliji je jednostavna i intuitivna, zahvaljujući ugrađenom sustavu za upravljanje paketima `Pkg`. Za instalaciju paketa potrebno je otvoriti Julia REPL (eng. Read-Eval-Print Loop) i upisati sljedeće naredbe:

```
using Pkg  
Pkg.add("NazivPaketa")
```

3.2.2 LinearAlgebra

Paket `LinearAlgebra` pruža osnovne alate linearne algebre, kao što su matrice, vektori i razne operacije nad njima. Julia ima ugrađenu podršku za većinu funkcija linearne algebre, ali paket `LinearAlgebra` omogućuje još jednostavniji rad s matricama, vektorima i operacijama poput množenja, faktorizacije i rješavanja sustava linearnih jednadžbi.

Paket se jednostavno koristi uključivanjem naredbom:

```
using LinearAlgebra
```

Nakon čega je omogućena upotreba funkcija poput operatora "`\`" za rješavanje linearnih sustava jednadžbi. Na primjer, za rješavanje sustava $Ax = b$, gdje je A matrica koeficijenata, a b vektor slobodnih članova, možemo napisati:

```
A = [2 1; 3 4]  
b = [5; 6]  
x = A \ b
```

Ovaj kod će izračunati vektor x koji zadovoljava sustav jednadžbi.

3.2.3 Plots

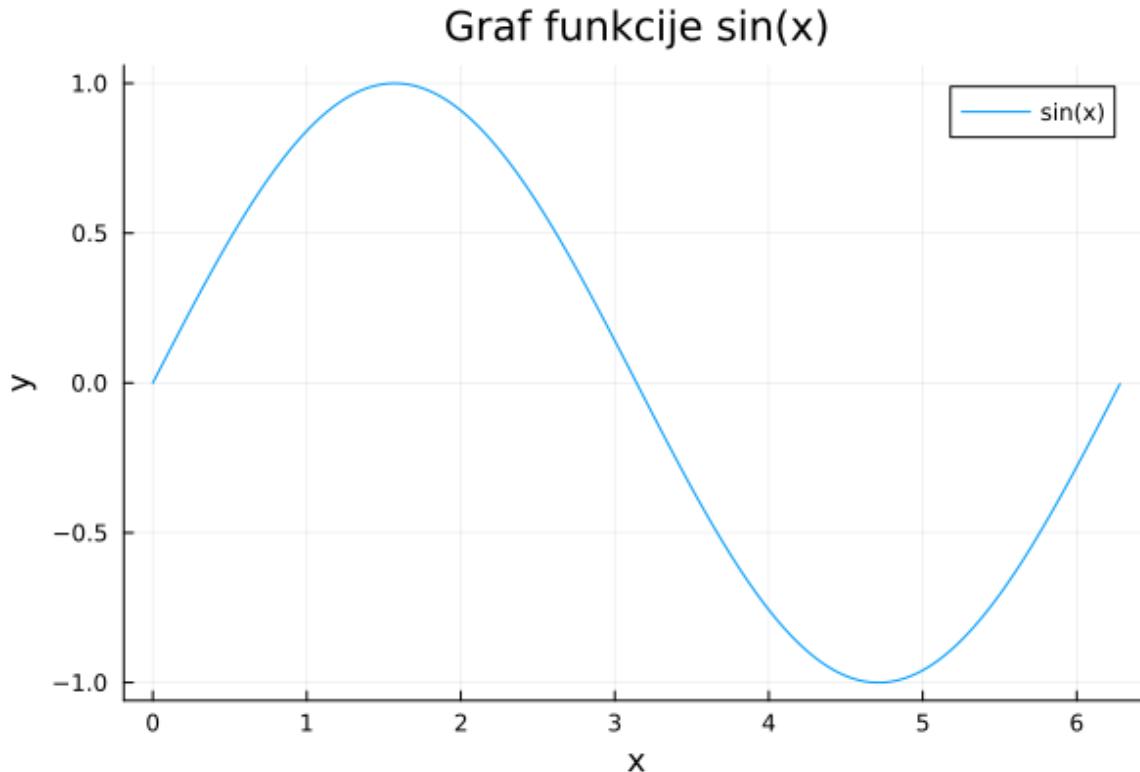
Paket *Plots* je snažan alat za vizualizaciju podataka u programskom jeziku Julia. Omogućuje korisnicima jednostavno i intuitivno stvaranje grafova i drugih vizualizacija. Uključujemo ga naredbom:

```
using Plots
```

Crtanje funkcije ($y = \sin(x)$) na intervalu od 0 do 2π :

```
x = 0:0.01:2pi  
y = sin.(x)  
plot(x, y, label="sin(x)", xlabel="x", ylabel="y",  
      title="Graf funkcije sin(x)")
```

Ovaj kod će generirati graf sinusne funkcije s označenim osima i naslovom.

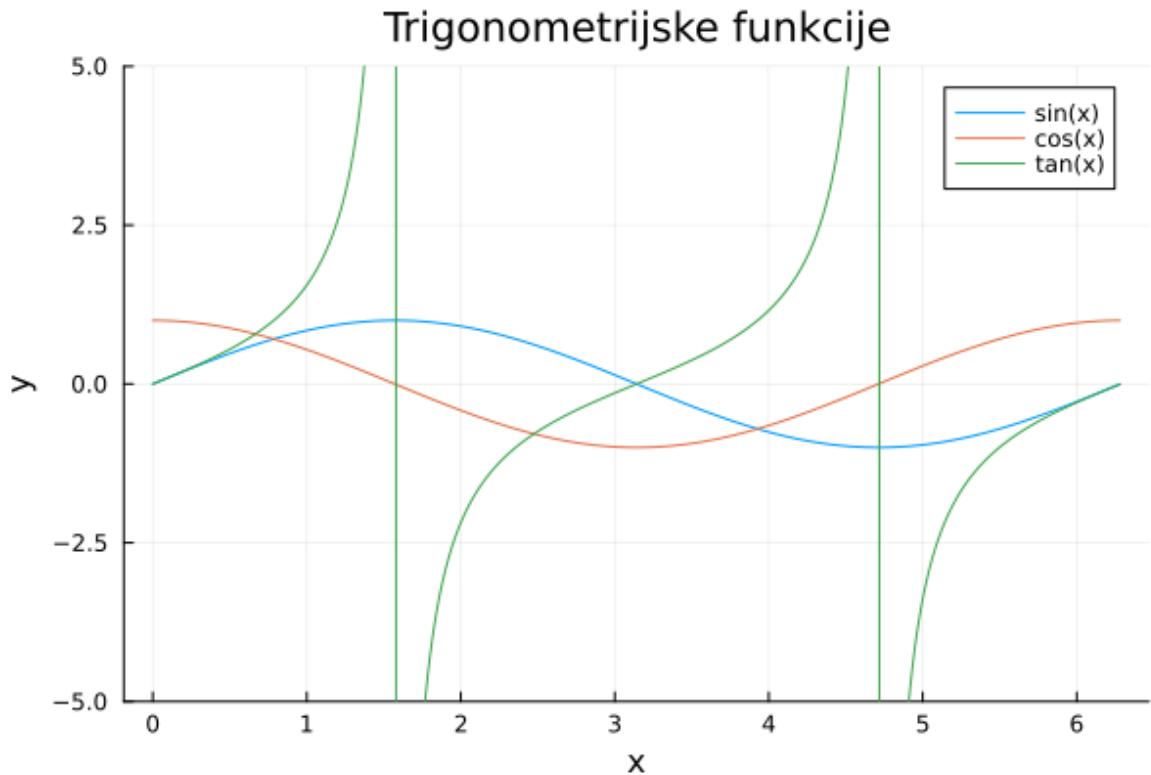


Za usporedbu više funkcija na istom grafu, možemo ih jednostavno dodavati

```
x = 0:0.01:2pi  
plot(x, sin.(x), label="sin(x)")  
plot!(x, cos.(x), label="cos(x)")  
plot!(x, tan.(x), label="tan(x)", ylim=(-5, 5))  
title!("Trigonometrijske funkcije")
```

```
xlabel!("x")
ylabel!("y")
```

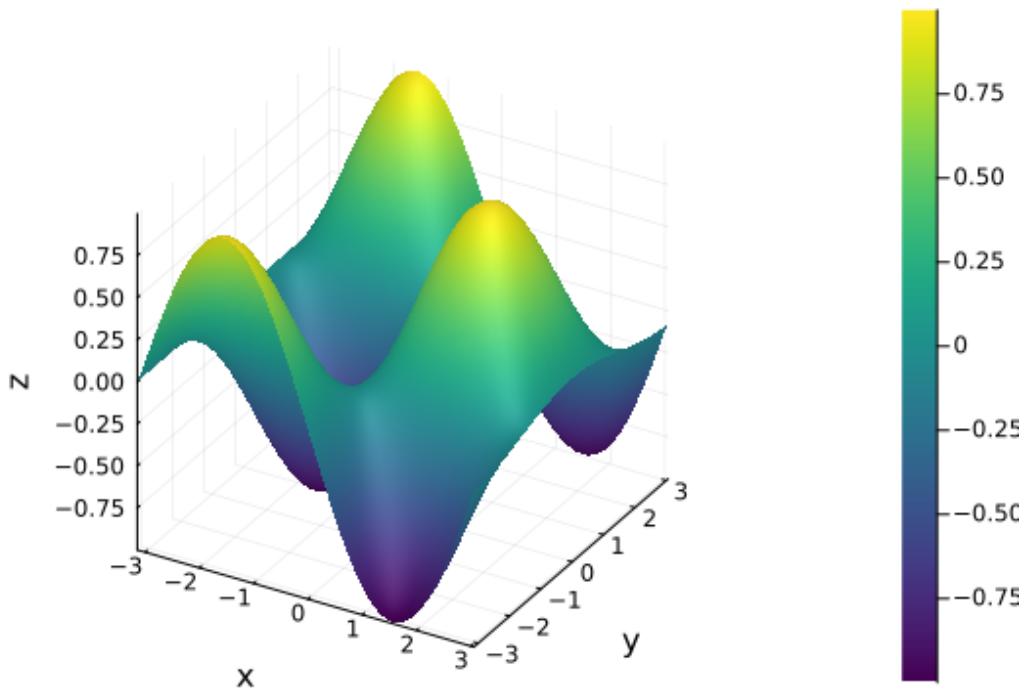
Ovdje funkcija `plot!` dodaje nove krivulje na postojeći graf. Također smo ograničili y-os na interval $[-5, 5]$ kako bi prikaz $\tan(x)$ bio pregledniji.



Plots podržava i crtanje 3D grafova. Na primjer, za prikaz funkcije $z = \sin(x) \cdot \cos(y)$:

```
x = y = -pi:0.1:pi
surface(x, y, (x, y) -> sin(x) * cos(y),
        xlabel="x", ylabel="y", zlabel="z",
        color=:viridis)
```

Ovdje funkcija `surface` crta površinu. Također z -os automatski označava bojom. `color` argument određuje mapu boja (end. Color Map). To su unaprijed definirani nizovi boja koji se koriste za prikazivanje vrijednosti na grafu. *Viridis* mapa boja je perceptualno uniformna. Prelazi iz tamno ljubičaste preko plave i zelene do žute boje. Dizajnirana je tako da bude lako čitljiva i u crno-bijelim ispisima te za osobe s poteškoćama u percepciji boja.



3.3 Specifičnosti jezika

3.3.1 Višestruko distribuiranje (eng. *Multiple Dispatch*)

Jedna od ključnih karakteristika Julije je podrška za multiple dispatch funkcija. To znači da izbor funkcije koja će se pozvati ovisi o tipu svih njezinih argumenata, a ne samo o prvom ili eksplisitnom tipu. Ovo omogućuje pisanje vrlo generičnog i učinkovitog koda.

Definirajmo funkciju koja zbraja dva argumenta:

```
function add(a, b)
    a + b
end
```

Julia automatski generira optimalnu verziju funkcije za kombinacije tipova koje se koriste. Međutim, možemo definirati različite implementacije za različite tipove:

```
function add(a::Int, b::Int)
    println("Zbrajam dva cijela broja.")
    a + b
end

function add(a::Float64, b::Float64)
    println("Zbrajam dva realna broja.")
    a + b
end
```

```
end
```

Poziv funkcije će sada odabrat odgovarajuću verziju ovisno o tipovima argumenta.

3.3.2 Dinamički tipiziran sustav (eng. *Dynamic typed system*)

Julia je *dynamic typed system*, što znači da tipovi varijabli mogu biti određeni u vrijeme izvođenja. Međutim, tipovi su *strong*, što sprječava neželjene konverzije i potencijalne pogreške. Možemo definirati varijablu bez eksplisitnog navođenja tipa:

```
x = 10
typeof(x) # Int64
```

Tip varijable *x* je automatski određen na temelju dodijeljene vrijednosti. Ako pokušamo zbrojiti nespojive tipove, Julia će generirati pogrešku:

```
x = 10
y = "20"
x + y # Greška: Metoda nije definirana za tipove Int64 i String
```

3.3.3 Pokretanje koda iz drugih jezika

Julia nudi jednostavno pokretanje postojećeg koda iz jezika kao što su C i Python, što omogućuje korištenje postojećeg koda i biblioteka iz tih jezika.

Julia se ističe nizom jedinstvenih značajki koje je čine moćnim alatom. U ovom poglavlju istaknuli smo neke od specifičnosti jezika Julia. Međutim, ovo su samo neke od karakteristika koje Julia nudi. Jezik kontinuirano evoluira uz doprinos aktivne zajednice, donoseći nove mogućnosti i poboljšanja. Zbog svoje fleksibilnosti, performansi i čitljivosti koda, Julia predstavlja izuzetno perspektivan izbor za razvoj visokoučinkovitih aplikacija u različitim područjima znanosti i tehnike.

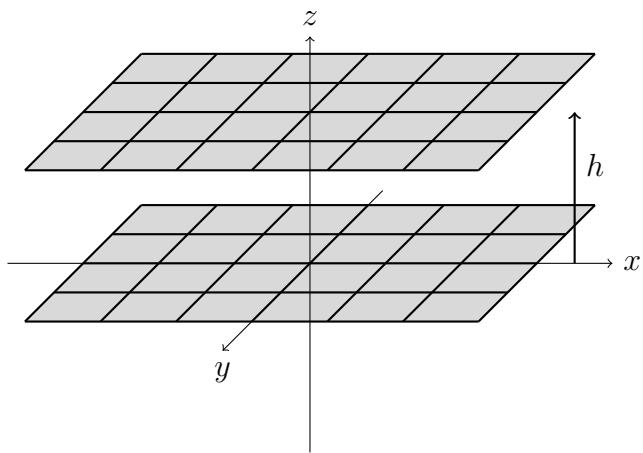
4 Numeričke metode

Numeričke metode predstavljaju ključni alat za rješavanje složenih matematičkih i fizičkih problema gdje analitička rješenja nisu izvediva. U kontekstu elektrostatike, mnogi problemi koji uključuju složene geometrije ili raspodjеле naboja zahtijevaju pristup koji se oslanja na numeričke metode. U ovom poglavlju fokusirat ćemo se na primjenu numeričkih metoda za rješavanje elektrostatičkih problema, kao što su rješavanje Laplaceove jednadžbe. Cilj je demonstrirati kako se pomoću Julije mogu efikasno implementirati ove metode, te analizirati dobivene rezultate.

4.1 Metoda Momenata

Metoda momenata (MoM, eng. *Method of Moments*) je numerička metoda rješavanja integralnih jednadžbi pretvaranjem ih u sustav linearnih jednadžbi. Ključ ove metode leži u diskretizaciji kontinuiranog problema na konačan broj elemenata.

Zamislimo neograničeni prostor⁵ u kojem se nalazi neka površina idealnih vodiča. Svaki vodič postavljen je na određeni potencijal V u odnosu na neki referentni potencijal V_r , što znači da na njima postoji raspodjela električnog naboja. U slučaju tankih vodičkih ploča (elektroda), raspodjela naboja je dvodimenzionalna i koncentrirana na samoj površini tih elektroda. Ako je sustav bio neutralan prije primjene potencijala, on će i ostati neutralan, pri čemu se naboј premješta s jedne elektrode na drugu kako bi se podržali primjenjeni potencijali. Naš zadatak je, uz poznatu geometriju i napone, odrediti raspodjelu naboja na elektrodama. Za primjer uzeli dvije paralelne elektrode u $X - Y$ ravnini razmaknute za h kao što je prikazano na slici 4.1



Slika 4.1: Dvije paralelne elektrode u razmagnute za h

Elektrode smo podijelili na N ne nužno jednakih dijelova površine A (pod-elektrode).

⁵U praksi, neograničen prostor znači da vodiči koji nisu predmet naše analize smatramo da su dovoljno daleko da njihov utjecaj možemo zanemariti.

Ako pretpostavimo da je površinska gustoća naboja σ na svakom dijelu uniformna možemo ju izraziti kao:

$$\sigma_i = \frac{Q_i}{A_1} \quad (4.1)$$

Gdje je Q_i ukupni naboј na i-toj pod-elektrodi.

Potencijal na i-toj pod-elektrodi je suma doprinosa svih pod-elektroda uključujući i samu i-tu pod-elektrodu. Primjenjujući izraz za potencijal uz danu gustoću naboja (2.45) u središtu pod-elektrode u odnosu na neki referentni potencijal V_r dobit ćemo izraz:

$$V_i - V_r = \sum_j \frac{Q_j}{4\pi\varepsilon_0 A_j} \iint \frac{dx dy}{\sqrt{(x_{0,i} - x_j)^2 + (y_{0,i} - y_j)^2 + h^2}} \quad (4.2)$$

S obzirom na to da smo pretpostavili da je površinska gustoću naboja σ_i konstanta, možemo ju izbaciti iz integracije i zamijeniti s njenom definicijom iz (4.1). Primjetimo da su sve veličine osim naboja ili fizikalne konstante ili geometrijske varijable. To nam omogućuje da definiramo koeficijent $L_{i,j}$ i preko njega izrazimo potencijal V_i

$$L_{i,j} = \frac{1}{4\pi\varepsilon_0 A_j} \iint \frac{dx dy}{\sqrt{(x_{0,i} - x_j)^2 + (y_{0,i} - y_j)^2 + h^2}} \quad (4.3)$$

$$V_i = \sum_j L_{i,j} Q_j \quad (4.4)$$

Sada imamo sustav od N jednadžbi koje opisuju potencijal u središtu svake pod-elektrode u terminima naboja na pod-elektrodi i koeficijenta $L_{i,j}$.

$$\begin{bmatrix} V_1 - V_r \\ V_2 - V_r \\ \vdots \\ V_n - V_r \end{bmatrix} = \begin{bmatrix} L_{1,1} & L_{1,2} & \cdots & L_{1,n} \\ L_{2,1} & L_{2,2} & \cdots & L_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ L_{n,1} & L_{n,2} & \cdots & L_{n,n} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_n \end{bmatrix} \quad (4.5)$$

Na primjeru dvije jednake elektrode sustavi bi izgledao ovako:

$$\begin{bmatrix} V_1 - V_r \\ V_2 - V_r \end{bmatrix} = \begin{bmatrix} L_{1,1} & L_{1,2} \\ L_{2,1} & L_{2,2} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \quad (4.6)$$

No ovaj sustav nije generalno rješenje jer ne sadržava informaciju o neutralnosti sustava. Dodavanjem toga uvjeta i preraspodjelom jednadžbi dolazimo do generalnog sustava jednadžbi na primjeru dvije elektrode:

$$\begin{bmatrix} V_1 \\ V_2 \\ 0 \end{bmatrix} = \begin{bmatrix} L_{1,1} & L_{1,2} & 1 \\ L_{2,1} & L_{2,2} & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ V_r \end{bmatrix} \quad (4.7)$$

Sada kada smo definirali set jednadžbi, moramo za svaku pod-elektrodu izračunati koeficijent $L_{i,j}$. Izraz (4.3) nije pogodan za rješavanje ovog problema jer, koliko god numeričko integriranje u Juliji bilo brzo i efikasno, nije *scalable*. Broj izvrednjavanja

integrala brzo poraste na $\approx 10^3$. Trebamo se poslužiti nekom aproksimacijom koja će biti brza i točna.

$$L_{i,j} \approx \frac{1}{4\pi\epsilon_0} \left[\frac{1}{r + \left\{ L_A \left[1 + \left(\frac{r}{a} \right) \right] \right\}^{-1}} \right] \quad (4.8)$$

$$r = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \quad (4.9)$$

Gdje je L_a :

$$L_A = \frac{1}{4\pi\epsilon_0} \frac{1}{4ab} \int_{-b}^b \int_{-a}^a \frac{dx dy}{\sqrt{x^2 + y^2}} \quad (4.10)$$

Nakon rješavanja ovog integrala dobijemo generalni izraz:

$$L_A(a, b) = \frac{1}{4\pi\epsilon_0} \left(\frac{1}{b} \ln \frac{b + \sqrt{a^2 + b^2}}{a} - \frac{1}{a} \ln \frac{-a + \sqrt{a^2 + b^2}}{b} \right) \quad (4.11)$$

Gdje su nam a i b stranice pod-elektroda. Ova jednadžba nema nikakvu fizikalnu osnovu već je samo jako dobra aproksimacija za (4.3). Time smo izbjegli izvrednjavanje izraza (4.3) mnogo puta već smo ga koristili samo kako bi došli do izraza (4.8) kojeg ćemo onda zapravo koristiti u stvarnom rješenju.

4.1.1 Definiranje elektroda i pod-elektroda

Na početku je važno napomenuti da se ovaj dio mogao izvesti jednostavnije, ili čak u potpunosti izbjegći. Da smo *hardkodirali* rezultat ove skripte, elektrode bi bile unaprijed definirane. Međutim, ovim pristupom ostavljamo otvorenu mogućnost korištenja CAD⁶ datoteka za definiranje elektroda u budućnosti. U tom slučaju, sve što bi bilo potrebno jest izraditi skriptu, odnosno svojevrsni *interface*, koja bi CAD datoteku pretvorilo u tablicu za opis elektroda. Ovim pristupom širimo spektar primjene, no izrada tog dijela nadilazi opseg ovog rada. Stoga ćemo zasad definirati protokol za definiranje elektroda u obliku tablice, koju ćemo ručno sastaviti. Tu tablicu smo definirali na sljedeći način.

	a					
X_0	Y_0	N_x	N_y	h	V	

⁶Computer-Aided Design

a	Polu-razmak između pod-elektroda
X_0	Centar elektrode u x smjeru
Y_0	Centar elektrode u y smjeru
N_x	Broj dijelova na koji će se elektroda podijeliti u x smjeru
N_y	Broj dijelova na koji će se elektroda podijeliti u y smjeru
h	Visina elektrode u odnosu na $X - Y$ ravninu
V	Potencijal na elektrodi

Za naš konkretan primjer dvije elektrode u $X - Y$ ravnini razmagnute za $h = 1$ od koje je svaka na potencijalu $|V| = 0.5$ i podijeljena na 2500 jednakih pod-elektroda tablica izgleda ovako:

0.01							
0	0	50	50	0	0.5	0	
0	0	50	50	1	-0.5	0	

Funkcija `generate_electrode_values_from_table` prima kao argumente vrijednost a te listu n -torki, od kojih svaka n -torka predstavlja jedan red u tablici (tj. jednu pod-elektrodu). Rezultat funkcije su koordinate elektroda te pripadajući potencijali u formatu koji će kasnije biti proslijeđen funkciji `lfit`. Budući da je a isti za sve n -torke, on se proslijeđuje funkciji kao zasebna varijabla.

```

a = 0.01
electrode_table = [
    (0.0, 0.0, 50, 50, 0, 0.5, 0),
    (0.0, 0.0, 50, 50, 1, -0.5, 0)
]

```

```

function generate_electrode_values_from_table(a, electrode_table)
    sub_electrodes = []
    subElectrodes_volts = []

    for (x0, y0, nx, ny, h, V) in electrode_table
        x_ll = x0 - a * nx
        y_ll = y0 - a * ny

        for j in 1:nx
            xj = x_ll - a + j * 2 * a
            for k in 1:ny
                yj = y_ll - a + k * 2 * a
                line = [xj, yj, h]
                push!(sub_electrodes, line)
                push!(subElectrodes_volts, V)
            end
        end
    end

    sub_electrodes = hcat(sub_electrodes...) |> permutedims
    subElectrodes_volts = vcat(subElectrodes_volts...)

    return sub_electrodes, subElectrodes_volts
end

```

U prvom dijelu funkcije početne koordinate svake pod-elektrode se korigiraju pomoću parametara N_x , N_y i a . Zatim se računa položaj svake pojedine pod-elektrode unutar same elektrode te se za svaku generira red s koordinatama (x, y, h) te se dodaje odgovarajući potencijal V . Koordinate pod-elektroda i odgovarajući naponi, organiziraju se u dvodimenzionalno polje za pod-elektrode i vektor⁷ za potencijale. Ova funkcija je ključna za generiranje diskretizirane mreže pod-elektroda na ravnini, uzimajući u obzir geometrijske i električne karakteristike pod-elektroda.

⁷Mislimo na *vektor* u Juliji. Specifičan podatkovni tip koji predstavlja jednodimenzionalno polje

4.1.2 Generiranje matrice L

```
function l_fit(a, sub_electrodes)
    sub_electrodes = Matrix(sub_electrodes)
    eps0 = 8.854e-12
    xi = sub_electrodes[:, 1]
    yi = sub_electrodes[:, 2]
    zi = sub_electrodes[:, 3]

    n = length(xi)
    xs = (repeat(xi, 1, n) - repeat(xi', n, 1)).^2
    ys = (repeat(yi, 1, n) - repeat(yi', n, 1)).^2
    zs = (repeat(zi, 1, n) - repeat(zi', n, 1)).^2

    p = 1.414 * a
    La = (log((a + p) / a) / a - log((-a + p) / a) / a)

    r = sqrt.(xs + ys + zs)
    L = r .+ 1 ./ La ./ (1 .+ r / a)
    L = 1.0 ./ L / (4 * pi * eps0)

    return L
end
```

Funkcija *l_fit* koristi se za izračunavanje elemenata $L_{i,j}$ i generiranje matrice L . Ona prima dva parametra: *sub_electrodes* i *a*. *sub_electrodes* je matrica s prostornim koordinatama pod-elekroda dobivena kao rezultat funkcije *generate_electrode_values_from_table*. Konačna matrica L računa se na temelju aproksimacije (4.8).

4.1.3 Rješavanje sustava

```
eps0 = 8.854e-12
sub_electrodes, subElectrodes_volts =
    electrode.generate_electrode_values_from_table(a, electrode_table)

l = l_value.l_fit(a, sub_electrodes)

nr_vars = size(l, 1)
new_col = fill(l[1, 1], nr_vars)
l = hcat(l, new_col)
new_row = vcat(new_col, 0)
l = vcat(l, new_row')

subElectrodes_volts = vcat(subElectrodes_volts, 0)
```

Nakon što smo dobili $sub_electrodes$, $subElectrodes_volts$ i l potrebno je dodati matricama novi red i stupac kako bi uzeli u obzir referentni potencijal V_r i neutralnost sustava. Na kraju rješavamo sustav jednadžbi $q = L/V$

```
q = l \ subElectrodes_volts
```

U programskom jeziku Julia, operator " \backslash " služi za rješavanje linearnih sustava jednadžbi. Primjerice, kada želimo riješiti sustav $l \cdot q = subElectrodes_volts$ s obzirom na vektor q , ovaj operator nam omogućuje upravo to. Njegova snaga leži u tome što automatski odabire najprikladniju metodu rješavanja ovisno o obliku matrice l , što je često skriveno od korisnika. Julia interni koristi različite algoritme na temelju strukture matrice:

- Kvadratna matrica: Julia primjenjuje LU faktorizaciju ili slične direktnе metode za učinkovito rješavanje sustava.
- Predeterminiran sustav (više redaka nego stupaca): Kada matrica l ima više redaka nego stupaca, sustav je predeterminiran. U tom slučaju, Julia koristi metodu najmanjih kvadrata, često putem QR dekompozicije ili sličnih pristupa, kako bi minimizirala kvadratnu razliku između $l \cdot q$ i $subElectrodes_volts$.
- Nedeterminiran sustav (više stupaca nego redaka): Ako matrica l ima više stupaca nego redaka, sustav je nedeterminiran s beskonačno mnogo mogućih rješenja. Julia tada odabire ono rješenje q koje ima najmanju Euklidsku normu, odnosno najmanju duljinu među svim mogućim rješenjima.

Ovaj automatizirani pristup omogućuje korisnicima da jednostavno riješe složene linearne sustave bez potrebe za ručnim odabirom najprikladnije metode.

4.1.4 Prikaz rezultata

```
function plot_charge_distribution(electrode, q)

    e = 1.602176634e-19
    q_norm = q / e

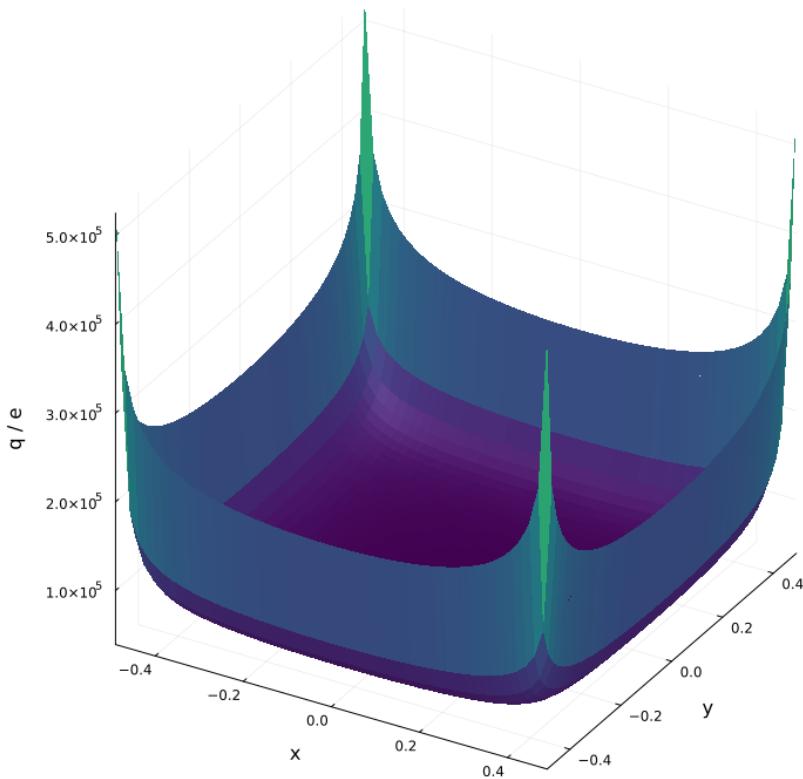
    x = electrode[:, 1]
    y = electrode[:, 2]

    nx = length(unique(x))
    ny = length(unique(y))

    x_grid = reshape(x, nx, ny)
    y_grid = reshape(y, nx, ny)
    q_grid = reshape(q_norm, nx, ny)

    surface(x_grid, y_grid, q_grid,
             xlabel = "x", ylabel = "y", zlabel = "q / e",
             size=(1000,800),
             color = :viridis)
end
```

Za prikaz distribucije naboja koristit ćemo *plot_charge_distribution* funkciju koja koristi podatke o prostornim koordinatama pod-elektroda i normiranu raspodjelu naboja kako bi generirala trodimenzionalni grafički prikaz raspodjele naboja na elektrodi. Koordinate i vrijednosti naboja preoblikuju se u odgovarajuće dvodimenzionalne mreže (*grids*), koje služe za stvaranje 3D površinskog grafa.



Slika 4.2: Distribucija naboja na elektrodi

Zbog visoke mobilnosti naboja na elektrodi, naboji se koncentriraju na rubovima, s obzirom da na rubovima nema dovoljno suprotnog naboja koji bi ih balansirao. Ovaj fenomen je posebno izražen u vrhovima gdje nedostatak odbojnog naboja djeluje u dvije dimenzije, za razliku od situacije uzduž ravnog dijela ruba gdje djeluje u jednoj dimenziji. U ovom primjeru, vrhovi elektroda formiraju pravi kut, što dodatno povećava koncentraciju naboja u tim točkama.

4.2 Metoda konačnih elemenata

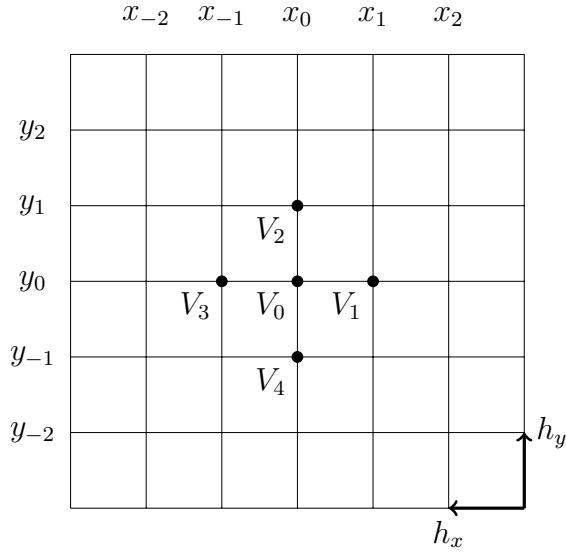
Metoda konačnih elemenata (FDM, eng. *Finite Difference Method*) predstavlja jednu od najmoćnijih i najšire primjenjivanih numeričkih metoda za rješavanje parcijalnih diferencijalnih jednadžbi. Primjenjuje se u širokom rasponu znanstvenih i inženjerskih područja. Osnovni princip metode konačnih elemenata temelji se na diskretizaciji prostora na manja, jednostavnija područja, unutar kojih se rješenje aproksimira pomoću jednostavnih funkcija. Ova metoda omogućava analizu složenih geometrija gdje analitička rješenja nisu dostupna, te pruža visoku fleksibilnost u modeliranju rubnih i početnih uvjeta.

U ovom primjeru bavit ćemo se rješavanjem Laplaceove jednadžbe (2.46) u 2D:

$$\nabla^2 V(x, y) = \frac{\partial^2}{\partial x^2} V(x, y) + \frac{\partial^2}{\partial y^2} V(x, y) = 0 \quad (4.12)$$

4.2.1 Definiranje mreže

Zamislimo 2D mrežu.



Promatramo sve susjedne točke u okolini (x_0, y_0) . Točke su uniformno raspoređene i razmaknute za h_1 u x smjeru i h_1 u y smjeru.

$$h_x = x_{i+1} - x_i \quad (4.13)$$

$$h_y = y_{i+1} - y_i \quad (4.14)$$

Možemo pretpostaviti da su točke razmaknute u x smjeru koliko i u y smjeru.

$$h_x = h_y \equiv h \quad (4.15)$$

4.2.2 Potencijal na čvorovima

Sada možemo raspisati potencijale u terminima h te ih razviti u Taylorov red

$$V_0 \equiv V_{0,0} = V(x_0, y_0) \quad (4.16)$$

$$V_1 \equiv V_{1,0} = V(x_0 + h, y_0) \quad (4.17)$$

$$V_2 \equiv V_{0,1} = V(x_0, y_0 + h) \quad (4.18)$$

$$V_3 \equiv V_{-1,0} = V(x_0 - h, y_0) \quad (4.19)$$

$$V_4 \equiv V_{-1,-1} = V(x_0 - h, y_0 - h) \quad (4.20)$$

$$V_1 = V(x_0, y_0) + h \frac{\partial}{\partial x} V(x_0, y_0) + \frac{h^2}{2} \frac{\partial^2}{\partial x^2} V(x_0, y_0) + \frac{h^3}{6} \frac{\partial^3}{\partial x^3} V(x_0, y_0) + \dots \quad (4.21)$$

$$V_2 = V(x_0, y_0) + h \frac{\partial}{\partial y} V(x_0, y_0) + \frac{h^2}{2} \frac{\partial^2}{\partial y^2} V(x_0, y_0) + \frac{h^3}{6} \frac{\partial^3}{\partial y^3} V(x_0, y_0) + \dots \quad (4.22)$$

$$V_3 = V(x_0, y_0) - h \frac{\partial}{\partial x} V(x_0, y_0) + \frac{h^2}{2} \frac{\partial^2}{\partial x^2} V(x_0, y_0) - \frac{h^3}{6} \frac{\partial^3}{\partial x^3} V(x_0, y_0) + \dots \quad (4.23)$$

$$V_4 = V(x_0, y_0) - h \frac{\partial}{\partial y} V(x_0, y_0) + \frac{h^2}{2} \frac{\partial^2}{\partial y^2} V(x_0, y_0) - \frac{h^3}{6} \frac{\partial^3}{\partial y^3} V(x_0, y_0) + \dots \quad (4.24)$$

Zbrojimo li pripadne jednadžbe vidimo da se svi članovi koji sadrže derivacije neparnog reda poništavaju

$$V_1 + V_3 = 2V_0 + h^2 \frac{\partial^2}{\partial x^2} V_0 + \frac{h^4}{12} \frac{\partial^4}{\partial x^4} V_0 + \dots \quad (4.25)$$

$$V_2 + V_4 = 2V_0 + h^2 \frac{\partial^2}{\partial y^2} V_0 + \frac{h^4}{12} \frac{\partial^4}{\partial y^4} V_0 + \dots \quad (4.26)$$

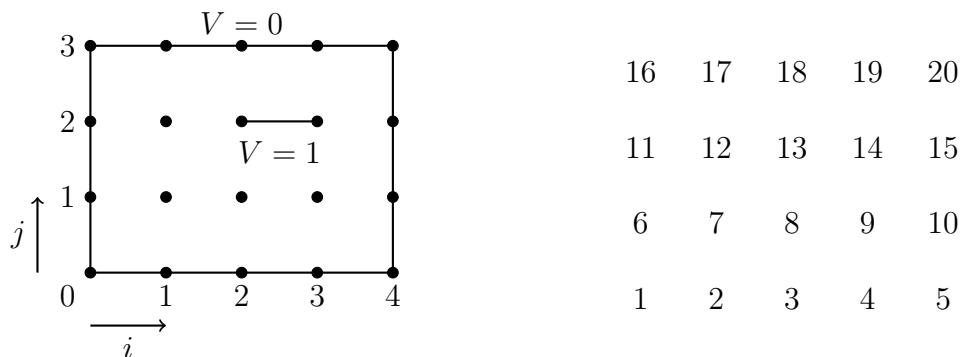
Zbrojimo li ove dvije jednadžbe i preuređimo ih dolazimo do aproksimacije Laplace-ove jednadžbe u točki (x_0, y_0)

$$\frac{\partial^2 V_0}{\partial x^2} + \frac{\partial^2 V_0}{\partial y^2} = \frac{V_1 + V_3 - 2V_0}{h^2} + \frac{V_2 + V_4 - 2V_0}{h^2} = 0 \quad (4.27)$$

$$V_1 + V_2 + V_3 + V_4 + 4V_0 = 0 \quad (4.28)$$

$$V_0 = \frac{V_1 + V_2 + V_3 + V_4}{4} \quad (4.29)$$

4.2.3 Sustav jednadžbi



Radi jednostavnosti i bolje preglednosti, mrežu i pripadajuće jednadžbe prikazat ćemo koristeći primjer s malim brojem čvorova. Ovaj pristup omogućuje nam da lakše objasnimo osnovne koncepte bez dodatne složenosti koju bi veća mreža donijela. Na slici 4.2.3 prikazana je mreža čvorova zajedno s definiranim rubnim uvjetima. Rubovi mreže imaju potencijal $V = 0$, što znači da su na referentnom potencijalu. Dodatno, dva unutarnja čvora postavljeni su na potencijal $V = 1$. Ovi potencijali predstavljaju naše početne uvjete i utječu na raspodjelu potencijala kroz cijelu mrežu. U ovom primjeru primjećujemo da imamo više čvorova s unaprijed definiranim potencijalima nego onih čiji su potencijali nepoznati. Čvorove s

unaprijed definiranim potencijalima nazivamo vezanim čvorovima jer su njihova stanja fiksirana i ne mijenjaju se tijekom analize. Oni služe kao poznate vrijednosti u našem sustavu jednadžbi. Nasuprot tome, čvorovi čiji su potencijali nepoznati nazivamo slobodnim čvorovima. To su čvorovi za koje trebamo izračunati potencijale rješavajući sustav jednadžbi koji opisuje ponašanje mreže. Broj slobodnih čvorova određuje veličinu našeg sustava jednadžbi i složenost problema. Razlog zbog kojeg u ovom primjeru imamo više vezanih nego slobodnih čvorova je taj što smo odabrali vrlo malu mrežu radi jednostavnosti. U praksi, kada povećamo veličinu mreže, broj slobodnih čvorova značajno raste, a broj vezanih čvorova obično ostaje isti ili se povećava mnogo sporije. To znači da u većim mrežama imamo mnogo više potencijala koje trebamo izračunati. Na slici 4.2.3 prikazan je način na koji pretvaramo notaciju potencijala s dvostrukim indeksom $V_{i,j}$ u jednostruku notaciju V_n . U $V_{i,j}$ notaciji, potencijali su označeni prema svojim pozicijama na mreži, gdje i i j predstavljaju koordinatne indekse čvora. Međutim, prilikom rješavanja sustava jednadžbi često je praktičnije koristiti jednostruku notaciju V_n , gdje je svaki čvor označen jedinstvenim brojem n . Korištenje jednostrukih notacija omogućuje nam da jednadžbe izrazimo u matricama i vektorima, što je pogodno za numeričke metode. Numerički algoritmi, poput metoda za rješavanje sustava linearnih jednadžbi, često su optimizirani za rad s vektorima i matricama u jednostrukoj notaciji. Budući da je odabir numeriranja proizvoljan, možemo ga prilagoditi tako da optimiziramo performanse ili da olakšamo implementaciju posebnih rubnih uvjeta. Svaki vezani čvor postavit ćemo na vrijednost početnog uvjeta, a na svaki slobodni čvor primijenit ćemo izraz (4.28). Potencijal na slobodnom čvoru $\equiv V_0$ dok su $[V_1, V_4]$ potencijali na čvorovima u njegovoj okolini. Rješenje sustava jednadžbi dat će nam vrijednost potencijala u svakom čvoru. Za naš konkretni primjer sustav jednadžbi izgleda ovako:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \\ V_9 \\ V_{10} \\ V_{11} \\ V_{12} \\ V_{13} \\ V_{14} \\ V_{15} \\ V_{16} \\ V_{17} \\ V_{18} \\ V_{19} \\ V_{20} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

4.2.4 Rješavanje sustava

Imin = -20
Imax = 20
Jmin = -20
Jmax = 20

Prije svega, potrebno je definirati mrežu s kojom ćemo raditi. Na slici 4.2.3, parametri i i j su uvijek pozitivni cijeli brojevi, što znači da se čvor s koordinatama $(0, 0)$ nalazi u donjem lijevom kutu mreže. Ovakav način indeksiranja čvorova pojednostavljuje generiranje indeksa za jednostavne mreže. Međutim, kada radimo s kompleksnijim ili nepravilnim mrežama, ovaj pristup može postati ograničavajući i otežati proces generiranja indeksa čvorova. Kako bismo našu Julia skriptu učinili što univerzalnijom i prilagodljivijom različitim vrstama mreža, redefinirat ćemo koordinatni sustav tako da je čvor $(0, 0)$ smješten u središtu mreže. Ovo znači da parametri i i j više nisu nužno pozitivni.

U našem primjeru koristit ćemo mrežu dimenzija 20×20 .

Ic1 = -10
Ic2 = 10
Jc = 0

Definiramo čvorove koji će nam predstavljati krajnje točke dužine na kojoj je potencijal $V = 1$. Ovime smo definirali mrežu u terminima $V_{i,j}$.

```
function get_ijk(Imin, Imax, Jmin, i, j)
    return (i - Imin) + (j - Jmin) * (Imax - Imin + 1) + 1
end
```

Da bi jednostavno prelazili iz notacije s dvostrukim indeksom $V_{i,j}$ u jednostruku notaciju V_n i obrnuto potrebna nam je funkcija *get_ijk*.

```
Kmax = get_ijk(Imin, Imax, Jmin, Imax, Jmax)
```

```
a = spzeros(Kmax, Kmax)
b = zeros(Kmax)
```

Matricu a inicijaliziramo kao rijetku (*sparse*) matricu jer će većina njezinih elemenata biti nule, što znači da bi pohranjivanje svih tih nula u memoriju bilo neefikasno. Funkcija *spzeros* učinkovito koristi memoriju tako što pohranjuje samo indekse i vrijednosti elemenata koji nisu nula, dok nule implicitno tretira kao zadane vrijednosti. S druge strane, matrica b je toliko mala u smislu memorijskog prostora i nije nužno ispunjena uglavnom nulama, pa korištenje *spzeros* funkcije za nju ne bi donijelo značajne prednosti. Veličine matrica određene su indeksom posljednjeg čvora.

```
for j in Jmin:Jmax
    for i in Imin:Imax
        k0 = get_ijk(Imin, Imax, Jmin, i, j)
        a[k0, k0] = -4
        if k0 < Kmax
            a[k0, k0 + 1] = 1
        end
        if k0 > 1
            a[k0, k0 - 1] = 1
        end
        k1 = k0 + (Imax - Imin) + 1
        if k1 <= Kmax
            a[k0, k1] = 1
        end
        k2 = k0 - (Imax - Imin) - 1
        if k2 > 0
            a[k0, k2] = 1
        end
    end
end
```

Na početku, sve čvorove tretiramo kao slobodne, a zatim naknadno dodjeljujemo

vrijednosti vezanim čvorovima. Iako se to može činiti kao dodatni ili nepotrebni korak, važno je istaknuti da će u ovom primjeru (kao i u svim drugim *use casevima*) broj vezanih čvorova biti zanemarivo mali u usporedbi s brojem slobodnih čvorova.

```

for i in Imin:Imax
    k = get_ijk(Imin, Imax, Jmin, i, Jmin)
    a[k, :] .= 0
    a[k, k] = 1
end

for j in Jmin + 1:Jmax - 1
    k = get_ijk(Imin, Imax, Jmin, Imin, j)
    a[k, :] .= 0
    a[k, k] = 1
end

```

Prolazimo kroz sve čvorove na donjem i lijevom rubu mreže gdje je koordinata $j = J_{min}$ odnosno $i = I_{min}$. Redak k u matrici a postavlja se na nule, a zatim se dijagonalni element postavlja na 1. Time efektivno fiksiramo potencijal na tim čvorovima na $V = 0$. Ovo predstavlja rubni uvjet na donjem i lijevom rubu mreže.

```

for i in Ic1:Ic2
    k = get_ijk(Imin, Imax, Jmin, i, Jc)
    a[k, :] .= 0
    a[k, k] = 1
    b[k] = 1
end

```

Nakon toga prolazimo kroz određeni raspon čvorova unutar mreže gdje je koordinata $j = J_c$ i i varira od I_{c1} do I_{c2} . Redak k u matrici a postavlja se na nule, dijagonalni element postavlja se na 1, a odgovarajući element u vektoru b postavlja se na 1. Ovim postupkom fiksiramo potencijal na $V = 1$ za ove čvorove. To predstavlja unutarnju traku s potencijalom $V = 1$, što je u skladu s našim prethodnim definiranjem vezanih čvorova.

```
v = a \ b
```

Ovo je kulminacija našeg prethodnog rada gdje smo definirali mrežu, postavili rubne uvjete i konstruirali sustav jednadžbi. Rješavamo sustav linearnih jednadžbi odnosno, izračunavamo potencijale v na svim slobodnim čvorovima mreže.

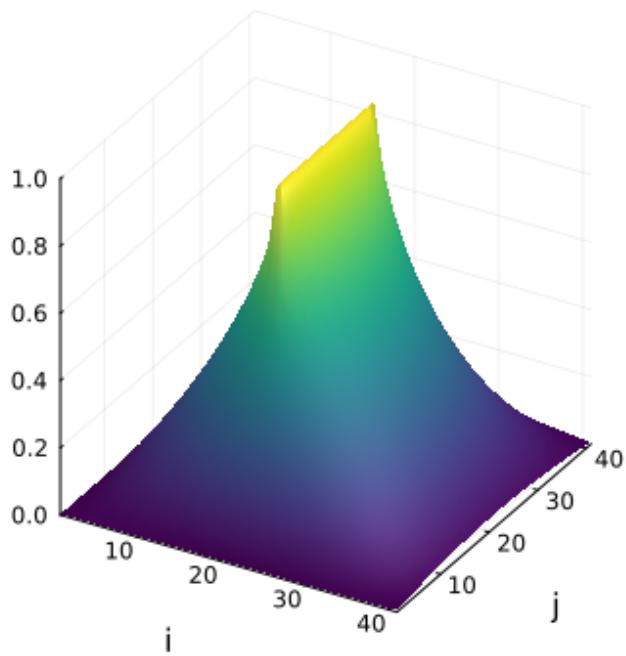
4.2.5 Prikaz rezultata

```
Sx = Imax - Imin + 1
Sy = Jmax - Jmin + 1
Volts = zeros(Sx, Sy)

for ii in Imin:Imax
    i = ii - Imin + 1
    for jj in Jmin:Jmax
        j = jj - Jmin + 1
        k = get_ijk(Imin, Imax, Jmin, ii, jj)
        Volts[i, j] = v[k]
    end
end

surface(Volts, xlabel="i", ylabel="j", color=:viridis)
```

Nakon što smo riješili sustav jednadžbi i dobili potencijal za sve čvorove, potrebno je te vrijednosti prebaciti natrag u jednostruku notaciju kako bismo mogli vizualizirati raspodjelu potencijala.



Slika 4.3: Potencijal na čvorovima

Primjena metode konačnih elemenata omogućila nam određivanje potencijala $V_{i,j}$ na čvorovima. Međutim, i dalje ne znamo potencijal u bilo kojoj točki prostora.

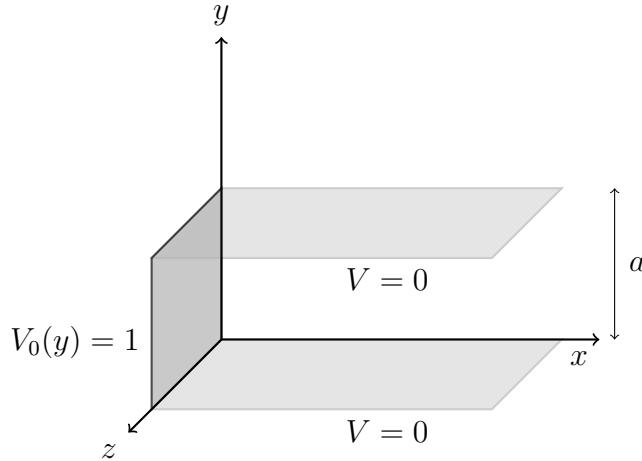
Kako bismo to dobili primijenit ćemo aproksimaciju koristeći linearu interpolaciju. Pretpostaviti ćemo da se funkcija ponaša linearno između čvorova što nam omogućuje procjenu vrijednosti funkcije na nepoznatim točkama unutar čvorova.

Važno je napomenuti da je numeričko rješenje aproksimacija izvedena isključivo za čvorove. Što znači da radimo aproksimaciju temeljnu na prethodnoj aproksimaciji, stoga ne bismo trebali imati prevelika očekivanja u pogledu njene preciznosti.

$$V(x, y) = V_{i,j} + (V_{i+1,j} - V_{i,j}) \frac{x}{h} + (V_{i,j+1} - V_{i,j}) \frac{y}{h} + (V_{i+1,j+1} - V_{i+1,j} - V_{i,j+1} + V_{i,j}) \frac{xy}{h^2} \quad (4.30)$$

4.2.6 Usporedba analitičkog i numeričkog rješenja

Zamislimo dvije uzemljene paralelne beskonačne metalne ploče u $X - Z$ ravnini razmaknute za a . Na lijevoj strani ($x = 0$) nalazi se beskonačna izolirana traka na potencijalu $V_0 = 1$. Zanima nas potencijal u prostoru između ploča.



Budući da u prostoru između ploča nema rasподјеле naboja, rješenje ovog problema svodi se na rješavanje Laplaceove jednadžbe (2.46). No, s obzirom na to da V nema ovisnost o z , jednadžba se svodi na dvodimenzionalni oblik:

$$\frac{\partial^2}{\partial x^2} V(x, y) + \frac{\partial^2}{\partial y^2} V(x, y) = 0 \quad (4.31)$$

Rješenje jednadžbe određeno je rubnim uvjetima:

- $y = 0 \rightarrow V = 0$
- $y = a \rightarrow V = 0$
- $x = 0 \rightarrow V = V_0$
- $V \rightarrow 0$ kada $x \rightarrow \infty$

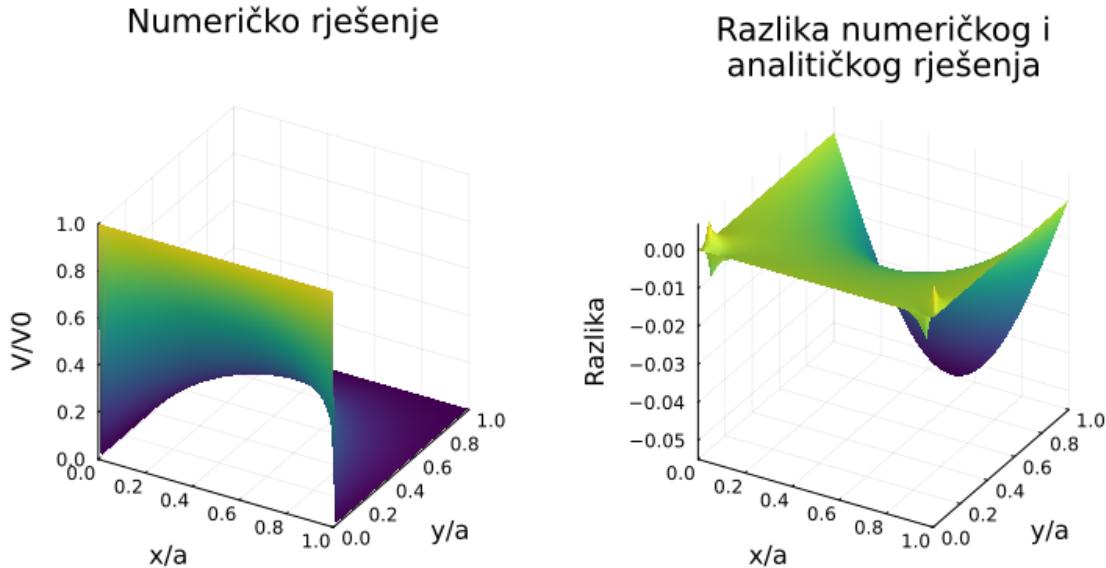
Analitičko rješenje:

$$V(x, y) = \frac{2V_0}{\pi} \tan^{-1} \left(\frac{\sin\left(\frac{\pi y}{a}\right)}{\sinh\left(\frac{\pi x}{a}\right)} \right) \quad (4.32)$$

Za $a = 1$ i $V_0 = 1$ ono se svodi na:

$$V(x, y) = \frac{2}{\pi} \tan^{-1} \left(\frac{\sin(\pi y)}{\sinh(\pi x)} \right) \quad (4.33)$$

Sada kada imamo analitičko rješenje problema možemo ga usporediti s numeričkim.



Analitičko rješenje (4.32) uključuje činjenicu da $V \rightarrow 0$ kada $x \rightarrow \infty$ dok je u numeričkom rješenju domena ograničena na $x \in [0, a]$ i potencijal na $x = a$ je $V = 0$. Taj rubni uvjet implicira da je $V = 0$ kada je $x = a$ što se kosi s analitičkom rješenjem po kojem je V i dalje neka pozitivna vrijednost. Zbog toga se tamo nalazi najveća razlika. Također, primijetimo da na uglovima lijevog ruba $x = 0$ gdje je potencijal definiran rubnim uvjetom $V = 1$ imamo skokove u razlici numeričkog i analitičkog rješenja. Do toga dolazi jer na tim uglovima dolazi do nagle promjene potencijala s $V = 1$ na $V = 0$ u malom prostoru.

5 Zaključak

U ovom radu prikazali smo primjenu numeričkih metoda za rješavanje složenih elektrostatičkih problema metodom momenata (MoM) i metodom konačnih razlika (FDM). Kroz detaljnu analizu i implementaciju u programskom jeziku Julia, demonstrirali smo kako se kontinuirani problemi mogu diskretizirati i pretvoriti u sustave linearnih jednadžbi, što omogućuje njihovo efikasno numeričko rješavanje. Primjenom MoM-a, uspjeli smo odrediti raspodjelu naboja na elektrodama diskretizacijom vodiča na pod-elektrode te formuliranjem sustava jednadžbi koji povezuje potencijale i naboje. Iskoristili smo aproksimacije za izračun koeficijenata $L_{i,j}$ kako bismo optimizirali računanje i izbjegli numerički neefikasne integrale. Rezultati su prikazani vizualno, što omogućuje bolje razumijevanje ponašanja naboja na elektrodama. Korištenjem FDM-a za rješavanje Laplaceove jednadžbe u dvodimenzionalnom prostoru, diskretizirali smo prostor na mrežu čvorova i formulirali sustav jednadžbi na temelju potencijala na susjednim čvorovima. Implementacija u Juliji omogućila nam je efikasno rješavanje velikih sustava jednadžbi te vizualizaciju raspodjele potencijala u prostoru. Ovi primjeri ističu važnost numeričkih metoda u elektrostatici i pokazuju kako moderni alati poput Julije mogu značajno olakšati implementaciju i analizu složenih fizičkih problema. Demonstrirali smo da je moguće postići visoku točnost i efikasnost u rješavanju problema, što otvara vrata za daljnje primjene.

Literatura

- [1] Balbaert, I. Julia 1.0 Programming Second Edition. Birmingham-Mumbai : Packt Publishing Ltd, 2018.
- [2] Dworsky, L. N. Introduction to Numerical Electrostatics Using MATLAB. New Jersey : John Wiley & Sons, Inc, 2014.
- [3] Griffiths, D. J. Introduction to electrodynamics - Fourth edition. New Jersey : Pearson Education, Inc., 2013.
- [4] Julia.org, <http://julialang.org/>, 6.9.2024.