Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Davor Dobrota, Nikola Sočec, Lara Vrabac

Općeniti pristup za paralelizirani izračun polja kružnih zavojnica pravokutnog presjeka

Zagreb, 2022.

Ovaj rad izrađen je u Fakultetu elektrotehnike i računarstva sveučilišta u Zagrebu na Zavodu za komunikacijske i svemirske tehnologije pod vodstvom doc. dr. sc. Daria Bojanjca i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2021./2022.

Sadržaj

1 Uvod

Kružne zavojnice su tip zavojnice koji je u širokoj primjeni, od velikih zavojnica u supravodljivim magnetima do bežičnih punjača prisutnih u većini modernih pametnih telefona. Specifično ćemo promatrati zavojnice pravokutnog presjeka uz pretpostavku da je materijal od kojeg je zavojnica napravljena homogen, odnosno da je gustoća struje homogena po cijelome presjeku. Ovime apstrahiramo većinu realnih zavojnica koje se sastoje od N navoja žice u slučajevima u kojima je N dovoljno velik i zavojnica uredno namotana te u slučajevima u kojima polje ne treba biti precizno izračunato u neposrednoj blizini zavojnice koja ga stvara.

Imajući navedena ograničenja na umu, možemo detaljnije razmotriti ovaj model realne kružne zavojnice. Za opis geometrije danog problema potrebna su 3 parametra: radijus R, debljina a i duljina b. Dodatno je za izračun međuindukcije i sile potrebno poznavati broj navoja N i struju I. Jačina polja, međuindukcija, sila i zakretni moment linearno ovise o navedenim veličinama tako da one nisu geometrijski faktori pa je s njima jednostavno računati. Ovisno o tome u kakvom su međusobnom odnosu debljina i duljina, razlikuju se 4 slučaja:

1) kružna petlja – zanemariva debljina i duljina (petlja)

2) tanka zavojnica – zanemariva debljina (tanka)

3) plosnata zavojnica – zanemariva duljina (plosnata)

4) kružna zavojnica s pravokutnim presjekom (pravokutna)

U zagradama su dani nazivi koji se skraćeno odnose na pojedini tip zavojnice. Posebno su zanimljive pravokutne zavojnice jer je vrijeme izračuna najduže i nema mnogo metoda razvijenih specifično za njih.

Kako se velik broj zavojnica u svakodnevnoj upotrebi može aproksimirati pravokutnom zavojnicom, brz i efikasan izračun polja te međusobne interakcije (u obliku međuindukcije, sile i zakretnog momenta) takvih zavojnica mogao bi omogućiti bolji i brži dizajn sustava koji ih koriste. Primjerice, za dizajn bežičnog punjača potrebno je varirati čak 6 parametara (dimenzije svake od dvije zavojnice) pa je za pronalazak optimalnog oblika potrebno izvršiti proračune za tisuće, ako ne i milijune, različitih dimenzija zavojnica. U navedenomu bi uvelike doprinio pristup koji može izračunati međuindukciju u redu milisekundi, a to je i jedan od ciljeva ovoga rada.

Postoje brojni pristupi izračuna polja, međuindukcije, sile i zakretnog momenta [1-15], no unatoč tome malen broj radova se bavi ovim problemom u njegovom najopćenitijem obliku. Općeniti slučaj je obično riješen korištenjem metode konačnih elemenata - FEM (*finite element method*). Nedostatak navedene metode je korištenje skupog softwarea te spori izračuni koji mogu trajati i sat vremena [7]. Redovito je korisno uzeti neki pristup koji se koriste za petlje, poput [3] i [14], te ga proširiti na pravokutnu zavojnicu, preko npr. filament metode. Dodatni problem je činjenica da podatci o performansama pristupa nekada nisu diskutirani u radu pa, iako su neke metode vrlo precizne, mogu biti i vrlo spore.

Prođimo kratko što do sada objavljene metode nude. Rad [1] predstavlja početak niza radova koji će uslijediti od autora Slobodana Babica, a u njemu su dane osnovne formule za međuindukciju sustava dviju tankih i plosnatih koaksijalnih zavojnica te njihovu samoindukciju. U radu [2] je predstavljena numerička metoda za međuindukciju pravokutne zavojnice i petlje koje leže na istoj osi, dok rad [3] razmatra međuindukciju dvije zavojnice sa zarotiranim osima. Rad [4] predstavlja velik korak naprijed u preciznosti izračuna međuindukcije dviju pravokutnih zavojnica s paralelnim osima te njihove samoindukcije, no na račun dugačkog vremena izračuna, reda minuta. Radovi [5] i [6], predstavljaju dopunu u obliku novih konfiguracija koaksijalnih zavojnica te dodatni razvoj prethodnih modela uz dodatak izračuna sile. U radu [7] je napravljen vrlo koristan pregled performansi i preciznosti dosad napravljenih metoda za izračun međuindukcije dviju koaksijalnih pravokutnih zavojnica, a također je predstavljen pristup s najbržim vremenom izračuna od 0.03 sekunde.

U [8] je prikazan noviji i znatno brži pristup za izračun međuindukcije zavojnica s paralelnim osima, no ne radi za sve geometrijske konfiguracije pa je uz njega prikazana efikasnija implementacija [4] (vrijeme izračuna reda sekundi). Rad [9] je možda najvažniji jer je u njemu prezentiran najnoviji precizni i općeniti pristup izračuna međuindukcije dviju pravokutnih zavojnica. Vrijeme izračuna je 8 sekundi za preciznost od 8 značajnih znamenaka, a otprilike sat vremena za 16 značajnih znamenaka. Ovo predstavlja vrijeme koje je potrebno nadmašiti nekoliko puta, uz usporedivu preciznost, kako bi naš novi pristup pokazao svoju vrijednost. Općeniti pristup je prikazan i u nešto novijem radu [10] no nema mnogo točaka za usporedbu te su one uglavnom dane grafički pa je precizna usporedba teška. Važno je napomenuti da je najbrže vrijeme izračuna oko 0.3 sekunde. Još jedan općeniti pristup diskutiran je u radu [11] no usporedba je opet prilično teška zbog grafičkog prikaza izračunatih vrijednosti. Usporedba s navedenim radom nije zanimljiva jer su pogreške reda 10^{-3}, što je znatno veće od pogreške koju nam je cilj ostvariti $\left(10^{-7}\right)$.

Radovi koji se bave silom i zakretnim momentom koje osjećaju parovi zavojnica znatno su rjeđi, vjerojatno zato što mehanička sila nije nešto što je potrebno razmatrati u većini primjena kružnih zavojnica. Također je lako dobiti dobru aproksimaciju preko numeričkog izračuna gradijenta međuindukcije u sfernim koordinatama. Slučajevi u kojima su sila i zakretni moment važni su supravodljive zavojnice i određeni tipovi elektromotora. Radovi [13] i [15] upravo se bave silama koje nastaju zbog odstupanja položaja od zajedničke osi u sustavu tri supravodljive zavojnice. Proračunate vrijednosti nisu pretjerano precizne no to su jedini pristupi ciljano razvijeni za općeniti slučaj sile i zakretnog momenta između dvije kružne zavojnice pravokutnoga presjeka. Za evaluaciju preciznosti nam je posebno važan Babicev rad [14] u kojemu su dani precizni rezultati za silu između dvije kružne petlje u proizvoljnom položaju. Za usporedbu preciznosti izračuna sile između tankih i plosnatih zavojnica sa zajedničkom osi također nam je koristan rad [12] istoga autora.

Imajući na umu navedene rezultate dosadašnjih radova i pristupa možemo jasno formulirati cilj ovoga rada - razraditi i implementirati brz i precizan pristup izračuna magnetskog polja, njegovog potencijala i gradijenta te pristup za izračun međuindukcije, sile i zakretnog momenta u slučaju proizvoljnoga broja kružnih zavojnica u proizvoljnoj konfiguraciji. Navedeni izrazi će se izvesti i implementirati u apstraktnom razredu Coil koji će predstavljati kružnu zavojnicu, sadržavati sve njene važne atribute te metode za izvršavanje proračuna (polje zavojnice, samoindukcija, međuindukcija para zavojnica, sila i zakretni moment para zavojnica i druge). Također, za razliku od većine ostalih pristupa koji su implementirani u programskom jeziku vrlo visoke razine poput Mathematice, MATLAB-a ili Pythona, naš pristup će biti implementiran u C++ programskom jeziku što bi trebalo znatno poboljšati performanse. Uz korištenje efikasnijeg jezika, također će biti korištena paralelizacija na dvije razine – na razini jedne zavojnice i na razini sustava zavojnica. Navedeno će omogućiti efikasnu uporabu velikog broja procesorskih jezgri u slučaju velikog broja proračuna, npr. tisuća vrijednosti međuindukcije za različite konfiguracije para zavojnica u optimizacijskom problemu ili velikog sustava zavojnica kakvog možemo pronaći u radarskom sustavu.

Korištenje grafičke kartice za ubrzanje izračuna velikog broja vrijednosti polja će se implementirati na osnovnoj razini u svrhu testiranja potencijala korištenja grafičke kartice u budućem radu. Stoga je pristup razvijen s višejezgrenim procesorima na umu, a zatim adaptiran za implementaciju u kontekstu GPGPU (*General-Purpose computing on Graphics Processing Units*) paradigme korištenjem Nvidia CUDA alata. CUDA je platforma za paralelno programiranje na grafičkim akceleratorima tvrtke Nvidia. Primjena GPGPU pristupa nije pronađena u relevantnoj znanstvenoj literaturi. Također valja napomenuti da u literaturi nismo pronašli detaljnu razradu gradijenta magnetskoga polja, ali kako je ona nužna za MRI (*Magnetic Resonance Imaging*) i proračune sa česticama, nju ćemo isto uključiti u rad.

Implementacija će biti dostupna na operacijskim sustavima Windows i Linux, u obliku koda u C++ jeziku te Python modula, koji je omotač navedenog C++ koda, što bi trebalo pružiti gotovo jednake performanse kao da je korišteno direktno iz C++ jezika, uz dodatnu fleksibilnost Pythona.

Konačno, kako ćemo u daljnjemu tekstu vrlo često pričati o poljima, definirajmo sljedeće pokrate. Kada se kaže potencijal misli se na $\boldsymbol{A}$ odnosno magnetski vektorski potencijal, kada se kaže magnetsko polje misli se na $\boldsymbol{B}$ odnosno gustoću toka magnetskog polja, a kada se kaže gradijent misli se na gradijent magnetskog polja, odnosno matematički rečeno totalnu derivaciju gustoće magnetskog toka koju ćemo označavati s $\mathbf{G}$. Ove tri fizikalne veličine zajednički će se nazivati poljima.

2 Osnove teorije

2.1 Filozofija pristupa

Prije nego razložimo detalje našega pristupa, važno je razumjeti zašto smo se baš za njega odlučili. U literaturi su pronađene brojne matematički impresivne metode rješavanja problema; uglavnom se peterostruki ili šesterostruki integrali nastoje svesti na što manje stupnjeva integracije analitičkim ili polu-analitičkim metodama, a neki pristupi zatim primjenjuju numeričku integraciju nad preostalih nekoliko slojeva, ako takvih ima. Nedostatak navedenog je korištenje funkcija koje je teško izračunati. Ako testiramo brzine izračuna pojedinih funkcija u C++ jeziku na Windows operacijskom sustavu uočit ćemo da je svaka funkcija koja nije obuhvaćena zbrajanjem, oduzimanjem, množenjem i dijeljenjem prilično spora – potreban je velik broj procesorskih ciklusa za izračun za razliku od samo jednog za ostale operacije. Kako bismo otključali puni potencijal modernoga procesora moramo se odmaknuti od čistih x86 instrukcija.

Kako je C++ prevedeni programski jezik (*compiled language*) možemo prevoditelju zadati da ubrza kod koji smo napisali te omogućiti upotrebu AVX i AVX2 registara kako bi se provele SIMD (*Single Instruction Multiple Data*) optimizacije. Navedeno dramatično ubrzava izvođenje osnovnih operacija – moguće je nekoliko zbrajanja i množenja napraviti u jednome ciklusu. Dodatna korist je paralelno korjenovanje 4 broja u 2 ciklusa [19]. Druge funkcije poput sinusa, kosinusa i logaritma su također znatno brže na oba operacijska sustava i prema našem testiranju izvršavaju se u 5 do 10 ciklusa. Ovo će naravno ovisiti o samome procesoru no slične vrijednosti se očekuju na svim novijim procesorima jer se AVX2 prvi put pojavio na Intelovim procesorima 2011. godine, uz raširenu upotrebu od 2013.

Dakle zaključujemo da već vrlo osnovne funkcije mogu znatno usporiti izvođenje programa tako da valja reducirati njihovu uporabu koliko god je moguće. Zbog ovoga ograničenja, vjerojatno će biti nemoguće znatno smanjiti broj slojeva integracije pa će se gotovo svi slojevi numerički integrirati. Kako bismo poboljšali preciznost koristit ćemo Gauss-Legendre kvadraturu te će svakome sloju biti iskorišten specifičan broj inkremenata određen po algoritmu kojeg ćemo diskutirati kasnije. Ovo teoretski nije najpreciznija kvadratura koju smo mogli koristiti no implementacija je prilično jednostavna i efikasna.

Veliki broj slojeva integracije također nosi jednu veliku prednost – moguće je efikasno raspodijeliti teret između više jezgara i tako poboljšati performanse za red veličine ili više (većina modernih procesora posjeduje 4 ili više jezgri, odnosno 8 ili više dretvi). Korištenje jednostavnih funkcija, fiksne kvadrature i masovne paralelizacije je također iznimno pogodno za grafički akceleriranu implementaciju.

Kako bismo izbjegli direktno integriranje pet ili šest slojeva, problem međuindukcije, sile i zakretnog momenta možemo svesti na trostruki integral uz poznat potencijal, odnosno magnetsko polje, u određenim točkama. Zato, ako znamo efikasno izračunati polja, znamo izračunati i njih.

2.2 Osnovne jednadžbe

Kako želimo dobiti što jednostavnije izraze za polja, prirodno je započeti s najjednostavnijim jednadžbama koje ujedno i tvore temelj klasičnog elektromagnetizma - Maxwellovim jednadžbama [16]. Izražene u diferencijalnome obliku, one su redom

$$\nabla \cdot \boldsymbol{E} = \frac{\rho}{\epsilon_0}, \tag{1}$$

$$\nabla \cdot \boldsymbol{B} = 0, \tag{2}$$

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t}, \tag{3}$$

$$\nabla \times \boldsymbol{B} = \mu_0 \boldsymbol{J} + \mu_0 \epsilon_0 \frac{\partial \boldsymbol{E}}{\partial t}. \tag{4}$$

S obzirom da je magnetsko polje solenoidalno, njemu se pridružuje pripadni vektorski potencijal $\boldsymbol{A}$ te vrijedi relacija $\boldsymbol{B} = \nabla \times \boldsymbol{A}$. Manipulacijom 2 i 4, pod pretpostavkom homogene gustoće struje, dobiva se Biot-Savartov zakon za vektorski potencijal i magnetsko polje

$$\boldsymbol{A}[\boldsymbol{r}] = \frac{\mu_0}{4\pi} I \int_C \frac{d\boldsymbol{l}}{|\boldsymbol{r} - \boldsymbol{R}|}, \tag{5}$$

$$\boldsymbol{B}[\boldsymbol{r}] = \frac{\mu_0}{4\pi} I \int_C \frac{(d\boldsymbol{l} \times (\boldsymbol{r} - \boldsymbol{R}))}{|\boldsymbol{r} - \boldsymbol{R}|^3}. \tag{6}$$

Ako umjesto tanke žice imamo površinu kroz koju teče određena gustoća struje (5) i (6) postaju

$$\boldsymbol{A}[\boldsymbol{r}] = \frac{\mu_0}{4\pi} \iiint_\Omega \frac{\boldsymbol{J} dV}{|\boldsymbol{r} - \boldsymbol{R}|}, \tag{7}$$

$$\boldsymbol{B}[\boldsymbol{r}] = \frac{\mu_0}{4\pi} \iiint_\Omega \frac{(\boldsymbol{J} \times (\boldsymbol{r} - \boldsymbol{R}))dV}{|\boldsymbol{r} - \boldsymbol{R}|^3}. \tag{8}$$

Zadnje polje koje treba definirati je gradijent magnetskog polja. Njegova definicija slijedi iz razmatranja magnetskog dipola konstantnog dipolnog momenta $\boldsymbol{m}$ čija je potencijalna energija u magnetostatskom slučaju jednaka $U = -\boldsymbol{m} \cdot \boldsymbol{B}$. Za magnetsku silu i zakretni moment tada vrijede izrazi

$$\boldsymbol{F}_{dipol}[\boldsymbol{r}] = \nabla(\boldsymbol{m} \cdot \boldsymbol{B}[\boldsymbol{r}]) = (\boldsymbol{m} \cdot \nabla)\boldsymbol{B}[\boldsymbol{r}] = \mathbf{G}[\boldsymbol{r}]\boldsymbol{m}, \tag{9}$$

$$\boldsymbol{\tau}_{dipol}[\boldsymbol{r}] = \boldsymbol{m} \times \boldsymbol{B}[\boldsymbol{r}]. \tag{10}$$

Ovdje se $\mathbf{G}$ definira kao 3x3 matrica, koja će u magnetostatskom slučaju biti simetrična

$$\mathbf{G}[\boldsymbol{r}] = \begin{bmatrix} \frac{\partial B_x}{\partial x} & \frac{\partial B_x}{\partial y} & \frac{\partial B_x}{\partial z} \\ \frac{\partial B_y}{\partial x} & \frac{\partial B_y}{\partial y} & \frac{\partial B_y}{\partial z} \\ \frac{\partial B_z}{\partial x} & \frac{\partial B_z}{\partial y} & \frac{\partial B_z}{\partial x} \end{bmatrix} \tag{11}$$

Manipulacijom Faradayjevog zakona, odnosno jednadžbe (3) i vektorskog potencijala dobivamo izraz za međuindukciju izvora magnetskog polja (u našemu slučaju zavojnice) i petlje proizvoljnoga oblika. $\boldsymbol{R}_2$ je radijvektor od središta petlje do neke točke na zatvorenoj krivulji γ, a $d\boldsymbol{l}_2$ je tangenta na tu krivulju

$$M_{12} = M_{21} = \frac{1}{I_1} \oint_{\gamma} \boldsymbol{A}_1[\boldsymbol{r}_{12} + \boldsymbol{R}_2] \cdot d\boldsymbol{l}_2 \tag{12}$$

Ovaj općeniti izraz može se koristiti za izračun međuindukcije između proizvoljnih parametriziranih vodljivih elemenata, no kako je cilj ovoga rada pronaći izraz za međuindukciju dvije kružne zavojnice pravokutnoga presjeka, izraz (12) će kasnije biti zapisan u specifičnijem obliku uz pripadnu skicu.

Izraz za silu izvora magnetskog polja na petlju proizvoljnoga oblika je dan Amperovim zakonom i po 3. Newtonovom zakonu, sila koja djeluje na zavojnice jednaka je po magnitudi a suprotna po predznaku

$$\boldsymbol{F}_{12} = -\boldsymbol{F}_{21} = I_2 \oint_C d\boldsymbol{l}_2 \times \boldsymbol{B}_1[\boldsymbol{r}_{12} + \boldsymbol{R}_2]). \tag{13}$$

Kako bismo dobili zakretni moment u integralu je potrebno dodati još jedan vektorski produkt s radijvektorom $\boldsymbol{R}_2$. Jednadžba tada postaje

$$\boldsymbol{\tau}_{12} = -\boldsymbol{\tau}_{21} = I_2 \oint_C \boldsymbol{R}_2 \times (d\boldsymbol{l}_2 \times \boldsymbol{B}_1[\boldsymbol{r}_{12} + \boldsymbol{R}_2]). \tag{14}$$

Jednadžbe (12) i (13) moguće je zapisati i na nešto općenitiji način, za slučaj proizvoljne funkcije gustoće struje. Pritom je energija pohranjena u sustavu gustoće struje i vektorskog potencijala dana integralom (15) dok je jednadžba za silu magnetskog polja na slobodnu struju dana integralom (16)

$$U_B = \iiint_{\Omega} \boldsymbol{J} \cdot \boldsymbol{A} \; dV, \tag{15}$$

$$\boldsymbol{F}_L = \iiint_{\Omega} \boldsymbol{J} \times \boldsymbol{B} \; dV. \tag{16}$$

U slučaju da se sustav sastoji od vodiča (poput zavojnica), pohranjena energija je direktno povezana s međuindukcijom i samoindukcijom vodiča kao i strujom koja kroz njih teče (15). Međuindukciju je tada moguće dobiti integriranjem samo komponente energije U_{12} koja predstavlja međudjelovanje dvaju vodiča. Silu je moguće dobiti preko gradijenta pohranjene energije (18). Zakretni moment nije praktično zapisati u Kartezijevim koordinatama pa se zapisuje samo magnituda u zadanom smjeru rotacije (19)

$$U_B = \frac{1}{2} L_1 I_1^2 + \frac{1}{2} L_2 I_2^2 + M_{12} I_1 I_2 \implies M_{12} = \frac{U_{12}}{I_1 I_2}, \tag{17}$$

$$\boldsymbol{F}_B = \nabla U_B = I_1 I_2 \left(\frac{\partial M_{12}}{\partial x} \widehat{\boldsymbol{i}} + \frac{\partial M_{12}}{\partial y} \widehat{\boldsymbol{j}} + \frac{\partial M_{12}}{\partial z} \widehat{\boldsymbol{k}} \right), \tag{18}$$

$$\tau_B = I_1 I_2 \frac{\partial M_{12}}{\partial \alpha}. \tag{19}$$

Jednadžbe (7), (8), (11), (12), (13) i (14) su središnji izrazi koje ćemo u sljedećemu poglavlju raspisati za slučaj kružne zavojnice pravokutnoga presjeka. Oni će se pojednostaviti koliko je moguće uz korištenje što manje posebnih funkcija te će ih se potom zapisati u obliku sume. U obliku sume ih je zatim vrlo jednostavno i efikasno implementirati u programski okvir jer se radi o ugniježđenim *for* petljama.

3 Pristup izračunu

3.1 Matrica rotacije i koordinatni sustav

Kako je cilj učiniti jednadžbe što manje kompliciranima valja pomno postaviti problem. Krećemo od jednadžbe za samo jednu petlju. Zbog simetrije problema, prirodni koordinatni sustav je cilindrični, a petlju je posljedično najjednostavnije postaviti u xOy ravninu odnosno u ravninu $z = 0$. No ovakva konfiguracija je vrlo specifična pa je potrebno uvesti linearnu transformaciju koja će transformirati prostor i petlju u opisanu orijentaciju, bez obzira na kojim koordinatama i u kojoj orijentaciji je petlja početno bila. Translacija središta petlje u ishodište je vrlo jednostavna - potrebno je samo je od radijvektora točke u kojoj želimo pronaći polje $\boldsymbol{r}_p$ oduzeti radijvektor točke u kojoj je središte petlje $\boldsymbol{r}_c$. Orijentaciju petlje prije transformacije je moguće zapisati preko jediničnog vektora normale, a njega je pak moguće zadati preko dvaju kutova. Navedene kutove odabiremo kao da pokušavamo jedinični vektor zapisati u sfernim koordinatama $(1, \theta, \vartheta)$ gdje $\theta \in [0, \pi]$ i $\vartheta \in [0, 2\pi]$. Dakle da bismo jedinstveno zapisali poziciju i orijentaciju petlje potreban nam je radijvektor $\boldsymbol{r}_c$ i uređeni par (θ, ϑ). Moguće je zadati i ravninu $Ax + By + Cz + D = 0$ u kojoj leži petlja, no takav slučaj je vrlo jednostavno pretočiti u (θ, ϑ).

Potrebno je definirati matricu rotacije koja će transformirati prostor na željeni način. Kao polaznu točku za konstrukciju pripadne matrice uzimamo matrice koje opisuju rotaciju oko pojedine koordinatne osi, a one su redom

$$R_x[\theta] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \tag{20}$$

$$R_y[\theta] = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{21}$$

$$R_z[\theta] = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{22}$$

Dakle rotaciju Kartezijevog koordinatnog sustava možemo prikazati preko množenja ove 3 matrice uz izbor 3 pripadna kuta. Ovi kutovi se nazivaju Eulerovi kutovi. Postoji više načina za provesti ovu rotaciju, 6 osnovnih konvencija koje se odabiru ovisno o simetriji problema. Možda najpoznatija i najočitija je konvencija koju redovito koriste zrakoplovi - kotrljanje, nagib i skretanje. U navedenomu slučaju zrakoplov rotiramo oko svake od triju glavnih osi. Zbog linearnosti ovih operacija, možemo ih primjenjivati jednu za drugom, a ukupnu transformaciju prikazati umnoškom matrica $R_z[\gamma]R_y[\beta]R_x[\alpha]$. Navedena konvencija se označava skraćeno $X_1Y_2Z_3$. Valja uočiti da nakon primjene svake rotacije nova

rotacija djeluje nad novim koordinatnim osima. Navedeno u pojedinim slučajevima može dovesti da zaglavljenja rotacije poznatijeg pod nazivom *gimbal lock*.

Pokušavši više konvencija, utvrdili smo da našemu slučaju najbolje odgovara $Z_1Y_2Z_3$ konvencija. Pritom je očito da imamo jednu informaciju previše - za orijentirati petlju potrebna su dva kuta, a za ovu rotaciju ih je potrebno specificirati tri. Kut rotacije oko y osi odgovara kutu θ dok jedan od kutova rotacije oko z-osi sigurno odgovara kutu φ. Potrebno je pronaći treći kut takav da, ako je vektor normale zavojnice usmjeren u y smjeru i radijvektor točke čije polje tražimo leži na y osi, inverzna transformacija preslikava radijvektor na z-os. Različitim pokušajima se utvrdilo da se navedeno postiže samo ako su dva kuta rotacije oko z osi jednaka i iznose φ. Dobivamo matricu rotacije

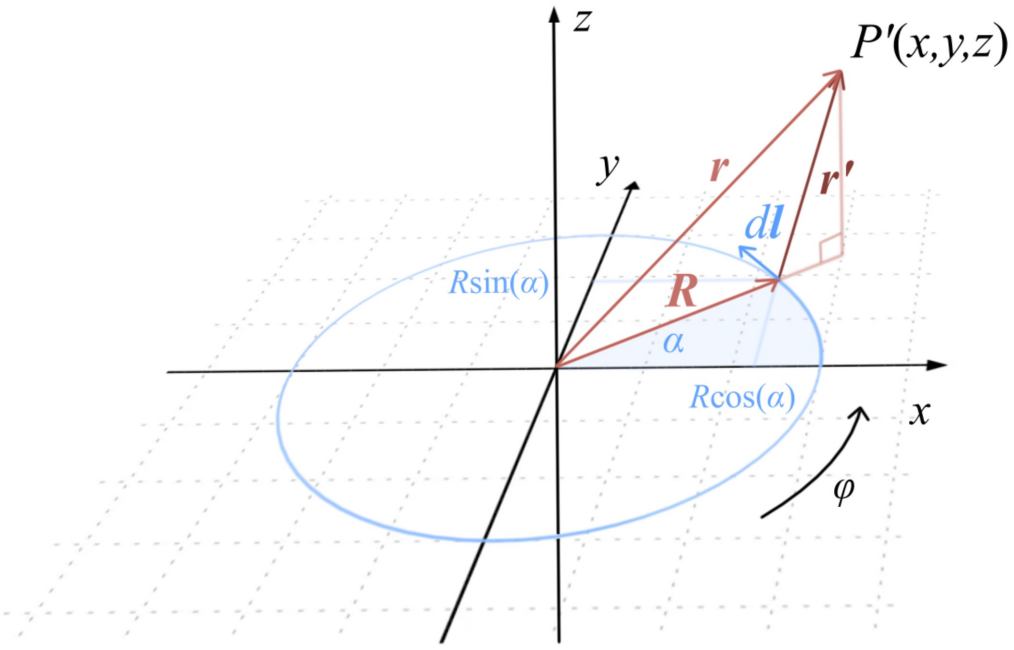
$$\mathbf{T}[\theta,\vartheta] = \begin{bmatrix} \cos\theta\cos\vartheta^2 - \sin\vartheta^2 & -\cos\theta\sin\vartheta\cos\vartheta - \sin\vartheta\cos\vartheta & \sin\theta\cos\vartheta \\ \cos\theta\sin\theta\cos\vartheta + \sin\vartheta\cos\vartheta & -\cos\theta\sin\vartheta^2 + \cos\vartheta^2 & \sin\theta\sin\vartheta \\ -\sin\theta\cos\vartheta & \sin\theta\sin\vartheta & \cos\theta \end{bmatrix}. \tag{23}$$

Navedena matrica (23) ima potencijalni nedostatak u slučaju $\theta = 0$, jer će se petlja tada zarotirati za kut 2ϑ zbog izostanka rotacije oko y-osi. Općenito ovakav odabir kutova dovodi do prekomjernih rotacija u nekim slučajevima no spomenuto za ovu primjenu ipak nije problem zato što su i petlja i polje koje generira radijalno simetrični. Dakle bilo kakva prekomjerna rotacija oko vektora normale ne predstavlja problem.

Konačno je važno napomenuti da matricu rotacije primjenjujemo dva puta. Prije nego počnemo s izračunom polja oduzimamo vektoru položaja radijvektor središta zavojnice i primjenjujemo inverznu rotaciju $\mathbf{T}[-\theta,-\vartheta]$ kako bismo petlju postavili u ravninu $z = 0$. Nakon izračuna komponenti polja u tom koordinatnom sustavu potrebno je dobiveni vektor polja (odnosno matricu gradijenta) transformirati matricom $\mathbf{T}[\theta,\vartheta]$ kako bismo dobili reprezentaciju u vanjskom koordinatnom sustavu. U ovakvome postupku se ne pojavljuje prethodno spomenuti problem pretjerane rotacije baznih vektora koji leže u ravnini petlje zato što se bilo kakva prekomjerna rotacija ispravi prekomjernom rotacijom u suprotnomu smjeru

$$\boldsymbol{r} = \mathbf{T}[-\theta,-\vartheta](\boldsymbol{r}_p - \boldsymbol{r}_c), \tag{24}$$

$$\boldsymbol{B}_p = \mathbf{T}[\theta,\vartheta]\boldsymbol{B}. \tag{25}$$



Slika 1: Shema za primjenu Biot-Savartovog zakona na petlju, prikaz novog koordinatnog sustava nakon primjene transformacije koordinata (24), s P' je označena transformirana točka P

3.2 Integralni izrazi za polja

Na slici 1 je prikazana petlja nakon primjene rotacije i translacije. Zahvaljući transformaciji, moguće je riješiti jednostavniji problem bez gubitka na općenitosti. Postavlja se parametrizacija petlje radijusa R, a točka u kojoj se računa polje zapisuje se u cilindričnim koordinatama

$$\boldsymbol{R} = R\cos(\varphi+\alpha)\boldsymbol{i} + R\sin(\varphi+\alpha)\boldsymbol{j}, \tag{26}$$

$$\boldsymbol{r} = r\cos(\alpha)\boldsymbol{i} + r\sin(\alpha)\boldsymbol{j} + z\boldsymbol{k}. \tag{27}$$

Promatranjem jednadžbi Biot-Savartovog zakona (5) i (6) dalje se računaju

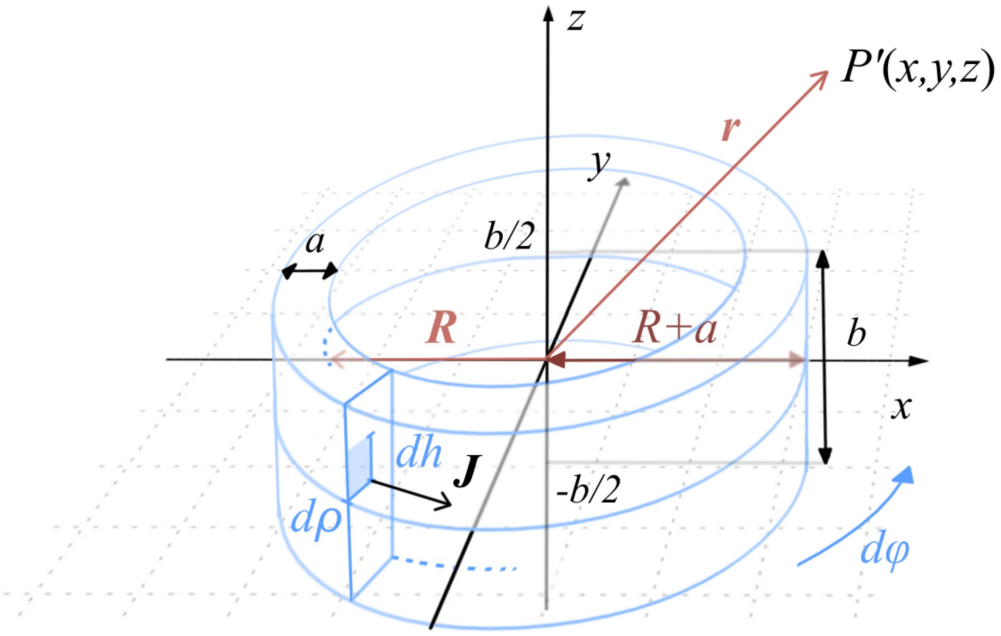
$$d\boldsymbol{l} = \frac{d\boldsymbol{R}}{d\varphi}d\varphi = Rd\varphi(-\sin(\varphi+\alpha)\boldsymbol{i} + \cos(\varphi+\alpha)\boldsymbol{j}), \tag{28}$$

$$D = |\boldsymbol{r}-\boldsymbol{R}|^2 = r^2 + R^2 + z^2 - 2rR\cos\varphi. \tag{29}$$

Nakon malo manipulacije, dobivaju se konačne jednadžbe za magnetski vektorski potencijal (30) i magnetsko polje (31) petlje

$$\boldsymbol{A}[\boldsymbol{r}] = \frac{\mu_0}{4\pi}\int_0^{2\pi} d\varphi \frac{R\cos(\varphi)}{D^{1/2}}\widehat{\boldsymbol{\alpha}}, \tag{30}$$

$$\boldsymbol{B}[\boldsymbol{r}] = \frac{\mu_0}{4\pi} I \int_0^{2\pi} d\varphi \frac{zR\cos(\varphi)\widehat{\boldsymbol{r}} + \left(R^2 - rR\cos(\varphi)\right)\widehat{\boldsymbol{z}}}{D^{3/2}}. \tag{31}$$



Slika 2: Shema za primjenu Biot-Savartovog zakona na zavojnicu pravokutnog presjeka, prikaz novog koordinatnog sustava nakon primjene transformacije (24), s P' je označena transformirana točka P

Vektori polja su u izrazima (30) i (31) zapisani preko baznih vektora cilindričnog koordinatnog sustava, a nakon izračuna ih je lako pretvoriti u Kartezijeve vektore koje onda transformiramo jednadžbom (25). Za petlju se neće raspisati gradijent, nego će nešto kasnije biti dan izraz za gradijent zavojnice. Generalizacija za kružnu zavojnicu pravokutnog presjeka je dosta jednostavna. Kako se radijus petlje mijenja, konstantni radijus R zamjenjujemo s promjenjivom varijablom ρ. Središta pojedinačnih petlji su pomaknuta za h od ishodišta pa z supstituiramo sa $z+h$ (može se supstituirati $z-h$ jer su granice integrala simetrične). Definiramo gustoću struje J i njome mijenjamo struju I. Navedene zamjene su ekvivalentne direktnom rješavanju Biot-Savartovog zakona za zadanu gustoću struje (7) i (8). Konačno se dobivaju jednadžbe (34) i (35)

$$J = \frac{NI}{ab}, \tag{32}$$

$$D' = r^2 + \rho^2 + (z+h)^2 - 2\rho r\cos\varphi, \tag{33}$$

$$\boldsymbol{A}[\boldsymbol{r}] = \frac{\mu_0}{4\pi} J \int_R^{R+a} d\rho \int_{-b/2}^{b/2} dh \int_0^{2\pi} d\varphi \frac{\rho\cos\varphi}{D'^{1/2}} \widehat{\boldsymbol{\alpha}}, \tag{34}$$

$$\boldsymbol{B}[\boldsymbol{r}] = \frac{\mu_0}{4\pi} J \int_R^{R+a} d\rho \int_{-b/2}^{b/2} dh \int_0^{2\pi} d\varphi \frac{(z+h)\,\rho\cos\varphi\,\widehat{\boldsymbol{r}} + \left(\rho^2 - \rho r\cos\varphi\right)\widehat{\boldsymbol{z}}}{D'^{3/2}}. \tag{35}$$

Preostaje još razmotriti gradijent. Kako granice integracije jednadžbe (35) ne ovise o izboru točke, derivira se izraz unutar integrala. Zbog najjednostavnijeg zapisa u cilindričnim koordinatama potrebno

je koristiti lančano pravilo za funkcije više varijabli. Navedeno primjenjujemo 9 puta, jednom za svaku komponentu matrice. Matrica je simetrična tako da zapravo ima svega 6 različitih vrijednosti. Raspisivanjem je također utvrđeno da postoje 4 različita integrala koja treba izračunati, označena s P_1, P_2, P_3 i P_4. Izračun jedne komponente prikazan je u (36)

$$\begin{aligned}\frac{\partial}{\partial x}\left(\frac{(z+h)\,\rho\cos\varphi\cos\alpha}{D'^{3/2}}\right) &= \frac{\partial}{\partial r}\left(\frac{(z+h)\,\rho\cos\varphi\cos\alpha}{D'^{3/2}}\right)\frac{\partial r}{\partial x}+\frac{\partial}{\partial \alpha}\left(\frac{(z+h)\,\rho\cos\varphi\cos\alpha}{D'^{3/2}}\right)\frac{\partial r}{\partial \alpha}\\ &= \frac{\partial}{\partial r}\left(\frac{(z+h)\,\rho\cos\varphi}{D'^{3/2}}\right)\cos\alpha^2+\frac{1}{r}\frac{(z+h)\,\rho\cos\varphi}{D'^{3/2}}\sin\alpha^2\\ &= \frac{3\rho\,(z+h)\,(\rho\cos\varphi-r)\cos\varphi}{D'^{5/2}}\cos\alpha^2+\frac{1}{r}\frac{(z+h)\,\rho\cos\varphi}{D'^{3/2}}\sin\alpha^2.\end{aligned} \tag{36}$$

Konačni rezultat je dan matricom gradijenta (37) čiji su elementi izraženi jednadžbama (38), (39), (40) i (41)

$$\mathbf{G}[\boldsymbol{r}] = \begin{bmatrix} P_2\cos^2\alpha+P_1\sin^2\alpha & (P_2-P_1)\sin\alpha\cos\alpha & P_4\cos\alpha \\ (P_2-P_1)\sin\alpha\cos\alpha & P_2\sin^2\alpha+P_1\cos^2\alpha & P_4\sin\alpha, \\ P_4\cos\alpha & P_4\sin\alpha & P_3 \end{bmatrix} \tag{37}$$

$$P_1[\boldsymbol{r}] = \frac{\mu_0}{4\pi}J\int_R^{R+a}d\rho\int_{-b/2}^{b/2}dh\int_0^{2\pi}d\varphi\frac{1}{r}\frac{(z+h)\,\rho\cos\varphi}{D'^{3/2}}, \tag{38}$$

$$P_2[\boldsymbol{r}] = \frac{\mu_0}{4\pi}J\int_R^{R+a}d\rho\int_{-b/2}^{b/2}dh\int_0^{2\pi}d\varphi\frac{3\rho\,(z+h)\,(\rho\cos\varphi-r)\cos\varphi}{D'^{5/2}}, \tag{39}$$

$$P_3[\boldsymbol{r}] = \frac{\mu_0}{4\pi}J\int_R^{R+a}d\rho\int_{-b/2}^{b/2}dh\int_0^{2\pi}d\varphi\frac{3\rho\,(z+h)\,(r\cos\varphi-\rho)}{D'^{5/2}}, \tag{40}$$

$$P_4[\boldsymbol{r}] = \frac{\mu_0}{4\pi}J\int_R^{R+a}d\rho\int_{-b/2}^{b/2}dh\int_0^{2\pi}d\varphi\frac{\rho\cos\varphi\left(2\rho^2+2r^2-(z+h)^2-\rho r\cos\varphi\right)-3\rho^2 r}{D'^{5/2}}. \tag{41}$$

Korištenje cilindričnih koordinata ima jedan nedostatak - matrica zapisana izrazom (37) ne vrijedi u singularom slučaju $r=0$. Ako promotrimo limes izraza (38), (39), (40) i (41) kada $r\to 0$ dobivamo da vrijedi $P_1 = P_4 = 0$, dok su druga dva integrala konačna i vrijedi $P_3 = -2P_2$. Maxwellova jednadžba (2) mora biti zadovoljena pa na dijagonali moraju ležati redom P_2, P_2, P_3. Elementi van glavne dijagonale su jednaki 0, a navedeno je provjereno prebacivanjem izraza (35) u Kartezijeve koordinate za $x=0$ i $y=0$ i parcijalnom derivacijom po prikladnim varijablama. Matrica u singularnom slučaju postaje

$$\mathbf{G}_z[\boldsymbol{r}] = \begin{bmatrix} P_2 & 0 & 0 \\ 0 & P_2 & 0 \\ 0 & 0 & P_3 \end{bmatrix}. \tag{42}$$

Ovime je razrada osnovnih jednadžbi polja završena. Jednadžbe se uglavnom sastoje od osnovnih aritmetičkih operacija i korjenovanja, uz izuzetak funkcije $\cos\varphi$, što je pogodno za buduću implementaciju. Izraze nije moguće lijepo integrirati po φ jer se radi o eliptičkom integralu, a nije moguće ni po ρ jer integral nije elementaran. Iako bi implementacija posebne funkcije za izračun eliptičkog integrala mogla

potencijalno ubrzati izračun, njena implementacija onemogućila bi alternativu - integriranje izraza po h. Integriranje po h neće ubrzati izračun u slučaju petlje i plosnate zavojnice, no navedeni slučajevi su već dobro pokriveni u literaturi i imaju jedan sloj integracije manje od pravokutne zavojnice. Zbog toga funkcije izračuna polja dijelimo na dvije skupine koje će se označavati kao spore metode (za petlju i plosnatu zavojnicu) i brze metode (za tanku i pravokutnu zavojnicu). Izraz metoda je ovdje korišten u programskom kontekstu zbog implementacije dviju različitih metoda u razredu zavojnica. Matematički rečeno, koriste se dvije različite vrste izraza ovisno o tipu zavojnica.

Izrazi nakon integriranja postaju nešto kompliciraniji, no najvažnija promjena je pojava funkcije $\sinh^{-1}(x)$. Uvode se pokrate (43) i dobivaju konačni izrazi za brze metode

$$h_t = z + \frac{b}{2}, \quad h_b = z - \frac{b}{2}, \tag{43}$$

$$D_t = r^2 + \rho^2 + {h_t}^2 - 2\rho r\cos\varphi, \quad D_b = r^2 + \rho^2 + {h_b}^2 - 2\rho r\cos\varphi,$$

$$C = r^2 + \rho^2 - 2\rho r\cos\varphi,$$

$$C_1 = (\rho^2 + r^2)\cos\varphi - 2\rho r,$$

$$C_2 = 2\rho^2 r^2\cos\varphi\left(\cos^2\varphi + 2\right) - \rho r\left(3\cos^2\varphi + 1\right)\left(\rho^2 + r^2\right) + \cos\varphi(r^4 + \rho^4),$$

$$\boldsymbol{A}'[\boldsymbol{r}] = \frac{\mu_0}{4\pi} J \int_R^{R+a} d\rho \int_0^{2\pi} d\varphi \rho\cos\varphi \left(\sinh^{-1}\frac{h_t}{\sqrt{C}} - \sinh^{-1}\frac{h_b}{\sqrt{C}}\right)\widehat{\boldsymbol{\alpha}}, \tag{44}$$

$$\boldsymbol{B}'[\boldsymbol{r}] = \frac{\mu_0}{4\pi} J \int_R^{R+a} d\rho \int_0^{2\pi} d\varphi \left(\rho\cos\varphi\left(\frac{1}{\sqrt{D_b}} - \frac{1}{\sqrt{D_t}}\right)\widehat{\boldsymbol{r}} + \frac{\rho^2 - \rho r\cos\varphi}{C}\left(\frac{1}{{D_t}^{3/2}} - \frac{1}{{D_b}^{3/2}}\right)\widehat{\boldsymbol{z}}\right), \tag{45}$$

$$P_1'[\boldsymbol{r}] = \frac{\mu_0}{4\pi} J \int_R^{R+a} d\rho \int_0^{2\pi} d\varphi \frac{\rho\cos\varphi}{r}\left(\frac{1}{\sqrt{D_b}} - \frac{1}{\sqrt{D_t}}\right), \tag{46}$$

$$P_2'[\boldsymbol{r}] = \frac{\mu_0}{4\pi} J \int_R^{R+a} d\rho \int_0^{2\pi} d\varphi \rho\cos\varphi\left(\rho^2 - \rho r\cos\varphi\right)\left(\frac{1}{{D_t}^{3/2}} - \frac{1}{{D_b}^{3/2}}\right), \tag{47}$$

$$P_3'[\boldsymbol{r}] = \frac{\mu_0}{4\pi} J \int_R^{R+a} d\rho \int_0^{2\pi} d\varphi \rho\left(\rho^2 - \rho r\cos\varphi\right)\left(\frac{1}{{D_t}^{3/2}} - \frac{1}{{D_b}^{3/2}}\right), \tag{48}$$

$$P_4'[\boldsymbol{r}] = \frac{\mu_0}{4\pi} J \int_R^{R+a} d\rho \int_0^{2\pi} d\varphi \frac{\rho}{C^2}\left(h_t\left(C_1 D_t + \frac{C_2}{D_t^{3/2}}\right) - h_b\left(C_1 D_b + \frac{C_2}{D_b^{3/2}}\right)\right). \tag{49}$$

Pojava funkcije $\sinh^{-1} x$ pogotovo je problematično jer se pokazala iznimno sporom u usporedbi s korijenom i kosinusom. Rješenje ovoga problema bit će detaljnije diskutirano u 4.2, no očito je da će računanje potencijala biti najsporija od svih metoda samo zbog prisutnosti ove funkcije. Performanse su detaljno testirane u 5.2. Preostaje razmotriti kako svih 12 izraza numerički integrirati na način da ih se zapiše u obliku sume koji je pogodan za implementaciju. Navedeno je učinjeno u sljedećem poglavlju 3.3.

Za kraj valja napomenuti da je kod svih 12 izraza integral po φ simetričan oko π pa je moguće integrirati samo na intervalu $[0, \pi]$ i pomnožiti rezultat s 2. Ovo bi trebalo poboljšati performanse (ili preciznost) u numeričkoj integraciji.

3.3 Gauss-Legendre kvadratura za izračun polja

Gauss-Legendre kvadratura [17] je poseban slučaj Gaussove kvadrature u slučaju definiranog integrala. Gaussova kvadratura je jedna od niza numeričkih metoda i odlikuje se mogućnošću egzaktnog izračuna integrala polinoma stupnja $2n-1$ gdje je n broj danih točaka. Možemo je smatrati generalizacijom Newton-Cotesove kvadrature jer omogućuje odabir pozicija točaka čije će se vrijednosti uzeti u obzir prilikom integracije. Gaussova kvadratura je općenito dana izrazom

$$\int_a^b f[x]dx = \int_{-1}^{1} \frac{b-a}{2} f\left[\frac{a+b}{2} + \frac{b-a}{2}p\right] dp = \sum_i^n w_{n,i} \frac{b-a}{2} f\left[\frac{b+a}{2} + \frac{b-a}{2} p_{n,i}\right]. \tag{50}$$

Dokle god možemo pronaći prikladne težinske vrijednosti $w_{n,i}$ i pozicije $p_{n,i}$ da vrijedi (50), metoda se kvalificira kao Gaussova kvadratura. Dakle, Gauss-Legendre kvadraturu odlikuje poseban odabir težinskih vrijednosti i pozicija za n-ti red. Glavne prednosti su brza konvergencija za analitičke funkcije, visoka preciznost, otpornost na nestabilnost pri velikom redu kvadrature te dobra mogućnost implementacije s tipovima podataka ograničene preciznosti (nema pogreške zbog sumiranja). Težinske vrijednosti i pozicije za kvadrature reda $[1, 100]$ su preuzete sa stranice [18]. Jedan nedostatak je lošija preciznost kod funkcija koje jesu integrabilne ali imaju singularitete na rubu intervala integracije, što se može pojaviti prilikom izračuna polja unutar zavojnice ako petlja računa doprinos polju unutar sebe.

Imajući navedeno na umu, primjenjujemo Gaussovu kvadraturu uzastopno 3 puta na izraze (34), (35), (38), (39), (40) i (41). Definiraju se pokrate (51) i dobivaju izrazi za spore metode u obliku sume dani u (52), (53), (54), (55), (56) i (57)

$$\rho_i = R + \frac{a}{2}\left(1 + p_{n_a,i}\right), \quad h_j = \frac{b}{2} p_{n_b,j}, \quad \varphi_k = \frac{\pi}{2}(1 + p_{n_\varphi,k}), \tag{51}$$
$$D_{i,j,k} = r^2 + {\rho_i}^2 + (z + h_j)^2 - 2\rho_i r\cos\varphi_k,$$

$$\boldsymbol{A} = \frac{\mu_0}{2} NI \sum_i^{n_a} \frac{w_{n_a,i}}{2} \sum_j^{n_b} \frac{w_{n_b,j}}{2} \sum_k^{n_\varphi} \frac{w_{n_\varphi,k}}{2} \frac{\rho_i \cos\varphi_k}{\sqrt{D_{i,j,k}}} \widehat{\boldsymbol{\alpha}}, \tag{52}$$

$$\boldsymbol{B} = \frac{\mu_0}{2} NI \sum_i^{n_a} \frac{w_{n_a,i}}{2} \sum_j^{n_b} \frac{w_{n_b,j}}{2} \sum_k^{n_\varphi} \frac{w_{n_\varphi,k}}{2} \frac{\left(z + h_j\right)\rho_i \cos\varphi_k \widehat{\boldsymbol{r}} + ({\rho_i}^2 - \rho_i r\cos\varphi_k)\widehat{\boldsymbol{z}}}{D_{i,j,k}\sqrt{D_{i,j,k}}}, \tag{53}$$

$$P_1 = \frac{\mu_0}{2} NI \sum_i^{n_a} \frac{w_{n_a,i}}{2} \sum_j^{n_b} \frac{w_{n_b,j}}{2} \sum_k^{n_\varphi} \frac{w_{n_\varphi,k}}{2} \frac{1}{r} \frac{\left(z + h_j\right)\rho_i \cos\varphi_k}{D_{i,j,k}\sqrt{D_{i,j,k}}}, \tag{54}$$

$$P_2 = \frac{\mu_0}{2} NI \sum_i^{n_a} \frac{w_{n_a,i}}{2} \sum_j^{n_b} \frac{w_{n_b,j}}{2} \sum_k^{n_\varphi} \frac{w_{n_\varphi,k}}{2} \frac{3\rho_i \left(z + h_j\right)\left(\rho_i \cos\varphi_k - r\right)\cos\varphi_k}{D_{i,j,k} D_{i,j,k}\sqrt{D_{i,j,k}}}, \tag{55}$$

$$P_3 = \frac{\mu_0}{2} NI \sum_i^{n_a} \frac{w_{n_a,i}}{2} \sum_j^{n_b} \frac{w_{n_b,j}}{2} \sum_k^{n_\varphi} \frac{w_{n_\varphi,k}}{2} \frac{3\rho_i \left(z + h_j\right)\left(r\cos\varphi_k - \rho_i\right)}{D_{i,j,k} D_{i,j,k}\sqrt{D_{i,j,k}}}, \tag{56}$$

$$P_4 = \frac{\mu_0}{2} NI \sum_i^{n_a} \frac{w_{n_a,i}}{2} \sum_j^{n_b} \frac{w_{n_b,j}}{2} \sum_k^{n_\varphi} \frac{w_{n_\varphi,k}}{2} \frac{\rho_i \cos\varphi_k \left(2{\rho_i}^2 + 2r^2 - (z + h_j)^2 - \rho_i r\cos\varphi_k\right) - 3{\rho_i}^2 r}{D_{i,j,k} D_{i,j,k}\sqrt{D_{i,j,k}}}. \tag{57}$$

U ovim izrazima n_a, n_b i n_φ predstavljaju red kvadrature, odnosno broj točaka s kojima se računa, za svaki od 3 sloja integracije. Očekujemo da će više točaka povećavati preciznost izračuna dok ne dođemo do maksimalne preciznosti IEEE 754 podataka tipa double - 15 značajnih znamenki. Velika količina sumiranja bi mogla dovesti do eventualne kumulativne pogreške zbrajanja, no Gauss-Legendre metoda bi trebala biti otporna na to. Navedeno će biti precizno testirano kasnije u radu. Važno je napomenuti da ako je neka od dimenzija zanemariva, sve što treba napraviti je postaviti njezin n na 1 i neće biti utjecaja na izračun. Dakle ovakve sume primjenjive su na sva 4 tipa zavojnica bez dodatne manipulacije izraza. Ipak navedene koristimo samo za petlju i plosnatu zavojnicu.

Preostaje zapisati sumarni oblik brzih metoda za izračun. Uvodimo dodatne pokrate (58) te se dobivaju izrazi za brze metode u obliku sume prikazani u (59), (60), (61), (62), (63) i (64)

$$h_t = z + \frac{b}{2}, \quad h_b = z - \frac{b}{2}, \tag{58}$$

$$D_{t,i,k} = {h_t}^2 + r^2 + {\rho_i}^2 - 2\rho_i r\cos\varphi_k, \quad D_{b,i,k} = {h_b}^2 + r^2 + {\rho_i}^2 - 2\rho_i r\cos\varphi_k,$$

$$C_{i,k} = r^2 + {\rho_i}^2 - 2\rho_i r\cos\varphi_k,$$

$$C_{1,i,k} = \left({\rho_i}^2 + r^2\right)\cos\varphi_k - 2\rho_i r, \quad C_{2,i,k} = {\rho_i}^2 - r\rho_i\cos\varphi_k,$$

$$C_{3,i,k} = 2{\rho_i}^2 r^2\cos\varphi_k\left(\cos^2\varphi_k + 2\right) - \rho_i r\left(3\cos^2\varphi_k + 1\right)\left({\rho_i}^2 + r^2\right) + \cos\varphi_k\left(r^4 + {\rho_i}^4\right),$$

$$\boldsymbol{A}' = \frac{\mu_0}{2}NI\sum_i^{n_a}\frac{w_{n_a,i}}{2}\sum_k^{n_\varphi}\frac{w_{n_\varphi,k}}{2}\rho_i\cos\varphi_k\left(\sinh^{-1}\frac{h_t}{\sqrt{C_{i,k}}} - \sinh^{-1}\frac{h_b}{\sqrt{C_{i,k}}}\right)\widehat{\alpha}, \tag{59}$$

$$\boldsymbol{B}' = \frac{\mu_0}{2}NI\sum_i^{n_a}\frac{w_{n_a,i}}{2}\sum_k^{n_\varphi}\frac{w_{n_\varphi,k}}{2}\left(\left(\frac{\rho_i\cos\varphi_k}{\sqrt{D_{b,i,k}}} - \frac{\rho_i\cos\varphi_k}{\sqrt{D_{t,i,k}}}\right)\widehat{\mathbf{r}} + \frac{C_{2,i,k}}{C_{1,i,k}}\left(\frac{1}{{D_{t,i,k}}^{3/2}} - \frac{1}{{D_{b,i,k}}^{3/2}}\right)\widehat{\mathbf{z}}\right), \tag{60}$$

$$P_1' = \frac{\mu_0}{2}NI\sum_i^{n_a}\frac{w_{n_a,i}}{2}\sum_k^{n_\varphi}\frac{w_{n_\varphi,k}}{2}\frac{\rho_i\cos\varphi_k}{r}\left(\frac{1}{\sqrt{D_{b,i,k}}} - \frac{1}{\sqrt{D_{t,i,k}}}\right), \tag{61}$$

$$P_2' = \frac{\mu_0}{2}NI\sum_i^{n_a}\frac{w_{n_a,i}}{2}\sum_k^{n_\varphi}\frac{w_{n_\varphi,k}}{2}\rho_i\cos\varphi_k C_{2,i,k}\left(\frac{1}{{D_{t,i,k}}^{3/2}} - \frac{1}{{D_{b,i,k}}^{3/2}}\right), \tag{62}$$

$$P_3' = \frac{\mu_0}{2}NI\sum_i^{n_a}\frac{w_{n_a,i}}{2}\sum_k^{n_\varphi}\frac{w_{n_\varphi,k}}{2}\rho_i C_{2,i,k}\left(\frac{1}{{D_{t,i,k}}^{3/2}} - \frac{1}{{D_{b,i,k}}^{3/2}}\right), \tag{63}$$

$$P_4' = \frac{\mu_0}{2}NI\sum_i^{n_a}\frac{w_{n_a,i}}{2}\sum_k^{n_\varphi}\frac{w_{n_\varphi,k}}{2}\frac{\rho_i}{C_{i,k}^2}\left(h_t\left(C_{1,i,k}D_{t,i,k} + \frac{C_{3,i,k}}{D_{t,i,k}^{3/2}}\right) - h_b\left(C_{1,i,k}D_{b,i,k} + \frac{C_{3,i,k}}{D_{b,i,k}^{3/2}}\right)\right). \tag{64}$$

Navedenih 12 jednadžbi je vrlo lagano implementirati programski. Postupak transformacije koordinata opisan u prethodnim poglavljima i implementiran u kodu prikazujemo sažeto izrazima

$$\boldsymbol{A}_p[\boldsymbol{r}_p] = \mathbf{T}[\theta,\vartheta]\boldsymbol{A}[\mathbf{T}[-\theta,-\vartheta](\boldsymbol{r}_p - \boldsymbol{r}_c)], \tag{65}$$

$$\boldsymbol{B}_p[\boldsymbol{r}_p] = \mathbf{T}[\theta,\vartheta]\boldsymbol{B}[\mathbf{T}[-\theta,-\vartheta](\boldsymbol{r}_p - \boldsymbol{r}_c)], \tag{66}$$

$$\mathbf{G}_p[\boldsymbol{r}_p] = \mathbf{T}[\theta,\vartheta]\mathbf{G}[\mathbf{T}[-\theta,-\vartheta](\boldsymbol{r}_p - \boldsymbol{r}_c)]. \tag{67}$$

3.4 Međuindukcija i sila

Razmotrivši detaljno pristup izračuna polja, možemo primijeniti dobivene formule na izračun mjerljivih efekata polja. Magnetsko polje ne možemo mjeriti direktno, već samo posredno preko njegovih efekata. Kod kružnih zavojnica je zanimljiva sila i zakretni moment kojim ona djeluje na neki drugi izvor magnetskog polja koji ne mora nužno biti druga kružna zavojnica. Također je iznimno važna međuindukcija između zavojnice i proizvoljne gustoće struje u prostoru, obično u obliku vodiča zadanog oblika. Međuindukcija je simetrično svojstvo sustava, dakle isti rezultat se dobiva računa li se iz perspektive zavojnice ili struje, dok su sila i magnetski moment antisimetrična svojsta zbog 3. Newtonovog zakona. U ovome radu će dalje biti razmatran isključivo slučaj dvije kružne zavojnice pravokutnoga presjeka.

Za slučaj u kojem je kružna zavojnica pravokutnog presjeka sekundar možemo jednadžbe za međuindukciju (12), silu (13) i zakretni moment (14) zapisati u nešto specifičnijem obliku. Sekundarnu zavojnicu tretiramo kao i primarnu - kao niz infinitezimalnih petlji radijusa ρ. Tada se mogu zapisati općeniti integralni izrazi

$$M_{12} = \frac{N_2}{I_1 a_2 b_2} \int_{R_2}^{R_2+a_2} d\rho \int_{-b_2/2}^{b_2/2} dh \int_0^{2\pi} d\varphi \;\; \boldsymbol{A}_1 \left[\boldsymbol{r}_{c2} + \boldsymbol{r}_2[\rho, h, \varphi]\right] \cdot \boldsymbol{t}_2[\rho, h, \varphi], \tag{68}$$

$$\boldsymbol{F}_{12} = J_2 \int_{R_2}^{R_2+a_2} d\rho \int_{-b_2/2}^{b_2/2} dh \int_0^{2\pi} d\varphi \;\; \boldsymbol{t}_2[\rho, h, \varphi] \times \boldsymbol{B}_1 \left[\boldsymbol{r}_{c2} + \boldsymbol{r}_2[\rho, h, \varphi]\right], \tag{69}$$

$$\boldsymbol{\tau}_{12} = J_2 \int_{R_2}^{R_2+a_2} d\rho \int_{-b_2/2}^{b_2/2} dh \int_0^{2\pi} d\varphi \;\; \boldsymbol{r}_2[\rho, h, \varphi] \times \left(\boldsymbol{t}_p[\rho, h, \varphi] \times \boldsymbol{B}_1 \left[\boldsymbol{r}_{c2} + \boldsymbol{r}_2[\rho, h, \varphi]\right]\right). \tag{70}$$

Za međuindukciju su se isti izrazi mogli dobiti preko jednadžbi za pohranjenu magnetsku energiju (15) i (17). Magnetsku silu i zakretni moment može se također izračunati preko međuindukcije i to se redovito primjenjuje u literaturi. U našemu pristupu se ipak koristi Amperova sila, odnosno izračun preko magnetskog polja. Tada izraz (69) možemo dobiti i preko sile na gustoću struje danu izrazom (18). Izraz $\boldsymbol{r}_2[\rho, h, \varphi]$ predstavlja radijvektor parametrizacije volumena koji čini sekundarnu zavojnicu, dok je $\boldsymbol{t}_2[\rho, h, \varphi]$ derivacija parametrizacije po φ i zapravo predstavlja smjer struje.

Ostatak poglavlja je posvećen posebnim slučajevima ovih formula te razvoju zapisa u obliku sume koji je pogodan za izračun. Započinje se razmatranjem slučaja zajedničke osi koji je relativno dobro pokriven literaturom. Dio pododjeljka će se posvetiti i razmatranju izračuna samoindukcije zavojnice, jer se formula može primijeniti na način da je primarna zavojnica jednaka sekundarnoj. U drugome pododjeljku će se zapisati suma za općeniti slučaj.

3.4.1 Poseban slučaj zajedničke osi

Slučaj zajedničke osi podrazumijeva da su središta obje zavojnice na z osi, dakle da njihova središta imaju koordinate $(0,0,z_1)$ odnosno $(0,0,z_2)$ te da su oba kuta rotacije obje zavojnice postavljena na 0. Simetrija koja se pojavljuje u ovome slučaju znatno pojednostavljuje izraze (68), (69) i (70). Primjenom pravila desne ruke lako se možemo uvjeriti da zakretni moment potpuno nestaje. Sila se dijeli na dvije komponente - komponentu duž zajedničke osi i komponentu radijalno prema van. Ako se smjerovi struja podudaraju sila će biti privlačna, a ako su suprotni odbojna. Ta informacija sadržana je u jakosti struje odnosno u gustoći struje J_1 odnosno J_2 u obliku različitog predznaka za pozitivan i negativan smjer protjecanja struje. Komponenta radijalno prema van nastoji komprimirati navoje zavojnice ako se smjerovi podudaraju a proširiti ih ako su suprotni. Kako je u našemu modelu zavojnica kruto tijelo, otporna je na deformacije pa se radijalne sile pokrate i preostaje samo sila duž z-osi. Također zbog radijalne simetrije unutarnji sloj integracije se reducira na $2\pi\rho$. Pritom su eliminirani vektorski produkti i zadržane jedine komponente koje se nisu reducirale na 0. Dobivaju se izrazi

$$M_{z,12} = 2\pi \frac{N_2}{I_1 a_2 b_2} \int_{R_2}^{R_2+a_2} d\rho_2 \int_{-b_2/2}^{b_2/2} dh_2 \;\; \rho_2 A_{p1} \left[\rho_2 \widehat{\boldsymbol{i}} + (h_2 + z_2)\, \widehat{\boldsymbol{k}}\right], \tag{71}$$

$$\boldsymbol{F}_{z,12} = 2\pi \frac{N_2 I_2}{a_2 b_2} \widehat{\boldsymbol{k}} \int_{R_2}^{R_2+a_2} d\rho_2 \int_{-b_2/2}^{b_2/2} dh_2 \;\; \rho_2 B_{r1} \left[\rho_2 \widehat{\boldsymbol{i}} + (h_2 + z_2)\, \widehat{\boldsymbol{k}}\right]. \tag{72}$$

Pritom u (71) A_{p1} predstavlja apsolutnu vrijednost vektorskog potencijala, a B_{r1} u (72) predstavlja samo apsolutnu vrijednost radijalne komponente magnetskog polja. Dobiveni izrazi se zatim zapisuju u obliku Gauss-Legendre sume i prikazuju u (74) i (75). Koristi se pokrata (73) koja označava pozicije točke na pravokutnom presjeku sekundarne zavojnice

$$\boldsymbol{r}_{i,j} = \left(R_2 + \frac{a_2}{2}\left(1 + p_{n_{a_2},i}\right)\right)\widehat{\boldsymbol{i}} + \left(\frac{b_2}{2} p_{n_{b_2},j} + z_2\right)\widehat{\boldsymbol{k}}, \tag{73}$$

$$M_{z,12} = 2\pi \frac{N_2}{I_1} \sum_{i}^{n_{a_2}} \frac{w_{n_{a_2},i}}{2} \sum_{j}^{n_{b_2}} \frac{w_{n_{b_2},j}}{2} \left(R_2 + \frac{a_2}{2}\left(1 + p_{n_{a_2},i}\right)\right) A_1[\boldsymbol{r}_{i,j}], \tag{74}$$

$$\boldsymbol{F}_{z,12} = 2\pi N_2 I_2 (-\widehat{\boldsymbol{k}}) \sum_{i}^{n_{a_2}} \frac{w_{n_{a_2},i}}{2} \sum_{j}^{n_{b_2}} \frac{w_{n_{b_2},j}}{2} \left(R_2 + \frac{a_2}{2}\left(1 + p_{n_{a_2},i}\right)\right) B_{r1}[\boldsymbol{r}_{i,j}]. \tag{75}$$

Ove izraze za izračun sile i međuindukcije u slučaju z-osi vrlo je jednostavno implementirati. Računski gledano, ovi izrazi su efektivno četverostruke sume u slučaju dvije pravokutne zavojnice. Valja napomenuti da iako je svejedno koja zavojnica je primarna, ipak se očekuju bolji rezultati ako se uzima da je veća zavojnica primarna. U slučaju tanke i plosnate zavojnice, preporuča se staviti tanku kao primar, a plosnatu kao sekundar.

Dalje je uočeno da se može eliminirati još jedan sloj integracije ako se u izrazima (71) i (72) za potencijal i magnetsko polje direktno uvrste izrazi (44) i (45). Tako dobiven slučaj možemo integrirati

po h_2. Nakon što smo implementirali navedeno unaprjeđenje, ipak smo odustali od zamjene izraza za silu (72) zato što je dobiveni izraz u nekim slučajevima pogoršao performanse uz minimalno poboljšanje preciznosti. Stoga (72) i (75) ostavljamo u obliku u kojemu jesu, a izraz za međuindukciju (71) se dalje sređuje. Pritom je važno naglasiti da zapravo opet stvaramo sporu i brzu metodu, samo ovaj put za međuindukciju u slučaju zajedničke osi. Pritom se (71) koristi u slučaju da je sekundar petlja ili plosnata zavojnica, a novi izraz (77) u slučaju da se radi o tankoj ili pravokutnoj zavojnici. Uvode se pokrate

$$h_1 = z_2 - z_1 + \frac{b_2}{2} + \frac{b_1}{2}, \quad h_2 = z_2 - z_1 + \frac{b_2}{2} - \frac{b_1}{2}, \quad h_3 = z_2 - z_1 - \frac{b_2}{2} - \frac{b_1}{2}, \quad h_4 = z_2 - z_1 - \frac{b_2}{2} + \frac{b_1}{2},$$
$$K_1 = h_1 \sinh^{-1}\frac{h_1}{\sqrt{C}} - h_2 \sinh^{-1}\frac{h_2}{\sqrt{C}} + h_3 \sinh^{-1}\frac{h_3}{\sqrt{C}} - h_4 \sinh^{-1}\frac{h_4}{\sqrt{C}},$$
$$K_2 = \sqrt{h_1^2 + C} - \sqrt{h_2^2 + C} + \sqrt{h_3^2 + C} - \sqrt{h_4^2 + C}, \tag{76}$$

$$M'_{z,12} = \frac{\mu_0}{2}\frac{N_1 N_2}{a_1 b_1 a_2 b_2}\int_{R_2}^{R_2+a_2} dr \int_{R_1}^{R_1+a_1} d\rho \int_0^{2\pi} d\varphi \;\; r\rho\cos\varphi\,(K_1 - K_2). \tag{77}$$

Dobiveni izraz je znatno kompliciraniji od prethodnoga i uključuje funkciju $\sinh^{-1} x$ koju u ovome slučaju ne možemo ukloniti kako smo to učinili kod vektorskog potencijala. Unatoč tome, performanse su poboljšane oko 3 puta, uključujući i maleno poboljšanje preciznost u nekim slučajevima. Performanse i preciznost bit će detaljnije diskutirane u usporebi rezultata 5.3.1.

Uočava se znatno pojednostavljenje u slučaju samoindukcije - kada je efektivno ista zavojnica i primar i sekundar. Tada vrijedi $z_2 - z_1 = 0$, $b_2 = b_1 = b$. Jednadžba (77) se reducira na izraz

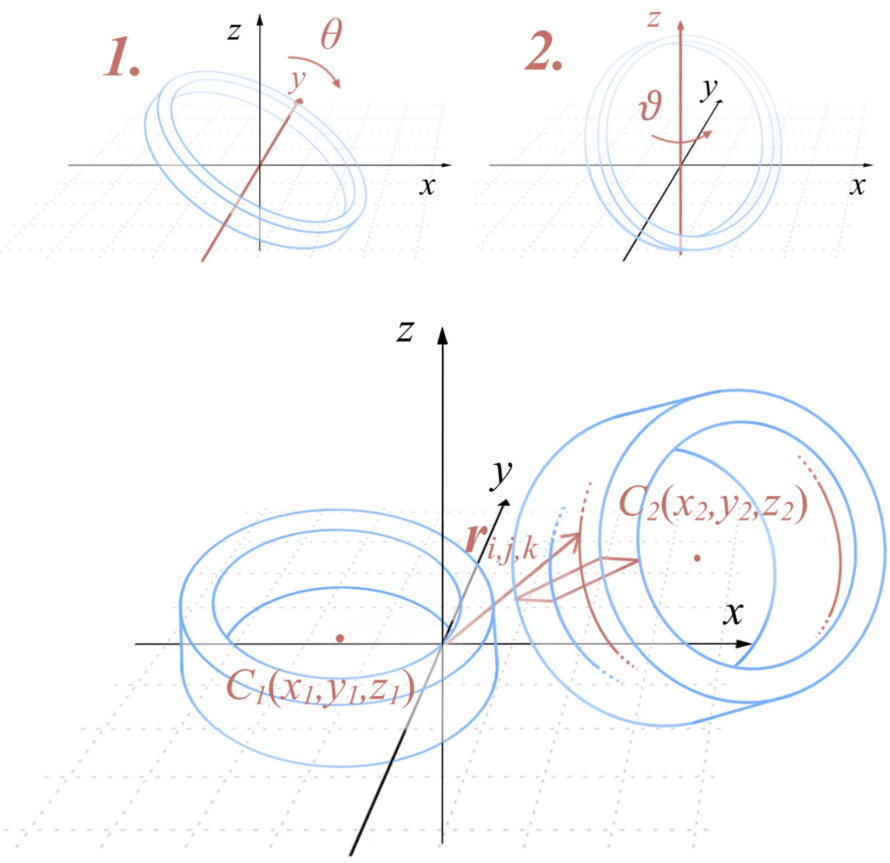
$$L = \frac{\mu_0 N^2}{a^2 b^2}\int_{R_2}^{R_2+a_2} dr \int_{R_1}^{R_1+a_1} d\rho \int_0^{2\pi} d\varphi \;\; r\rho\cos\varphi\left(b\sinh^{-1}\frac{b}{\sqrt{C}} + \sqrt{C} - \sqrt{C+b^2}\right). \tag{78}$$

U posebnom slučaju tanke zavojnice, radi se o jednostrukomu integralu. Valja napomenuti da (78) nije idealni kandidat za primjenu Gauss-Legendre kvadrature jer u slučaju $\rho = r$ divergiraju na rubovima domene. Ovo divergentno ponašanje je do neke mjere ublaženo balansiranjem inkremenata koje će biti diskutirano u 3.5. Izrazi (77) i (78) odstupaju od filozofije pristupa opisane u 2.1, no kako su izračun međuindukcije u slučaju zajedničke osi i izračun samoindukcije važni u primjeni napravljena je iznimka u svrhu bolje preciznosti i performansi ovih metoda. Korištenjem pokrata (79) dolazi se do izraza za brzu međuindukciju i samoindukciju u obliku sume zapisanih izrazima (80) i (81)

$$K_{1,i,k} = h_1 \sinh^{-1}\frac{h_1}{\sqrt{C_{i,k}}} - h_2 \sinh^{-1}\frac{h_2}{\sqrt{C_{i,k}}} + h_3 \sinh^{-1}\frac{h_3}{\sqrt{C_{i,k}}} - h_4 \sinh^{-1}\frac{h_4}{\sqrt{C_{i,k}}}, \tag{79}$$
$$K_{2,i,k} = \sqrt{h_1^2 + C_{i,k}} - \sqrt{h_2^2 + C_{i,k}} + \sqrt{h_3^2 + C_{i,k}} - \sqrt{h_4^2 + C_{i,k}},$$

$$M'_{z,12} = \mu_0\pi\frac{N_1 N_2}{b_1 b_2}\sum_i^{n_{a_2}}\frac{w_{n_{a_2},i}}{2}\sum_j^{n_{a_1}}\frac{w_{n_{a_1},j}}{2}\sum_k^{n_\varphi}\frac{w_{n_{a_2},k}}{2} r_i\rho_j\cos\varphi_k\left(K_{1,i,k} - K_{2,i,k}\right), \tag{80}$$

$$L = 2\pi\mu_0\frac{N^2}{b^2}{}'\sum_i^{n_{a_2}}\frac{w_{n_{a_2},i}}{2}\sum_j^{n_{a_1}}\frac{w_{n_{a_1},j}}{2}\sum_k^{n_\varphi}\frac{w_{n_{a_2},k}}{2} r_i\rho_j\cos\varphi_k\left(b\sinh^{-1}\frac{b}{\sqrt{C_{i,k}}} + \sqrt{C_{i,k}} - \sqrt{C_{i,k}+b^2}\right). \tag{81}$$



Slika 3: Shema za kutove rotacije petlji i postava za izračun međuindukcije, sile i zakretnog momenta

3.4.2 Općeniti slučaj

Konačno se dolazi do glavnog slučaja, slučaja općenitog postava dvije zavojnice. On je najvažniji u smislu usporedbe brzine i preciznosti predstavljenoga pristupa te će omogućiti direktnu provjeru preciznosti izraza za potencijal i polje u većem broju točaka. Dakle izračun polja će biti barem precizan koliko i najgora preciznost izračuna međuindukcije odnosno sile. Koristimo sliku 3 kako bi se problem mogao lakše predočiti.

Kreće se od općih izraza za međuindukciju (68), silu (69) i zakretni moment (70). Potrebno je pretočiti navedene izraze koji sadrže trostruki integral u trostruku sumu tako da se zadaju specifične točke koje su sadržane u volumenu zavojnice - $r[u, v, \varphi]$. Pozicije točaka koje sačinjavaju petlju dobivaju se parametrizacijom općenito orijentirane petlje. Za navedeno se može koristiti matrica rotacije (23) no ona se u ovomu slučaju može pojednostaviti na umnožak dviju matrica rotacija $R_y[\theta]$ (21) i $R_z[\vartheta]$ (22) bez gubitka općenitosti. Time se dobiva matrica rotacije prstena (82) i izraz za radijvektor točke na jediničnom prstenu parametriziran po φ (83). Kako se u izrazima pojavljuje diferencijal vektora položaja, odnosno tangencijalni vektor, zapisuje se i izraz za tangencijalni vektor u točki jediničnog prstena (84)

$$\mathbf{R}\left[\theta,\vartheta\right]=\begin{bmatrix}\cos\theta\cos\vartheta & -\sin\vartheta & \sin\theta\cos\vartheta\\ \cos\theta\sin\vartheta & \cos\vartheta & \sin\theta\sin\vartheta\\ -\sin\theta & 0 & \cos\theta\end{bmatrix}, \tag{82}$$

$$\begin{aligned}\widehat{\boldsymbol{P}}_r\left[\theta,\vartheta,\varphi\right]&=\mathbf{R}\left[\theta,\vartheta\right]\left(\cos\varphi\hat{\boldsymbol{i}}+\sin\varphi\hat{\boldsymbol{j}}\right)\\&=\left(\cos\theta\cos\vartheta\cos\varphi-\sin\vartheta\sin\varphi\right)\widehat{\boldsymbol{i}}+\left(\cos\theta\sin\vartheta\cos\varphi+\cos\vartheta\sin\varphi\right)\widehat{\boldsymbol{j}}-\sin\theta\cos\varphi\widehat{\boldsymbol{k}},\end{aligned} \tag{83}$$

$$\begin{aligned}\widehat{\boldsymbol{T}}_r\left[\theta,\vartheta,\varphi\right]&=\mathbf{R}\left[\theta,\vartheta\right]\left(-\sin\varphi\hat{\boldsymbol{i}}+\cos\varphi\hat{\boldsymbol{j}}\right)\\&=\left(-\cos\theta\cos\vartheta\sin\varphi-\sin\vartheta\cos\varphi\right)\widehat{\boldsymbol{i}}+\left(-\cos\theta\sin\vartheta\sin\varphi+\cos\vartheta\cos\varphi\right)\widehat{\boldsymbol{j}}+\sin\theta\sin\varphi\widehat{\boldsymbol{k}}.\end{aligned} \tag{84}$$

Za dobivanje položaja na petlji izraze (83) i (84) treba pomnožiti s radijusom dane petlje ρ. Dobivena parametrizacija se neće uvrštavati u integrale već direktno u sumu. Vektori su zapisani kao vektor-stupci zbog urednosti. Zadan je radijvektor $\boldsymbol{r}_2$ središta sekundarne petlje i njeni orijentacijski kutovi (θ_2,ϑ_2). Trostruka suma se pritom rastavlja na dvostruku sumu koja određuje koje petlje će se koristiti u proračunu te jednostruku sumu po odabranim točkama na toj petlji. Izbor petlji i točaka se provodi u skladu s Gauss-Legendre kvadraturom. Posebna pozornost pridaje se pozicijama središta svake od individualnih petlji koje su promatrane. Uvode se pokrate (85). Valja napomenuti da krak zakretnog momenta djeluje od središta cijele zavojnice a ne od središta individualne petlje. Konačno se zapisuju općeniti izrazi za međuindukciju (86), silu (87) i zakretni moment (88) između dvije kružne zavojnice pravokutnog presjeka u obliku sume

$$\begin{aligned}\boldsymbol{r}_{i,j,k}&=\begin{bmatrix}x_2+\left(\frac{b}{2}p_{n_{b_2},j}\right)\sin\theta_2\cos\vartheta_2\\ y_2+\left(\frac{b}{2}p_{n_{b_2},j}\right)\sin\theta_2\sin\vartheta_2\\ z_2+\left(\frac{b}{2}p_{n_{b_2},j}\right)\cos\theta_2\end{bmatrix}+\left(R+\frac{a}{2}\left(1+p_{n_{a_2},i}\right)\right)\widehat{\boldsymbol{P}}_r\left[\theta_2,\vartheta_2,\pi\left(1+p_{n_{\varphi_2},k}\right)\right],\\ d\boldsymbol{l}_{i,k}&=\left(R+\frac{a}{2}\left(1+p_{n_{a_2},i}\right)\right)\widehat{\boldsymbol{T}}_r\left[\theta_2,\vartheta_2,\pi\left(1+p_{n_{\varphi_2},k}\right)\right].\end{aligned} \tag{85}$$

$$M_{12}=2\pi\frac{N_2}{I_1}\sum_i^{n_a}\frac{w_{n_{a_2},i}}{2}\sum_j^{n_b}\frac{w_{n_{b_2},j}}{2}\sum_k^{n_\varphi}\frac{w_{n_{\varphi_2},k}}{2}\left(\boldsymbol{A}_1\left[\boldsymbol{r}_{i,j,k}\right]\cdot d\boldsymbol{l}_{i,k}\right), \tag{86}$$

$$\boldsymbol{F}_{12}=2\pi N_2 I_2\sum_i^{n_a}\frac{w_{n_{a_2},i}}{2}\sum_j^{n_b}\frac{w_{n_{b_2},j}}{2}\sum_k^{n_\varphi}\frac{w_{n_{\varphi_2},k}}{2}\left(d\boldsymbol{l}_{i,k}\times\boldsymbol{B}_1\left[\boldsymbol{r}_{i,j,k}\right]\right), \tag{87}$$

$$\boldsymbol{\tau}_{12}=2\pi N_2 I_2\sum_i^{n_a}\frac{w_{n_{a_2},i}}{2}\sum_j^{n_b}\frac{w_{n_{b_2},j}}{2}\sum_k^{n_\varphi}\frac{w_{n_{\varphi_2},k}}{2}\left(\mathbf{r}_{i,j,k}-\boldsymbol{r}_2\right)\times\left(d\boldsymbol{l}_{i,k}\times\boldsymbol{B}_1\left[\boldsymbol{r}_{i,j,k}\right]\right). \tag{88}$$

Navedene trostruke sume u sebi zapravo kriju još dvostruke sume za izračun samoga potencijala, odnosno polja u zadanim točkama, pa su sume efektivno peterostruke. Korištenje fiksnog broja inkremenata za svaki sloj se pokazalo iznimno neefikasnim - niti brzina niti preciznost nisu bile blizu željene razine. Rješenje ovoga problema će se detaljno razmotriti u sljedećem odjeljku 3.5.

3.5 Automatsko balansiranje broja inkremenata

Razmatranje optimizacije numeričke integracije preko Gauss-Legendre kvadrature započinje odabirom koja zavojnica je primar a koja sekundar. Ako se radi o kombinaciji različitih tipova zavojnica, pravokutna zavojnica ima uvijek prednost pred ostalim tipovima, tanka zavojnica ima prednost pred plosnatom, a tanka i plosnata pred petljom. Ako su zavojnice istog tipa a različitih dimenzija opet se preporučuje veću koristiti kao primar. Veličinu je moguće dobro procijeniti umnoškom $R\dot{a}\dot{b}$. Ovaj pristup pretpostavlja da precizno izračunato polje ima veći prioritet od preciznosti izračuna suma međuindukcije (86), sile (87) i zakretnog momenta (88). Ova jednostavna hijerarhija se empirijski pokazala učinkovitim kod asimetričnih sustava zavojnica jer smanjuje pogrešku izračuna, nekad i za više od reda veličine (uz jednako korištenje računalnih resursa).

Nakon izbora koja zavojnica je primar, potrebno je odrediti koliko inkremenata, odnosno koji red Gauss-Legendre sume koristimo za svaki od slojeva n_{a_1}, n_{b_1}, n_{φ_1}, n_{a_2}, n_{b_2} i n_{φ_2}. Navedeno će se nazivati brojem inkremenata umjesto redom sume zbog implementacijskih razloga - broj inkremenata u *for* petlji. Iako je sloj n_{b_1} uklonjen iz jednadžbi za polje, razmatrat će se zbog potpunosti. Većina autora čiji je doprinos analiziran u uvodu 1 koristila je uvijek jednak broj inkremenata za svaku dimenziju integriranja, bez obzira na mjeru te dimenzije. Ako uzimamo u obzir da je korištenje kvadrature reda n zapravo aproksimacija segmenta funkcije polinomom reda $2n-1$ jasno je da veći interval znači manju preciznost za isti broj inkremenata.

Započinje se definiranjem faktora preciznosti kojeg ćemo označiti s m. Kako svaka suma sadrži svoj skup točaka, onda umnožak svih slojeva odgovara Kartezijevom umnošku skupova, a ukupan broj točaka umnošku broja elemenata skupova. Ako je na raspolaganju veći ukupni broj inkremenata moguće je više inkremenata dati određenoj sumi, odnosno sloju integracije. Osnovni broj inkremenata je postavljen na 10 po svakom sloju integracije, odnosno ukupno 10^k, gdje k označava broj slojeva integracije. Tada je faktor preciznosti realan broj iz intervala $[1.0, 16.0]$, a ukupan broj inkremenata je dan izrazom

$$n_t = n_{a_1} n_{b_1} n_{\phi_1} n_{a_2} n_{b_2} n_{\phi_2} \approx 10^k \cdot 2^{m-1}. \tag{89}$$

Dakle povećavanje faktora preciznosti za 1.0 udvostručuje se ukupan broj inkremenata. Ovakva definicija omogućuje veliku kontrolu nad preciznošću izračuna i vremenom koje je korisnik spreman utrošiti. Važno je uočiti znak $\approx$ koji govori da se radi o približnoj relaciji, a navedeno će postati jasnije kada se opiše algoritam.

Za integriranje po duljini i debljini je vrlo prirodno definirati mjeru veličine koraka (u literaturi se to redovito naziva *step size*) kao omjer a_1/n_{a_1} odnosno b_1/n_{b_1}. Za kut ne možemo definirati direktno mjeru koja ima dimenziju metra, pa bi se kao približna mjera mogao koristiti omjer srednje duljine luka i broja inkremenata $\pi(a_1 + R_1/2)/n_{\varphi_1}$. Iako bi ovakve definicije bile vrlo intuitivne i jednostavne, u

praksi ne daju vrlo dobre rezultate. Cilj je definirati efektivnu veličinu koraka koja omogućuje direktno uspoređivanje preciznosti integracije različitih slojeva. Dakle kada su svi efektivni koraci približno jednaki (90), postignuta je ravnoteža. Efektivne veličine koraka definiramo izrazima definiramo ju izrazima (91) i (92)

$$\lambda_{a_1} \approx \lambda_{b_1} \approx \lambda_{\varphi_1} \approx \lambda_{a_2} \approx \lambda_{b_2} \approx \lambda_{\varphi_2}, \tag{90}$$

$$\lambda_{a_1} = \frac{\sqrt{2a_1}}{n_{a_1}}, \quad \lambda_{b_1} = \frac{\sqrt{2b_1}}{n_{b_1}}, \quad \lambda_{\varphi_1} = \frac{1}{n_{\varphi_1}}\sqrt{2\pi\left(R_1 + \frac{a_1}{2}\right)}, \tag{91}$$

$$\lambda_{a_2} = \frac{\sqrt{a_2}}{n_{a_2}}, \quad \lambda_{b_2} = \frac{\sqrt{b_2}}{n_{b_2}}, \quad \lambda_{\varphi_2} = \frac{1}{n_{\varphi_2}}\sqrt{2\pi\left(R_2 + \frac{a_2}{2}\right)}. \tag{92}$$

Ovakva definicija efektivne veličine koraka se pokazala najboljom od svih koje su isprobane te znatno boljom od intuitivne. Prethodna, intuitivna definicija bi govorila da je u slučaju $a : b = 1 : 100$ optimalno inkremente raspodijeliti u omjeru $n_a : n_b = 1 : 100$. Ovo se nije pokazalo optimalnim; znatno boljim se pokazao omjer $n_a : n_b = 1 : 10$. Relacija $\sqrt{a} : \sqrt{b} \approx n_a : n_b$ vrlo dobro određuje optimalnu raspodjelu inkremenata pa definicije (91) i (92) iz nje direktno slijede.

Preostaje razjasniti dodatni faktor 2 u (91). Radi se o još jednom empirijskom zaključku. Imajući na umu da precizno izračunato polje ima velik utjecaj na preciznost ukupnog izračuna, povećan je broj inkremenata iskorišten na primaru. Stoga se umjesto izmjene uvjeta (90), primarnim veličinama koracima dodaje težinski faktor. Empirijski je određeno da optimalni faktor iznosi približno 1.4 pa je zaokružen na $\sqrt{2}$. Ova izmjena je poboljšala preciznost većine rezultata za nekoliko redova veličine. Najvažnije je veliko poboljšanje preciznosti izračuna međuindukcije te omogućavanje izračuna samoindukcije. Kao što je spomenuto, suma za izračun samoindukcije (81) nije stabilna, no pokazalo se da omjer $\approx \sqrt{2}$ omogućuje najboljom konvergencijom dok omjer 1 rezultira divergencijom.

Algoritam započinje postavljanjem broja inkremenata svih slojeva na 1. Cilj je osmisliti jedan algoritam koji radi za sve tipove zavojnica i sve vrste proračuna - univerzalni recept. Ima ukupno 3 različite vrste proračuna: metode za polja, metode za slučaj zajedničke osi i metode za općeniti slučaj. Navedene slučajeve razdvaja broj slojeva integracije: metode za izračun polja zahtijevaju najviše dva, zajedničke osi najviše tri, a općeniti slučaj najviše pet. U prvom slučaju je potrebno samo balansirati broj inkremenata za jednu zavojnicu.

Uzimajući u obzir sve dosad navedeno algoritam je osmišljen ovako:

1. Postavi broj svih inkremenata na 1
2. Odredi koja zavojnica ima veći ukupni korak
3. Odredi koja dimenzija ima najveći korak
4. Uvećaj broj inkremenata korišten za tu dimenziju
5. Provjeri je li ukupan broj inkremenata veći od $10^k \cdot 2^{m-1}$
6. Ako je ukupan broj inkremenata manji, ponovi korake 2-6. Ako nije, završi balansiranje.

Efektivna veličina koraka, odnosno skraćeno korak, je već opisan, a potrebno je definirati ukupni korak koji možemo pridružiti cijeloj zavojnici na temelju triju osnovnih koraka. Umnožak koraka nije praktičan zato što određeni tipovi zavojnica imaju zanemarive dimenzije po duljini, odnosno debljini. Stoga definiramo linearne korake koji objedinjuju korake po duljini i debljini, dane izrazom (93). Ukupni korak je samo $\lambda_{l_1} + \lambda_{\varphi_1}$. Dodatni razlog za uvođenje linearnog koraka je određivanje hijerarhije slojeva unutar same zavojnice - kutni korak ima prednost pred linearnim korakom

$$\lambda_{l_1} = \frac{1}{2}\left(\lambda_{a_1} + \lambda_{b_1}\right), \quad \lambda_{l_2} = \frac{1}{2}\left(\lambda_{a_2} + \lambda_{b_2}\right). \tag{93}$$

Kako je algoritam implementiran u programskom jeziku korištenjem *if-else* naredbi, najbolje je zapisati ga u logičkom jeziku. Uvedene su pokrate (94) koje predstavljaju uvjete, odnosno logičke varijable (bool). Kombinacije ovih uvjeta omogućuju odabir sloja čiji će broj inkremenata biti uvećan za 1 u trenutnom prolasku petlje.

$$\begin{gathered} c_1 = \lambda_{l_1} + \lambda_{\varphi_1} \geq \lambda_{l_2} + \lambda_{\varphi_2} \\ c_{2,1} = \lambda_{\varphi_1} \geq \lambda_{l_1}, \quad c_{2,2} = \lambda_{\varphi_2} \geq \lambda_{l_2} \\ c_{3,1} = \lambda_{a_1} \geq \lambda_{b_1}, \quad c_{3,2} = \lambda_{a_2} \geq \lambda_{b_2} \end{gathered} \tag{94}$$

Zapisuje se samo algoritam za općeniti slučaj. Algoritam za slučaj zajedničke osi razlikuje se samo u uvjetu prekida petlje koji je tada $n_{\varphi_1} n_{a_1} n_{a_2} < 10^3 \cdot 2^{m-1}$. Za slučaj polja algoritam je trivijalan i svodi se na ponovljeno ispitivanje $c_{2,1}$ i $c_{3,1}$ te provjeru $n_{\varphi_1} n_{a_1} < 10^2 \cdot 2^{m-1}$ na kraju petlje. Kao što je rečeno, broj inkremenata n_{b_1} se računa zbog logičke konzistentnosti, ali se ne koristi direktno.

$$\begin{gathered} c_1 \wedge c_{2,1} \longrightarrow \text{uvećaj } n_{\varphi_1}, \quad c_1 \wedge \neg c_{2,1} \wedge c_{3,1} \longrightarrow \text{uvećaj } n_{a_1} \\ c_1 \wedge \neg c_{2,1} \wedge \neg c_{3,1} \longrightarrow \text{uvećaj } n_{b_1}, \quad \neg c_1 \wedge c_{2,2} \longrightarrow \text{uvećaj } n_{\varphi_2} \\ \neg c_1 \wedge \neg c_{2,2} \wedge c_{3,1} \longrightarrow \text{uvećaj } n_{a_2}, \quad \neg c_1 \wedge \neg c_{2,1} \wedge \neg c_{3,1} \longrightarrow \text{uvećaj } n_{b_2} \\ \text{ponavljaj dok:} \quad n_{\varphi_1} n_{a_1} n_{\varphi_2} n_{a_2} n_{b_2} < 10^5 \cdot 2^{m-1} \end{gathered} \tag{95}$$

Ovime je završeno razmatranje predložene metode izračuna polja, međuindukcije, sile te zakretnog momenta. Broj slojeva integracije je znatno veći od konkurentnih metoda koje redovito koriste svega jedan ili dva sloja integracije. Predloženi su relativno jednostavni izrazi s malenim brojem kompliciranih funkcija te upravo opisani algoritam za efikasno alociranje broja inkremenata po višestrukim slojevima integracije. Ovaj algoritam je bio zadnji komadić slagalice i doveo je preciznost metode na razinu 15 značajnih znamenki u slučajevima kada dvije zavojnice nisu tik jedna do druge. Faktor preciznosti je također znatno olakšao određivanje željene preciznosti izračuna i procjenu vremena izračuna.

4 Implementacija pristupa

4.1 Osnovno o strukturi programskog koda

Sada kada su matematički izrazi izvedeni i zapisani u obliku pogodnom za programsku implementaciju potrebno je opisati osnovnu strukturu programskog rješenja. Kako je već više puta rečeno, performanse konačnog programa i jednostavnost izraza su nužni uvjeti za djelotvornu programsku izvedbu pristupa. Ako bismo opisani pristup implementirali u jeziku koji nije prevođen izgubili bismo većinu prednosti koje pristup nudi. Stoga je nužno koristiti programski jezik C++ i posvetiti znatnu količinu vremena kvalitetnoj implementaciji svih potrebnih sastavnica. Konačni cilj je navedeni C++ kod prevesti u Python modul tako da ciljani korisnici mogu uživati gotovo jednaku performansu C++ jezika u jeziku kojega izgledno koriste u svakodnevnomu radu.

Kako se nastoji modelirati zavojnica, fizički objekt kojega smo dobro opisali odabranim svojstvima, prirodno je koristiti paradigmu objektno orijentiranog programiranja. Svi nazivi su na engleskome jeziku zbog lakšeg programiranja i dostupnosti široj, međunarodnoj akademskoj i stručnoj zajednici. Razred zavojnica, odnosno class `Coil`, je prilično kompliciran pa ćemo se u ovome odjeljku zadržati samo na osnovnim elementima, a neke implementacijske detalje istaknuti u sljedećem odjeljku. Uz razred `Coil` definirano je nekoliko pomoćnih modula, a neki od njih sadrže dodatne razrede radi olakšanja implementacije i uporabe metoda uključenih u `Coil`. Moduli su abecednim redom: `Benchmark`, `Coil`, `CoilGroup`, `Compare`, `CUDAFunctions`, `LegendreMatrix`, `Tensor`, `Test`, `Threadpool` i `Utils`.

U modulu `Benchmark` nalaze se funkcije za testiranje performansi metoda za izračun polja, međuindukcije, sile i zakretnog momenta. Pritom uvodimo tri različita načina izračuna koji nude različite razine performansi, a označeni su enumeracijom `ComputeMethod`. Tri načina su redom obično izvođenje na jednoj jezgri (`CPU_ST`), izvođenje na više dretvi (`CPU_MT`) te grafički ubrzano izvođenje (`GPU`). Ako se odjednom poziva izračun većeg broja točaka jasno je da se radi o međusobno nezavisnim proračunima pa je vrlo jednostavno paralelizirati izvođenje. `Threadpool` služi za vezu prema vanjskom modulu CTPL [24] koji se koristi za implementaciju višedretvenosti.

Korištenje `CPU_MT` donosi poboljšanje performansi čim je broj točaka nešto veći od broja dretvi koji se želi koristiti u izračunu. Pritom je važno naglasiti da je pogodno da svaka dretva računa što veći broj točaka kako bi se smanjio utjecaj dodatnog troška pokretanja zadatka (*overhead*). Za podjelu ukupnog broja točki na broj blokova jednak broju dretvi korišten je jednostavan algoritam. Svi blokovi se veličinom razlikuju za najviše 1. Ovakva podjela je u teoriji optimalna, no presudno je kako operacijski sustav daje resurse dretvama. Višedretvenost se primjenjuje i na višoj razini izračuna ako se za to pruži prilika - izračun međuindukcije para zavojnica za velik broj različitih konfiguracija. Ovakvu specifičnu upotrebu višedretvenosti, koja nije primijenjena na razini izračuna vrijednosti polja u točki,

nazivamo raspodijeljenom višedretvenosti, skraćeno `MTD` (*Multithreading Distributed*). U literaturi se ovakav pristup paralelizaciji najčešće naziva grubozrnatom višedretvenosti. Ona je implementirana kao poseban slučaj unutar `CPU_MT`. `MTD` se također koristi ako je broj zavojnica u objektu `CoilGroup` znatno veći od broja dretvi. Korištenje `MTD` pristupa uvijek povećava performanse u usporedbi s korištenjem obične višedretvenosti. Opisane optimizacije omogućuju efikasno opterećenje velikog broja fizičkih procesorskih jezgri. Performanse metoda izračuna polja će vrlo detaljno biti razmotrene u odjeljku 5.2.

Grafička akceleracija izvođenja (`GPU`) je još uvijek u ranoj fazi razvoja, no i rani rezultati su dovoljno impresivni da ovakav način izračuna ima svoju primjenu u kontekstu trenutnog rada. Grafička kartica je dizajnirana za masivno paralelno procesuiranje podataka, a način na koji su naše funkcije osmišljene je prikladan za primjenu ove paradigme. Valja pritom imati na umu da je potrebno zadati veliki broj točaka (veći od broja jezgri) kako bi grafička kartica bila potpuno opterećena - trošak pokretanja izračuna i prijenosa podataka je visok. Primjer takvog slučaja je izračun polja zavojnice u dovoljnom broju točaka prostora za detaljan slikovni prikaz polja. U takvome slučaju grafička kartica može biti od deset do stotinu puta brža od `CPU_MT` metode, no nedostatak je korištenje jednostrukte preciznosti float. Valja napomenuti da najskuplje kartice imaju mogućnost efikasno koristiti i dvostruku preciznost (double). Za grafičku karticu su implementirane samo brze metode za izračun polja.

Modul `Comparison` sadrži funkcije koje ispisuju rezultate izračuna posebnih konfiguracija zavojnica korištenih pri ispitivanju preciznosti našeg pristupa. Dakle svi podatci koji su prikazani u odjeljcima 5.3 i 5.4 mogu se dobiti korištenjem funkcija iz navedenog modula. Za testiranje funkcionalnosti C++ koda napravljen je modul `Test` koji nije vidljiv u Python modulu. Python modul je zapravo omotač oko C++ aplikacijskog programskog sučelja, a napravljen je korištenjem programskog alata PyBind11 [23].

Moduli `LegendreMatrix` i `Tensor` su napravljeni za jednostavnije korištenje programa te lakšu implementaciju matematičkih operacija. `LegendreMatrix` je upravo ono što ime sugerira, modul koji sadrži dvije statičke donje trokutaste matrice (`positionMatrix` i `weightsMatrix`) koje nose informaciju o pozicijama i težinskim vrijednostima prvih 100 redova Gauss-Legendre kvadrature. Red matrice `positionMatrix` s indeksom 0 sadrži jednu vrijednost koja odgovara poziciji točke kvadrature prvog reda, red s indeksom 1 sadrži vrijednosti dvije točke za kvadraturu drugog reda, itd.

`Tensor` sadrži tri objekta: `CoordVector3`, `FieldVector3` i `Matrix3`. `CoordVector3` je prilagodljivi radijvektor - moguće je postaviti mu vrijednost u Kartezijevim, cilindričnim ili sfernim koordinatama, a koristi se kao argument za poziciju točke pri izračunu polja i postavljanje prostorne pozicije zavojnice. Ovime je povećana fleksibilnost unosa i ugrađena pretvorba u cilindrične koordinate korištene u gornjim izrazima. `FieldVector3` predstavlja običan vektor s definiranim skalarnim i vektorskim umnoškom. `Matrix3` je matrica 3x3 uz definirano matrično zbrajanje i množenje te primjenu linearne transformacije nad objektom `FieldVector3`.

Tijekom razvoja potpore za Windows implementiran je još jedan modul `CustomMath` koji nije uključen u konačni kod. Razlog zašto `CustomMath` ipak nije korišten je nužnost korištenja Microsoft MSVC prevoditelja za izradu Python modula. Na početku je projekt razvijan korištenjem MinGW64 prevoditelja koji ima jedan veliki nedostatak - spore ugrađene matematičke funkcije (MSVC nema ovaj problem). Stoga su implementirane vlastite funkcije $\cos(x)$ i $\ln(x)$ jer su često korištene. Uočeno je da MinGW64 s tim funkcijama u mnogim slučajevima pokazuje bolje performanse od MSVC prevoditelja. Nakon dodatnog testiranja utvrđeno je da dvodimenzionalni `vector` u našem slučaju radi znatno bolje od jednodimenzionalnog i nakon te promjene MSVC je postigao performanse slične onima na Ubuntu. Ponašanje na operacijskom sustavu Ubuntu je bilo vrlo predvidljivo i nije bilo sličnih problema. Konačno, modul `Utils` sadrži funkcije za mjerenje vremena koje se koriste za profiliranje performansi kernela i koda napisanog za grafičku karticu.

Razmotrivši sve navedeno možemo proučiti detaljnije glavni razred - `Coil`. Specifično će biti prikazana znatno reducirana verzija sučelja `Coil.h`. Otprilike tri četvrtine koda su izostavljene kako bi se zadržala razumna duljina, a zadržani su dijelovi od posebnog interesa zato što predstavljaju deklaraciju prethodno razrađenih izraza. Definirane su pomoćne strukture podataka `PrecisionFactor`, `PrecisionArguments` i `CoilPairArguments` koje služe za definiranje preciznosti i broja inkremenata po određenom sloju integracije kako je diskutirano u odjeljku 3.5. Nešto preciznije ćemo implementaciju tih metoda objasniti u sljedećem odjeljku 4.2, no dostatno je reći da `PrecisionArguments` nosi informaciju o broju inkremenata korištenom za izračun polja, a `CoilPairArguments` informaciju o broju inkremenata korištenom za izračun međuindukcije, sile i zakretnog momenta. Obje strukture sadrže i pripadnu metodu za generiranje argumenata uz danu primarnu i sekundarnu zavojnicu, faktor preciznosti i metodu izračuna. U slučaju grafičke kartice broj inkremenata je fiksan.

Konačno se dolazi do prikaza objekta `Coil`. Osnovne karakteristike zavojnice su `innerRadius`, `thickness`, `length` i `numOfTurns` i one su nepromjenjive. Ostala svojstva objekta poput `current`, `wireResistivity`, `sineFrequency`, `PrecisionFactor` imaju definiranu pripadnu pretpostavljenu vrijednost, za jakost struje to je 1 A (preko nje se automatski izračunava `currentDensity`), za otpornost je uzeta otpornost bakra, za frekvenciju 50 Hz, dok se za faktor preciznosti koristi pretpostavljeni konstruktor koji `relativePrecision` postavlja na 5. Pretpostavljena vrijednost radijvektora `positionVector` je također implementirana pretpostavljenim konstruktorom koji postavlja zavojnicu u ishodište. Kutovi usmjerenja zavojnice `yAxisAngle` i `zAxisAngle` su pretpostavljeni na 0, a `threadCount` 8. Ostala svojstva, poput `transformationMatrix` izračunavaju se preko prethodno spomenutih varijabli. Za dohvat i postavljanje atributa napisane su pripadne metode za postavljanje i dohvaćanje vrijednosti. Važno je uočiti enumeraciju `CoilType` koja određuje pripadni tip zavojnice i ovisno o njoj se postavlja logička varijabla `useFastMethod`.

```
// Kratki pregled kljucnih dijelova datoteke Coil.h

#include "ComputeMethod.h"
#include "CoilType.h"
#include "Tensor.h"
#include "PrecisionGlobalVars.h"

const int precisionArraySize = 864;
const int defaultThreadCount = 8;

const extern int blockPrecisionCPUArray[precisionArraySize];
const extern int incrementPrecisionCPUArray[precisionArraySize];

struct PrecisionFactor
{
    PrecisionFactor();
    explicit PrecisionFactor(double relativePrecision);

    double relativePrecision;

    explicit operator std::string() const;
};

struct PrecisionArguments
{
    PrecisionArguments();
    explicit PrecisionArguments(int angularBlocks, int thicknessBlocks, int lengthBlocks, ↩
        int angularIncrements, int thicknessIncrements, int lengthIncrements);

    int angularBlockCount;
    int thicknessBlockCount;
    int lengthBlockCount;

    int angularIncrementCount;
    int thicknessIncrementCount;
    int lengthIncrementCount;

    static PrecisionArguments
    getCoilPrecisionArgumentsCPU(const Coil &coil, PrecisionFactor precisionFactor);
};

struct CoilPairArguments
{
    CoilPairArguments();
    explicit CoilPairArguments(const PrecisionArguments &primaryPrecision,
                               const PrecisionArguments &secondaryPrecision);

    PrecisionArguments primaryPrecision;
```

```
48      PrecisionArguments secondaryPrecision;
49
50      static CoilPairArguments
51      getAppropriateCoilPairArguments(const Coil &primary, const Coil &secondary, ↩
            PrecisionFactor precisionFactor, ComputeMethod computeMethod = CPU_ST, bool ↩
            zAxisCase = true);
52
53      private:
54          static CoilPairArguments
55          calculateCoilPairArgumentsCPU(const Coil &primary, const Coil &secondary, ↩
                PrecisionFactor precisionFactor, bool zAxisCase = false);
56  };
57
58  class Coil
59  {
60      private:
61          unsigned long long id;
62
63          double innerRadius;
64          double thickness;
65          double length;
66          int numOfTurns;
67
68          double currentDensity{};
69          double current{};
70
71          double wireResistivity{};
72          bool sineDriven{};
73          double sineFrequency{};
74
75          double magneticMoment{};
76          double averageWireThickness{};
77
78          double resistance{};
79          double selfInductance{};
80          double reactance{};
81          double impedance{};
82
83          CoilType coilType{};
84          bool useFastMethod{};
85          int threadCount{};
86
87          PrecisionArguments defaultPrecision;
88
89          vec3::CoordVector3 positionVector{};
90          double yAxisAngle{};
91          double zAxisAngle{};
92          vec3::Matrix3 transformationMatrix{};
93          vec3::Matrix3 inverseTransformationMatrix{};
```

```
94
95      public:
96          Coil(double innerRadius, double thickness, double length, int numOfTurns, double ↩
                current, double wireResistivity, double sineFrequency, PrecisionFactor ↩
                precisionFactor = PrecisionFactor(), int threadCount = defaultThreadCount, vec3↩
                ::CoordVector3 coordinatePosition = vec3::CoordVector3(), double yAxisAngle = ↩
                0.0, double zAxisAngle = 0.0);
97
98          [[nodiscard]] vec3::FieldVector3
99          computeAPotentialVector(vec3::CoordVector3 pointVector, const PrecisionArguments &↩
                usedPrecision) const;
100
101         [[nodiscard]] vec3::FieldVector3
102         computeBFieldVector(vec3::CoordVector3 pointVector, const PrecisionArguments &↩
                usedPrecision) const;
103
104         [[nodiscard]] vec3::Matrix3
105         computeBGradientTensor(vec3::CoordVector3 pointVector, const PrecisionArguments &↩
                usedPrecision) const;
106
107         [[nodiscard]] std::vector<vec3::FieldVector3>
108         computeAllAPotentialComponents(const std::vector<vec3::CoordVector3> &pointVectors, ↩
                const PrecisionArguments &usedPrecision, ComputeMethod computeMethod = CPU_ST) ↩
                const;
109
110         [[nodiscard]] std::vector<vec3::FieldVector3>
111         computeAllBFieldComponents(const std::vector<vec3::CoordVector3> &pointVectors, ↩
                const PrecisionArguments &usedPrecision, ComputeMethod computeMethod = CPU_ST) ↩
                const;
112
113         [[nodiscard]] std::vector<vec3::Matrix3>
114         computeAllBGradientTensors(const std::vector<vec3::CoordVector3> &pointVectors, ↩
                const PrecisionArguments &usedPrecision, ComputeMethod computeMethod = CPU_ST) ↩
                const;
115
116         static double
117         computeMutualInductance(const Coil &primary, const Coil &secondary, PrecisionFactor ↩
                precisionFactor = PrecisionFactor(), ComputeMethod computeMethod = CPU_ST);
118
119         static std::pair<vec3::FieldVector3, vec3::FieldVector3>
120         computeAmpereForce(const Coil &primary, const Coil &secondary, PrecisionFactor ↩
                precisionFactor = PrecisionFactor(), ComputeMethod computeMethod = CPU_ST);
121
122         [[nodiscard]] std::pair<vec3::FieldVector3, vec3::FieldVector3>
123         computeForceOnDipoleMoment(vec3::CoordVector3 pointVector, vec3::FieldVector3 ↩
                dipoleMoment, const PrecisionArguments &usedPrecision) const;
124         // ...
125 };
```

Preostaje još razmotriti javne metode dostupne korisniku. Ovdje je naveden samo malen podskup dostupnih metoda, samo one najvažnije te su izostavljene preopterećene metode. Korisnik može ručno specificirati preciznost (direktnim zadavanjem `PrecisionArguments usedPrecision`) ili prepustiti algoritmu da za dani faktor preciznosti automatski izračuna pripadnu preciznost. Što se tiče implementacije, metoda s argumentom tipa `PrecisionFactor` izračunava pripadni `usedPrecision` i zatim poziva metodu s argumentom `PrecisionArguments`. Uz ovu klasu metoda dostupnih korisniku koje nazivamo `compute` metode, postoje definirane `calculate` metode koje su privatne i sadrže funkcije za prave izračune. No za praktične svrhe, svaka od `compute` metoda utjelovljuje jedan od izraza prethodno izvedenih u poglavlju 3. Ovisno o korištenom tipu zavojnice, metoda `computeAPotentialVector` implementira izraze (52) ili (59), `computeBFieldVector` izraze (53) i (60), a metoda `computeBGradientTensor` matricu (37) ili (42) te niz izraza (54), (55), (56), (57) ili (61), (62), (63), (64). Pritom se primjenjuju transformacije opisane izrazima (65), (66) i (67). Ove metode računaju polje za samo jednu točku i izvode se na samo jednoj dretvi. Definiraju se i `computeAll` metode koje primjenjuju operaciju izračuna na velik broj točaka, pa primaju listu objekata tipa `CoordVector3` i imaju omogućeno korištenje različitih metoda izračuna. Korištenje `CPU_MT` gotovo uvijek daje poboljšanje performansi, a upravo performanse ovih metoda bit će detaljno prikazane u odjeljku 5.2. Metoda `computeForceOnDipoleMoment` može se koristiti za izračun sile i zakretnog momenta na česticu prema izrazima (9) i (10). Dodatno, ako su dvije zavojnice dovoljno daleko, ili je jedna znatno manja od druge, onda možemo izračunati magnetski moment manje zavojnice i koristiti ovu metodu. Pritom je važno imati na umu da se radi o aproksimaciji.

Na kraju još valja opisati statičke metode koje djeluju nad parovima zavojnica - metode za izračun međuindukcije, sile i zakrenog momenta. Obje `compute` metode u pozadini imaju implementirane posebne `calculate` metode, kao i metode za izračun polja. Također imaju definiran poseban slučaj zajedničke osi ako su oba kuta rotacije postavljena na 0 te oba središta zavojnica leže na z-osi. U slučaju zajedničke osi, ovisno o tipu zavojnice, metoda `computeMutualInductance` implementira izraze (74) ili (80), odnosno u općenitom slučaju izraz (86). Pritom valja naglasiti da se prvo generira lista točaka u kojima je potrebno izračunati vektorski potencijal a zatim se poziva metoda `computeAllAPotentialComponents`. Metoda `computeAmpereForce` vraća par vrijednosti tipa `FieldVector` od kojih prva predstavlja silu, a druga zakretni moment. Ona u slučaju zajedničke osi implementira izraz (75), a u općenitom slučaju izraz (87). Postoje i `computeAll` verzije ovih metoda, koje ovdje nisu prikazane. One kao argumente dodatno primaju liste položaja središta zavojnica i kutova rotacije, a mogu koristiti i `MTD` paradigmu.

Preostaje još detaljnije promotriti konkretnu implementaciju jedne od metoda za izračun polja te razjasniti globalne varijable `blockPrecisionCPUArray` i `incrementPrecisionCPUArray`, a navedeno će biti obrađeno u sljedećem odjeljku 4.2. Nazivi metoda u Python modulu jednaki su onima u C++, samo koriste `snake_case` umjesto `camelCase`.

4.2 Važniji implementacijski detalji

U ovome odjeljku će biti prikazana konkretna implementacija jedne od `calculate` metoda, specifično brza metoda za izračun vektorskog potencijala dana izrazom (59). Implementacije ostalih metoda su iznimno slične i oslanjaju se na ista načela. Prva stvar koju je potrebno razjasniti je podjela na broj inkremenata i broj blokova. Ova podjela nije prethodno diskutirana zato što u idealnom svijetu ne bi bila potrebna, no u stvarnosti je potrebno realizirati matrice sa što manjim brojem elemenata kako bi stale u privremenu memoriju što niže razine razine - navedeno je ključno za dobre performanse. Stoga su matrice u modulu `LegendreMatrix` dimenzija 100×100, dakle pokrivaju Gauss-Legendre kvadraturu do reda 100. Ovo bi trebalo biti dostatno za skoro sve slučajeve u kojima je primjena našeg pristupa izračunu opravdana, no kako bi se ostvarila dodatna fleksibilnost domena integracije koja podrazumijeva k koraka se može podijeliti na p blokova po l inkremenata tako da vrijedi $k = p \times l$. Dakle umjesto primjene kvadrature reda k na cijeli interval, primjenjuje se kvadratura reda $l, l \in 1, 2, ..., 100$ na p podintervala domene integracije. Ovo omogućuje praktično neograničeno povećanje broja inkremenata do granica mogućnosti izračuna, no zbog praktičnih razloga ćemo ga ipak ograničiti na $p \in 1, 2, ..., 10000$, što bi trebalo biti više nego dostatno za sve primjene.

Uočimo da prethodna promjena definicije mijenja algoritam za automatsko balansiranje broja inkremenata definiran izrazom (95). Potrebno je navedeni algoritam modificirati tako da napravimo dvije liste, koje su globalne varijable, a koje će pohranjivati sve dozvoljene kombinacije broja blokova i inkremenata. Prije se mogla dobiti kvadratura reda 101, a sada se mogu ostvariti samo dvije kvadrature reda 51 što znači da se u petljama pri izračunu izvode efektivno 102 koraka. Umjesto da n_{φ_1}, n_{a_1}, n_{b_1}, n_{φ_2}, n_{a_2} i n_{b_2} predstavljaju sam broj inkremenata, oni će predstavljati poziciju (indeks) u listama. Liste su definirane izrazima (96) i (97), a implementirane kao `incrementPrecisionCPUArray` odnosno `blockPrecisionCPUArray`. Dodatno je prikazana lista (98) koja nije implementirana ali prikazuje dozvoljene ukupne brojeve inkremenata

$$L_{\text{inc}} = \{1,\ 2,\ 3, \ldots,\ 99,\ 100,\ 51,\ 52, \ldots 99,\ 100,\ 68,\ 69,\ \ldots,\ \ 100,\ 76,\ 77, \ldots,\ 100,\ 81, \ldots\}, \tag{96}$$

$$L_{\text{bl}} = \{1,\ 1,\ 1, \ldots 1,\ 1,\ 2,\ 2, \ldots 2,\ 2,\ 3,\ 3, \ldots,\ 3,\ 4,\ 4, \ldots,\ 5,\ 6, \ldots\}, \tag{97}$$

$$L_{\text{it}} = \{1,\ 2,\ 3, \ldots 99,\ 100,\ 102,\ 104, \ldots, 198,\ 200,\ 204,\ 207,\ \ldots,\ \ 300,\ 304,\ 308, \ldots,\ 400,\ 405, \ldots\}. \tag{98}$$

Dodatna pogodnost korištenja ovakve implementacije je brže izvođenje algoritma (95). Brže će se izvoditi zato što je potrebno znatno manje prolaza petlje za dolazak do većeg broja koraka i broj prolaza će sigurno biti manji od 4320 (liste sadrže `precisionArraySize = 864` elemenata). Valja samo naglasiti da se sada uvjet ponavljanja petlje formulira preko izraza

$$c = L_{\text{bl}}[n_{\varphi_1}] L_{\text{inc}}[n_{\varphi_1}] L_{\text{bl}}[n_{a_1}] L_{\text{inc}}[n_{a_1}] L_{\text{bl}}[n_{\varphi_2}] L_{\text{inc}}[n_{\varphi_2}] L_{\text{bl}}[n_{a_2}] L_{\text{inc}}[n_{a_2}] L_{\text{bl}}[n_{b_2}] L_{\text{inc}}[n_{b_2}]. \tag{99}$$

```
double Coil::calculateAPotentialFast(double zAxis, double rPolar, const PrecisionArguments &↩
    usedPrecision) const
{
    double magneticPotential = 0.0;

    double thicknessBlock = thickness / usedPrecision.thicknessBlockCount;
    double angularBlock = M_PI / usedPrecision.angularBlockCount;

    int thicknessIncrements = usedPrecision.thicknessIncrementCount - 1;
    int angularIncrements = usedPrecision.angularIncrementCount - 1;

    double constant = g_MiReduced * currentDensity * thicknessBlock * angularBlock * 0.5;

    double topEdge = zAxis + length * 0.5;
    double bottomEdge = zAxis - length * 0.5;

    std::vector<std::vector<double>> cosPhiPrecomputeMat(usedPrecision.angularBlockCount);

    for (int indBlockPhi = 0; indBlockPhi < usedPrecision.angularBlockCount; ++indBlockPhi)
    {
        double blockPositionPhi = angularBlock * (indBlockPhi + 0.5);
        cosPhiPrecomputeMat[indBlockPhi].reserve(angularIncrements);

        for (int incPhi = 0; incPhi < angularIncrements; ++incPhi)
        {
            double incrementPositionFi = blockPositionPhi + (angularBlock * 0.5) * Legendre↩
                ::positionMatrix[angularIncrements][incPhi];
            cosPhiPrecomputeMat[indBlockPhi][incPhi] = std::cos(incrementPositionFi);
        }
    }

    for (int indBlockT = 0; indBlockT < usedPrecision.thicknessBlockCount; ++indBlockT)
    {
        double blockPositionT = innerRadius + thicknessBlock * (indBlockT + 0.5);

        for (int incT = 0; incT <= thicknessIncrements; ++incT)
        {
            double incrementPositionT = blockPositionT +
                    (thicknessBlock * 0.5) * Legendre::positionMatrix[thicknessIncrements][↩
                        incT];

            double incrementWeightT = Legendre::weightsMatrix[thicknessIncrements][incT];

            double tempConstA = 2.0 * incrementPositionT * rPolar;
            double tempConstB = incrementPositionT * incrementPositionT + rPolar * rPolar;

            for (int indBlockPhi = 0; indBlockPhi < usedPrecision.angularBlockCount; ++↩
                indBlockPhi)
```

```
45              {
46                  for (int incPhi = 0; incPhi <= angularIncrements; ++incPhi)
47                  {
48                      double incrementWeightFi = Legendre::weightsMatrix[angularIncrements][↩
                            incPhi];
49                      double cosinePhi = cosPhiPrecomputeMat[indBlockPhi][incPhi];
50
51                      double tempConstC = 1.0/std::sqrt(tempConstB - tempConstA * cosinePhi);
52
53                      double tempConstD1 = topEdge * tempConstC;
54                      double tempConstD2 = bottomEdge * tempConstC;
55
56                      double tempConstE1 = std::sqrt(tempConstD1 * tempConstD1 + 1.0);
57                      double tempConstE2 = std::sqrt(tempConstD2 * tempConstD2 + 1.0);
58
59                      double tempConstF = std::log((tempConstE1 + tempConstD1) / (tempConstE2 ↩
                            + tempConstD2));
60
61                      magneticPotential += constant * incrementWeightT * incrementWeightFi * ↩
                            incrementPositionT * cosinePhi * tempConstF;
62                  }
63              }
64          }
65      }
66      return magneticPotential;
67  }
```

Konačno će se razmotriti kako je napisana sama funkcija `calculateAPotentialFast`. Počinje se definiranjem veličine domene integracije, odnosno veličine bloka, te fiksne konstante koja je ispred sume. Također su uvedena dva indeksa koja predstavljaju redak Gauss-Legendre matrica pozicija i težina kojemu se pristupa. Dalje se uočava da se „spora" funkcija $\cos(x)$ poziva više puta nad istim skupom vrijednosti pa ju je efikasno prethodno izračunati, prije ulaska u petlju. Za implementaciju je korišten dvostruki `vector` zato što može imati proizvoljnu veličinu i svaki redak predstavlja vrijednosti iz istog bloka, no ponajprije zato što MSVC prevoditelj pokazuje najbolje performanse u ovome slučaju. Uvijek se prvo računa pozicija središta samoga bloka u skladu s definicijom kvadrature (50) a zatim pojedina pozicija unutar tog bloka - zbog toga svaka dimenzija integracije sadrži dvije petlje. Učitava se i pripadna težinska vrijednost. Izračunavaju se neke privremene vrijednosti kako bi se smanjio broj operacija unutar sljedeće petlje. U ovome slučaju je to malen broj vrijednosti no u drugima je prilično velik.

Ulaskom u najunutarnjiju petlju (po `incPhi`) uočavamo da se nigdje ne poziva funkcija $\sinh(x)$. Umjesto toga koristimo definiciju same funkcije i pojednostavljujemo izraz u (100). Iako izraz ne izgleda jednostavnije, izbjegli smo jedan poziv „spore" funkcije $\ln(x)$ koja je implicitno ugrađena u $\sinh(x)$. Ovo je osjetno poboljšalo performanse. Funkcija korijena je na modernim procesorima koji podržavaju

AVX2 naredbe iznimno brza i može djelovati nad 4 vrijednosti istovremeno [19], pa se ne smatra sporom.

$$\sinh^{-1} x_1 - \sinh^{-1} x_2 = \ln\left(\frac{\sqrt{x_1^2+1}+x_1}{\sqrt{x_2^2+1}+x_2}\right). \tag{100}$$

Sve vrijednosti se sumiraju u jednu varijablu `magneticPotential`, a zbog svojstava Gauss-Legendre kvadrature ovo ne unosi pogrešku u konačni rezultat već on teoretski može ostvariti svih 15 značajnih znamenki preciznosti. Metode za izračun polja i gradijenta vrlo su slične no imaju veći broj privremenih vrijednosti i vraćaju više vrijednosti, u slučaju polja dvije, a u slučaju gradijenta 4 vrijednosti. Posebno su implementirani i izrazi za međuindukciju na zajedničkoj osi (80) te samoindukciju (81).

Prikazane metode za izračun polja čine osnovu za daljnju implementaciju metoda za izračun međuindukcije, sile i zakretnog momenta. U slučaju dvije pravokutne zavojnice, koji je fokus ovoga rada, metode za izračun polja pozivaju se nad velikim brojem unaprijed definiranih točaka koje su međusobno neovisne. Stoga je ovo dobra prilika za korištenje višedretvenosti kako bi se posao izračuna paralelizirao. Kako je jedno izvođenje metode izračuna polja iznimno brzo, reda mikrosekunde, potrebno je koristiti bazen dretvi (*threadpool*) kako bi se eliminiralo vrijeme potrebno za stvaranje dretve koje je približno istoga reda veličine. Za navedeno se koristi modul CTPL [24]. Kao što je već spomenuto u prethodnome odjeljku, radi poboljšanja performansi broj točaka podijeli se na broj blokova jednak broju dretvi te svaka dretva izračunava svoj blok. Ovo znatno poboljšava performanse i kod malenog broja točaka i kod ostvarenja maksimalnih teoretskih performansi (pri velikom broju točaka).

Uz metode za procesor, napisani su i CUDA kerneli za izvođenje na Nvidia grafičkim karticama u skladu s preporukama proizvođača [20]. Kernel je poseban tip funkcije prilagođen za SIMD. Konceptualno je kod unutar kernela identičan kao onaj za procesor, samo što se transformacija koordinata i izlaznog vektora izvodi unutar kernela a ne u posebnim funkcijama. Druga značajna razlika je unaprijed određen broj inkremenata zato što kerneli primaju statičke liste fiksne veličine. Kerneli također imaju najviše 255 32-bitnih registara na raspolaganju pa je potrebno smanjiti veličinu listi koje kernel prima kao argument (liste s vrijednostima Gauss-Legendre kvadarture). Korištenje tipa float smanjuje najveću preciznost na 6 značajnih znamenki pa je ovo dobar kompromis.

Ovo zaključuje razmatranja koja su započeta još u odjeljku 2.1. Prikazan je cijeli proces rješavanja problema zadanog u uvodu, od filozofskog pristupa, preko fizikalnih jednadžbi i izraza u obliku suma do programske implementacije koja u ovome pristupu igra vrlo važnu ulogu. U sljedećem poglavlju će biti utvrđeno koliko su navedene metode doista precizne i ono što je još važnije - koliko su brze u usporedbi s drugim metodama. Korišten je znatno veći broj slojeva integracije od sličnih pristupa, a navedeni nedostatak se prevladava korištenjem znatno jednostavnijih izraza, prevođenog jezika s omogućenim optimizacijama, višedretvenosti te algoritma za automatsko balansiranje broja inkremenata.

5 Usporedba izračunatih rezultata i performansi

5.1 Korištena računala

Kako bi se pružio uvid u performanse našega pristupa važno je detaljno specificirati koji sustavi su korišteni zato što očekujemo da će to imati znatan utjecaj na performanse pristupa. Razmatraju se performanse na više sustava različitih arhitektura i starosti. Također će se razmatrati utjecaj operacijskog sustava jer su očekivane razlike u performansama višedretvenosti i C++ prevoditelju. Ovakve usporedbe nisu pronađene u literaturi, vjerojatno zato što je očekivana razlika u performansama malena od sustava do sustava. Naime, svi dosadašnji pristupi su, koliko se može iz radova zaključiti, napisani za izvođenje na jednoj jezgri i napisani u programskom jeziku vrlo visoke razine poput MATLAB-a i Mathematice. Ako postoji korištenje višedretvenosti, onda je to zbog unutarnje implementacije funkcija u tim jezicima. Kako nije posebno navedeno, razumno je pretpostaviti da je izračun proveden na operacijskom sustavu Windows. Velik dio ovog poglavlja se koristi upravo za opis hardverskih specifikacija i uvođenje osnovnih pojmova vezanih uz ocjenu performansi hardvera, zato što je ovo svojstveno našem pristupu rješavanja problema - njegovo oslanjanje na efikasnu implementaciju s ciljem maksimalnog korištenja performansi koje nudi moderni hardver.

U testiranju je korišteno ukupno 5 računala:

- Računalo A: AMD Ryzen 9 5900HX 8C/16T 4.4GHz, Nvidia RTX 3080 Laptop 16GB 115W, 32GB DDR4-3200 RAM, Windows 10 19043 i Ubuntu 22.04
- Računalo B: AMD Threadripper 1950X 16C/32T 4.0GHz, Nvidia RTX 2080 Ti 11GB, 32GB DDR4-3200 RAM, Windows 10 19043
- Računalo C: Intel Core i7 7700HQ 3.4GHz 4C/8T, Nvidia GTX 1050 4GB, 16GB DDR4-2400 RAM, Windows 10 build 19043
- Računalo D: Intel Core i7 8700K 6C/12T 4.4GHz, Nvidia RTX 2080 Ti 11GB, 32GB DDR4-3000 RAM, Windows 11 22000.652 i Pop!OS 22.04
- Računalo E: Intel Core i7-8750H 6C/12T 3.6GHz, Nvidia RTX 2070 Mobile 8GB, 16GB DDR4-2667 RAM, Windows 11 22000.652 i Pop!OS 22.04

Ovaj uzorak računala pokriva širok spektar performansi te će se razmotriti osnovna obilježja svakog sustava. Kao referencu za performanse i opis ovih sustava koristi se stranica AnandTech [21]. Ova stranica se odlikuje ogromnim brojem hardverskih recenzija i vrlo dubokim analizama performansi procesora, poput vremena potrebnog za pristup različitim razinama priručne memorije (cachea) i vremena potrebnog za komunikaciju među različitim jezgrama. Ovo su vrlo važne informacije za performanse naših funkcija,

uz performanse samih jezgri. Navedene su sve specifikacije potrebne za nedvosmislenu klasifikaciju performansi korištenog sustava. Važno je naglasiti razliku između procesorske jezgre (C) i procesorske dretve (T). Jedna jezgra sadrži sve komponente potrebne za obradu podataka, a to uključuje više vrsta registara, više razina priručne memorije (cache) i više vrsta jedinica za obradu podataka (tipično ALU *Arithmetic Logic Unit* i FPU *Floating-Point Unit*). Nama je posebno važna FPU jer sadrži strukture za prethodno spomenute AVX2 instrukcije. U tom kontekstu, jedna jezgra je cijeli procesor u užem smislu riječi. Današnji procesori, u širem smislu riječi, imaju još dodatnu priručnu memoriju (L3 cache), putove za komunikaciju među jezgrama, komponente za komunikaciju s memorijom (MMU *Memory Management Unit* i memorijski kontroler za radnu memoriju) te komponentu za komunikaciju s drugim komponentama i čipsetom (PCIe kontroler). Svaka jezgra x86 procesora tipično sadrži dvije procesorske dretve koje sadrže sve registre potrebne za pokretanje jedne instance softverske dretve. Ova tehnologija se naziva SMT (*Simultaneous Multithreading*). Dakle, određeni registri u procesoru su dvostruki. Moderni procesori imaju vrlo složene strategije za poboljšanje performansi i kombiniranje izvođenja dviju softverskih dretvi. Ako jedna dretva potpuno opterećuje jezgru, neće biti poboljšanja ukupnih performansi te će se dretve izvoditi naizmjence. Ako jedna dretva slabo iskorištava jezgru i jezgra uspješno kombinira izvođenje dvaju dretvi, moguće je dobiti znatno veće performanse. Dodatni problem može uzrokovati operacijski sustav ako ne razlikuje koja dretva pripada kojoj jezgri, redovito je slučaj da on prepoznaje samo procesorske dretve. U slučaju da je broj dretvi jednak broju procesorskih jezgri moguće je da neke jezgre ostanu neiskorištene ako planer operacijskog sustava nekoj jezgri zada dvije dretve. Za grubu procjenu očekivanih performansi moguće je koristiti neki od popularnih alata za mjerenje performansi, poput Cinebench R23.

Računalo A predstavlja moćan laptop prethodne generacije. Intelovi Core procesori 12. generacjie znatno su brži u većini primjena od prošlogodišnjeg AMD Ryzen 5900HX procesora u ovome računalu. No ovaj procesor, s 8 jezgara, 16 dretvi i 20MB cachea, je i dalje vrlo moćan te predstavlja performansu dostupnu na većini računala iz zadnje 2 godine. Upravo zbog toga će većina testova biti napravljena na njemu. Sljedeće performanse su karakteristične za Ryzen 5000 Cezanne seriju procesora: vrijeme komunikacije između jezgri je uniformno i iznosi približno 20 nanosekundi (oko 80 ciklusa), pristup 32 kB L1 cachea nema mjerljivo kašnjenje, a pristup 512 kB L2 cachea traje približno 6 ciklusa. Ovo je vrlo važno zato što svi podatci potrebni za izračun jedne točke stanu direktno u L1 cache, a sve statičke vrijednosti korištene u cijelom kodu stanu u L2 cache. Isto bi trebalo vrijediti za sve ostale testirane procesore. Ovako fini detalji su izvan naše kontrole, pa prevoditelj treba prepoznati ove značajke i optimizirati konačni strojni kod. Zadnje važno svojstvo procesora su dvije AVX2 jedinice po jezgri što znatno povećava najveće ostvarive performanse.

Računalo A također sadrži Nvidia RTX 3080 Laptop grafičku karticu koja, iako je prilično brza,

ipak nije blizu kartica koje se mogu pronaći u stolnom računalu, pogotovo za proračune. Njen glavni nedostatak je PCIe 3.0 x8 brzina prijenosa, odnosno najviše 8 GB/s. Ovo je malo u usporedbi s teoretskih 32 GB/s dostupnih na PCIe 4.0 x16 i može znatno smanjiti performanse pri korištenju grafičke kartice za proračune. Uzmimo naš konkretan slučaj: grafičkoj kartici je potrebno poslati tri float vrijednosti za svaki izračun vrijednosti polja, odnosno $12N$ bajtova za N točaka. Natrag je potrebno poslati tri vrijednosti u slučaju potencijala i polja ($12N$), odnosno devet vrijednosti ($36N$) u slučaju gradijenta, sve tipa float. Ovime možemo izračunati maksimalne teoretske performanse - performanse u slučaju da se vrijednost polja istoga trenutka izračunava u kartici i vrijeme prijenosa na karticu ne ovisi o broju točaka. 8 GB/s podijeljeno s 24 B daje 333 milijuna prijenosa po sekundi (odnosno točaka po sekundi) za potencijal i magnetsko polje, a 8 GB/s podijeljeno s 48B daje 167 milijuna točaka po sekundi za gradijent. Stvarni brojevi će, naravno, biti manji. U našem testiranju pri prijenosu 134 milijuna vrijednosti postiže se brzina prijenosa od približno 5 GB/s. Ovo bi trebalo predstavljati znatno manji problem na stolnom računalu gdje bi radna memorija sustava s brzinama prijenosa od 20-ak GB/s trebala postati usko grlo. Uz brzinu radne memorije, performanse ovise i o platformi (matičnoj ploči i čipsetu).

Računalo B predstavlja 5 godina starije stolno računalo vrlo visokih performansi (često se ova kategorija naziva HEDT). AMD Threadripper 1950X je procesor sa 16 jezgri podijeljenih na dva čipa po 8 jezgri. Nisu pronađene precizne vrijednosti za vremena pristupa između jezgri, no na novijim verzijama iste linije procesora iznose 20 nanosekundi za pojedine skupine od 4 jezgre i 80 nanosekundi prema jezgrama van skupine. Kako su prave vrijednosti gotovo sigurno veće, ne očekuje se dobro skaliranje performansi kod višedretvenosti, osim u slučaju kada je moguće koristiti `MTD`. Jedan od velikih nedostataka je činjenica da se AVX2 instrukcije izvršavaju u 2 ciklusa umjesto u 1, a navedeno je karakteristično za prvu generaciju AMD Zen arhitekture. Grafička kartica RTX 2080 Ti je nešto moćnija od RTX 3080 Mobile uz prednost PCIe 3.0 x16 brzine prijenosa (16 GB/s).

Računalo C je nešto stariji laptop s četverojezgrenim procesorom zasnovanim na Intel Skylake arhitekturi iz 2015. godine. Sama Skylake arhitektura po performansama nije jako različita od prve komercijalne arhitekture koja podržava AVX2 instrukcije - Core procesori 4. generacije. Dakle performanse na računalu C gotovo su identične performansama računala iz 2013. godine. Komunikacija između jezgri ovih starijih arhitektura je nekonzistentna zato što su jezgre međusobno povezane tehnologijom komunikacijskog prstena (*Ring Interconnect*). Ovo je sporije od većine modernih rješenja i vremena pristupa od jezgre do jezgre su redovito znatno viša, 30-60 nanosekundi. No i sam procesor je dosta sporiji od onog u računalu A, pa je moguće da višedretvenost i dalje donosi znatno unaprjeđenje. Grafička kartica GTX 1050 je jedna od najslabijih kartica pretprošle generacije (6 godina stara) sa 640 CUDA jezgri i frekvencijom rada od približno 1.4 GHz.

Računalo D je računalo visokih performansi staro nekoliko godina. Njegov Intel Core procesor

8. generacije je isto zasnovan na Skylake arhitekturi, ali sa 6 jezgara i većim brzinama. Očekuje se ponašanje slično računalu C uz ubrzanje od približno 30% slučajevima gdje se koristi jedna dretva, a dvostruko ubrzanje u idealnom višedretvenom slučaju. Grafička kartica je ista kao u računalu B. Ovo omogućava usporedbu utjecaja platforme na performanse.

Računalo E je laptop visokih performansi star nekoliko godina s procesorom koji je vrlo sličan procesoru u Računalu C, ali s 2 jezgre više, pa je u tom kontekstu sličniji računalu D. Radi se i dalje o istoj arhitekturi i očekuju se performanse između računala C i D. Grafička kartica je RTX 2070 Mobile i po svojim specifikacijama nije znatno sporija od RTX 3080 Laptop.

Za daljnju usporedbu performansi samih grafičkih kartica i procesora korištene su upute dane u [22]. Ako bi se RTX 3080 Laptop i RTX 2070 Mobile uspoređivale isključivo po broju CUDA jezgri, činilo bi se da je RTX 3080 Laptop, sa 6144 jezgre, barem 2 puta moćnija od RTX 2070 Mobile s 2560 jezgara. Pomnije razmatranje arhitekture kartice RTX 3080 Mobile otkriva da ona ima 48 SM-ova (*Streaming Multiprocessor*), a svaki SM ima 64 fizičke jedinice koje mogu paralelno izvoditi dvije instrukcije nad float tipom podataka. RTX 2070 Mobile ima 40 SM-ova, a svaki ima isto 64 fizičke jedinice za računanje koje mogu izvoditi jednu instrukciju nad float tipom podataka. Tradicionalno su te jedinice u SM-u mogle u danom trenutku izvoditi samo jednu instrukciju i upravo zato su CUDA jezgre bile vrlo korisna metrika performansi. Od generacije RTX 3000, CUDA jezgra nema više ovo značenje jer sada jedna jezgra može u određenim okolnostima izvoditi dvije instrukcije. Kako redovito nije moguće paralelno izvršavati dvije instrukcije na jednoj jezgri, grafička kartica RTX 3080 Laptop redovito je samo 30-50 % brža od RTX 2070 Mobile.

Grafička kartica sadrži vrlo velik broj jedinica za paralelno izvođenje pa očekujemo da će trebati vrlo velik broj točaka kako bi kartica bila potpuno opterećena tijekom izvođenja - barem nekoliko puta više točaka od broja CUDA jezgri. Iako su CUDA jezgre znatno slabije od procesorskih, jedna grafička kartica ih ima nekoliko tisuća u usporedbi s desetak u novijim procesorima. Ako računamo s podatcima tipa double, jedna procesorska jezgra korištenjem AVX2 instrukcija može izvesti najviše 4 paralelne instrukcije (SIMD) po jednoj AVX2 jedinici. U slučaju računala A koje ima dvije AVX2 jedinice, to je ukupno 8 instrukcija. Neke od tih operacija su vrlo moćne pa mogu izvršiti dvije aritmetičke operacije odjednom [19]. Primjer takve operacije je MAD (*multiply-add*), nekad se naziva i MAC (*Multiply-Accumulate*), koja predstavlja istovremeno množenje dva realna broja te pribrajanje odredišnoj varijabli. Na modernim procesorima se tipično uzima da jedna jezgra može izvršiti najviše 16 aritmetičkih operacija nad podatcima tipa double zato što sadrži dvije AVX2 jedinice ili jednu AVX512 jedinicu. AVX2 koristi YMM registre koji su široki 256 bitova pa mogu primiti ili 4 double podatka ili 8 float podataka. Kako moderni procesori tipično imaju 8 jezgara koje rade na frekvenciji od približno 4 GHz, moguće je izvršiti čak 512 GFLOPS (*Floating Point Operations Per Second*). U stvarnosti je ovaj

broj niži te je procjena performansi na 128-256 GFLOPS znatno realističnija kod praktičnih razmatranja zbog dohvaćanja potrebnih podataka. Svaka CUDA jezgra može izvršiti dvije aritmetičke operacije po ciklusu (jednu instrukciju MAD koja se broji kao dvije aritmetičke) i tipična grafička kartica radi na frekvenciji od 1.8GHz. Ako pretpostavimo da ima 3072 takve jezgre, kao što je slučaj kod RTX 3080 Laptop, dobivamo 11.1 TFLOPS. Dakle, grafička kartica teoretski posjeduje 22 puta veće performanse od procesora, a u praksi očekujemo 40-80 puta veće performanse. Nedostatak je već navedena potreba za prijenosom podataka iz radne memorije procesora na radnu memoriju grafičke kartice. Ove procjene su implicitno napravljene za performanse na računalu A uz pesimističnu pretpostavku da se dvije MAD instrukcije neće paralelno izvršavati na jednoj CUDA jezgri (što reklamni materijali pretpostavljaju).

Konačno valja naglasiti da i ostale komponente mogu imati utjecaj na brzinu izračuna - brzina radne memorije, svojstva dane generacije procesora, svojstva čipseta, operacijski sustav, itd. Jedina razlika koju ćemo detaljno razmotriti je operacijski sustav zato što se u kontekstu radnih stanica često koriste Linux i Windows operacijski sustavi. Brzina radne memorije, veza procesora prema ostatku sustava i specifičnosti arhitekture grafičke kartice mogu znatno utjecati na brzinu komunikacije između procesora i grafičke kartice. Također je poznato ograničenje da procesor na komercijalnim platformama može odjednom pristupati samo 256 MB radne memorije na grafičkoj kartici. Ovo ograničenje na modernim sustavima nestaje uvođenjem tehnologije *Resizable BAR*.

Ovo razmatranje je bilo vrlo detaljno kako bi se pružio bolji uvid u performanse testiranih sustava i napravila predviđanja performansi. Na temelju njih i samih testova performansi dostupnih u sljedećem poglavlju 5.2 moguće jer formulirati očekivanje za trenutno dostupne i buduće procesore. Performanse grafičkih kartica će se razmotriti na osnovnoj razini, u svrhu usporedbe s modernim procesorima.

Osvrnimo se konačno na individualne performanse procesora. Najčešće su korištene funkcije $\sqrt{x}$, $\cos(x)$ i $\ln(x)$ pa njihove performanse pobliže promatramo. Prikazane su 3 verzije izračuna logaritma kako bismo se pobrinuli da koristimo najbržu implementaciju - jednim množenjem moguće je korigirati korištenje drugačije baze. Kvantificiranje performansi funkcije nije jednostavan postupak, a mi smo mu pristupili tako da stvorimo varijablu i inicijaliziramo je na 0, a potom uđemo u petlju koja se vrti 200 milijuna puta i pribraja navedenoj varijabli `func(i)`, gdje je `i` iterator petlje. U slučaju sinusa i kosinusa pribraja se `cos(1.0/i)`, odnosno `sin(1.0/i)`, kako bi argument funkcije bio unutar intervala $[0, 2\pi]$. Iako ovo usporava izvođenje, izlazak iz intervala smanjuje performanse znatno više i nije reprezentativan za naš slučaj. Dodana je i osnovna funkcija pribrajanja iteratora označena sa sum, kao primjer iznimno brze operacije. Iako ovo nije savršen test, trebao bi nam pružiti osnovni uvid u brzinu korištenih funkcija i utjecaj AVX2 instrukcija. Kao referentnu vrijednost za performanse uzimamo Računalo A bez omogućenih AVX2 instrukcija (samo optimizacijska zastavica je postavljena), dok su u ostalim slučajevima one omogućene. Performanse su izražene u broju milijuna iteracija opisane petlje

po sekundi. Pop!OS 22.04 je zasnovan na Ubuntu 22.04 (uz nešto noviji Linux kernel, 11.7 na Pop, a 11.5 na Ubuntu) pa očekujemo minimalne razlike u performansama. U daljnjem razmatranju ova dva operacijska sustava nećemo razlikovati zato što su na oba instalirani Ubuntu driveri za grafičku karticu, a malena razlika u verziji kernela je zanemariva. U tablicama će se skraćeno pisati Pop i Ubu dok će se u tekstu samo govoriti Ubuntu. Slično Windows 10 i 11 ne razlikujemo u većini slučajeva zato što ne testiramo sa sustavom najnovije generacije čije performanse znatno ovise o planeru operacijskog sustava. Ipak moguće je da se pojave određene razlike u performansama kod višedretvenog opterećenja.

Tablica 1: Performanse osnovnih funkcija izražene u broju milijuna iteracija petlje po sekundi

func	ref Win	ref Ubu	A Win	A Ubu	B Win	C Win	D Win	D Pop	E Win	E Pop
sum	1502	1497	1508	11639	1199	738	1023	7853	910	7601
sqrt	507	508	2026	2035	997	1070	1396	1485	1256	1314
log	272	300	797	797	353	487	628	735	574	624
log10	264	171	798	739	325	419	392	444	365	383
log2	163	215	163	750	352	420	103	557	98.3	426
sin	429	163	930	906	329	576	681	752	585	621
cos	435	155	803	896	316	502	555	657	489	567

Uočava se da u referentnom slučaju postoje velike razlike u performansama između Windows i Ubuntu operacijskih sustava. Razumno je pretpostaviti da MSVC koristi određene dodatne mogućnosti procesora (poput SSE registara) koje GCC (Ubuntu prevoditelj) ne koristi osim ako mu eksplicitno nije zadano preko zastavice prevoditelju. Kada je AVX2 omogućen, performanse su vrlo slične uz neke zanimljive iznimke - log2 je iznimno spor na Windowsu. Za naše potrebe je optimalan običan log (prirodni logaritam) jer ima najbolje performanse i nalazi se u izrazu za potencijal. Dakle, performanse sporijih funkcija koje koristimo su i dalje vrlo visoke - izvršavaju se u približno 5-8 procesorskih ciklusa. Ovo dobivamo dijeljenjem frekvencije procesora koja iznosi približno 4400 MHz s brojem milijuna iteracija petlje po sekundi. Zanimljivo je uočiti da GCC pokazuje znatno veće performanse u slučaju čistog sumiranja na računalu A - inkrementiranje petlje i pribrajanje se izvrši skoro 3 puta u jednome ciklusu. Ekvivalentno sumiranje u MatLabu je znatno sporije; prema našem testiranju ostvaruje se samo 16 milijuna koraka po sekundi. Korjenovanje je također vrlo brzo na oba operacijska sustava - potrebna su približno 2 ciklusa. Korjenovati se mogu do 4 broja paralelno, a na nekoliko mjesta se navedeno i koristi.

Imajući sve navedeno na umu mogu se razmotriti performanse metoda za izračun polja.

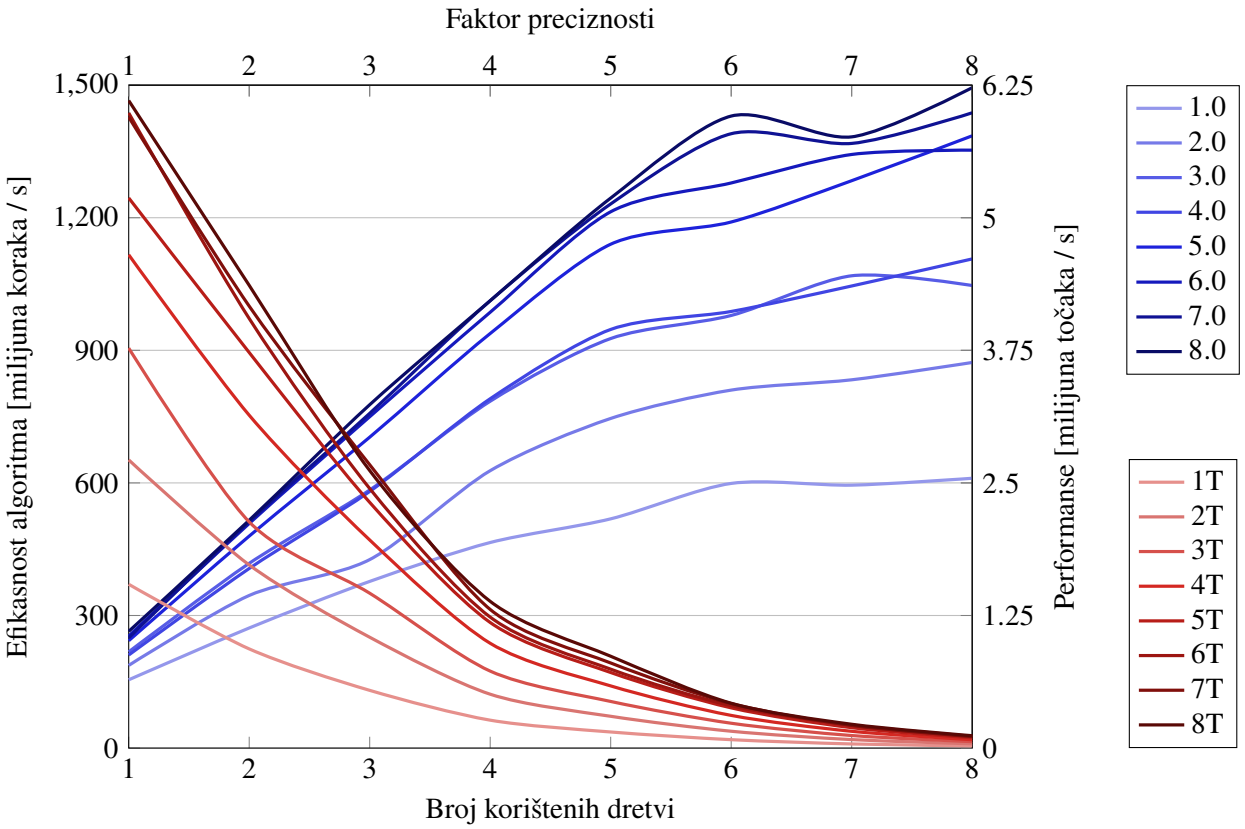
5.2 Performanse metoda za izračun polja

Kao što je već više puta rečeno, metode za izračun polja su središnji dio ovoga rada i njihova efikasna implementacija neophodna je za sve daljnje izračune. Preciznost ovih metoda ne analizira se direktno usporedbom preciznosti izračuna za različite točke u prostoru već na temelju izračuna međuindukcije, sile i zakretnog momenta. Zbog znatnog korištenja numeričke integracije (dva od tri sloja), očekuju se veće pogreške kada je odabrana točka vrlo blizu zavojnice. Ponašanje polja unutar zavojnice ne očekujemo da će nužno biti stabilno zato što svi izrazi za polja u nazivniku sadrže izraze koji divergiraju u određenim točkama. Kako točke divergencije uglavnom nisu na rubovima domene i strmog su karaktera, ne bi trebale predstavljati problem za odabranu metodu numeričke integracije osim u specifičnim slučajevima poput dviju vrlo blizu postavljenih petlji ili plosnatih zavojnica. Izračun magnetskog polja i gradijenta unutar zavojnice nisu nužni za izračun sile i zakretnog momenta, ali omogućavaju uvid u polje unutar vodiča pa će biti prikazana jedna takva slika na kraju poglavlja.

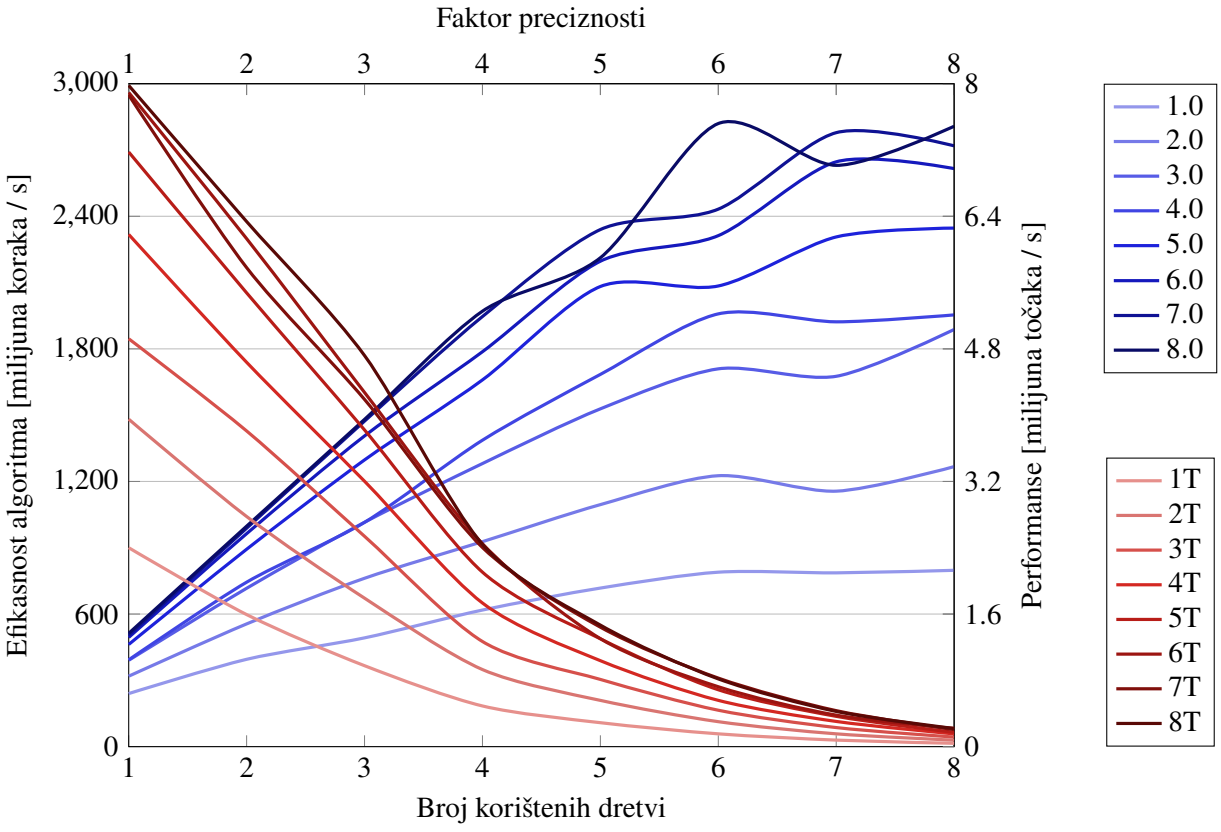
Performanse metoda za izračun polja primarno se razmatraju na 2 operacijska sustava: Windows 10 i Ubuntu 22.04, instalirana na računalu A. Važne su dvije osnovne mjere performansi: realne performanse, izražene kao broj točaka po sekundi, te efikasnost implementacije algoritma, broj prolazaka unutarnje petlje po sekundi. Realne performanse su one koje se mogu očekivati u slučaju izračuna velikog broja točaka - recimo kod izračuna vrijednosti za prikaz polja ili općenitog proračuna međuindukcije, sile i zakretnog momenta. Skaliranje performansi s brojem točaka razmatra se nešto kasnije, a ono će pružiti precizan odgovor na pitanje koliko je točaka potrebno za efikasno opterećenje višejezgrenog procesora. Efikasnosti algoritma je dana mjerna jedinica broj milijuna koraka po sekundi, a unutar funkcija za testiranje performansi implementiranih u kodu koristi se MInc/s (*MegaIncrements per second*). Radi se o istim mjernim jedinicama. Očekujemo da će performanse i efikasnost algoritma ovisiti o broju dretvi i korištenom faktoru preciznosti pa za potpuni prikaz trebamo dvije familije krivulja: jednu koja prikazuje koliko je performansi dostupno korisniku te jednu koja prikazuje koliko je algoritam efikasan.

U slučaju efikasnosti algoritma pogodno je na x-os postaviti broj korištenih dretvi jer tada dobivamo informaciju koliko se efikasno koristi višedretvenost. Različite krivulje prikazuju efikasnost za različite razine preciznosti. Očekuje se porast efikasnosti kako unutarnja petlja potiskuje konstantni i linearni član u algoritamskoj složenosti izračuna. Ovo se redovito naziva skaliranje višedretvenosti. U idealnome slučaju bi udvostručenje broja dretvi povećalo efikasnost algoritma dva puta, kao i broj točaka po sekundi. U stvarnosti očekujemo manje poboljšanje performansi zbog postojanja dvije procesorske dretve u jednoj jezgri - tek bi kod korištenja 16 dretvi trebali potpuno opteretiti osmojezgreni procesor sa SMT. Valja također imati na umu da moderni procesori imaju napredne strategije prilagodbe frekvencije rada ovisno o tome koliko jezgri se koristi. Za prikaz performansi pogodnije je na x-os postaviti željenu preciznost, a određena krivulja prikazuje performanse za određeni broj dretvi.

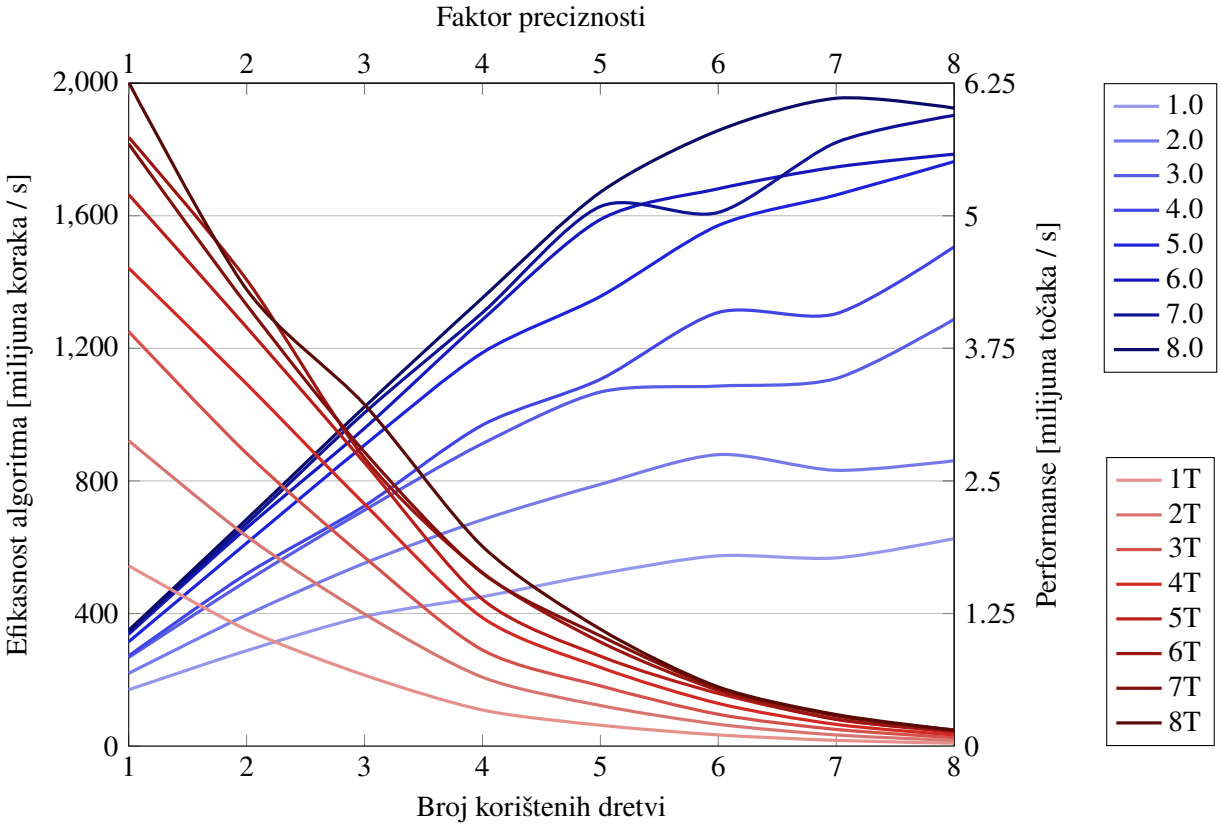
Slika 4: Prikaz performansi brze metode za izračun vektorskog potencijala (računalo A Win10)



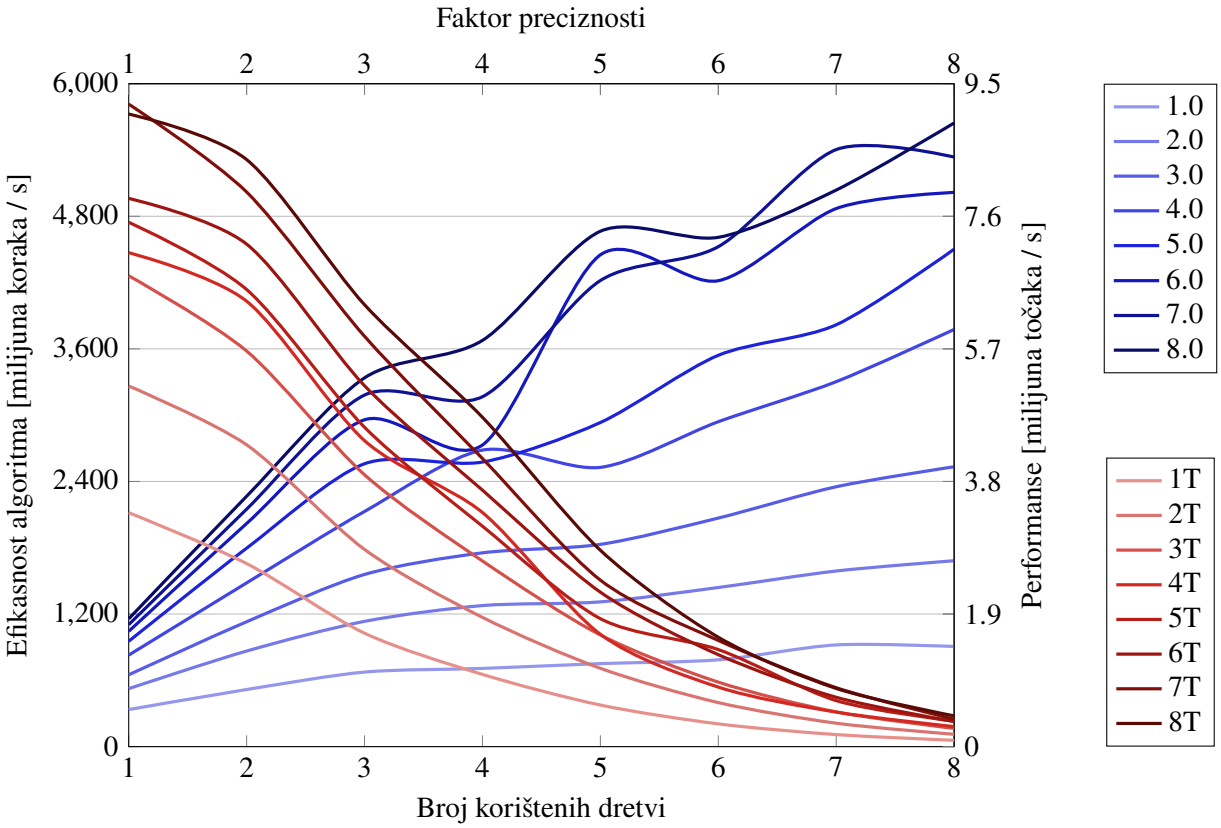
Slika 5: Prikaz performansi brze metode za izračun magnetskog polja (računalo A Win10)



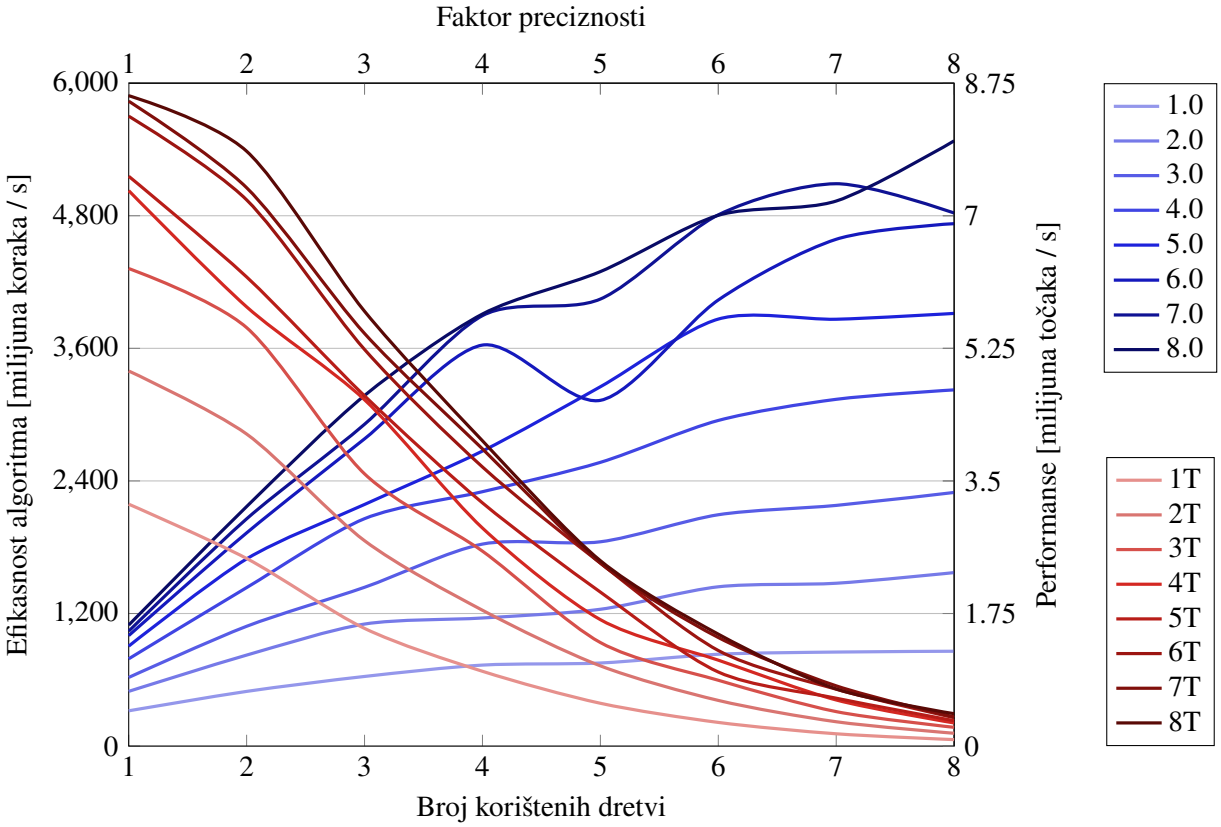
Slika 6: Prikaz performansi brze metode za izračun gradijenta magnetskog polja (računalo A Win10)



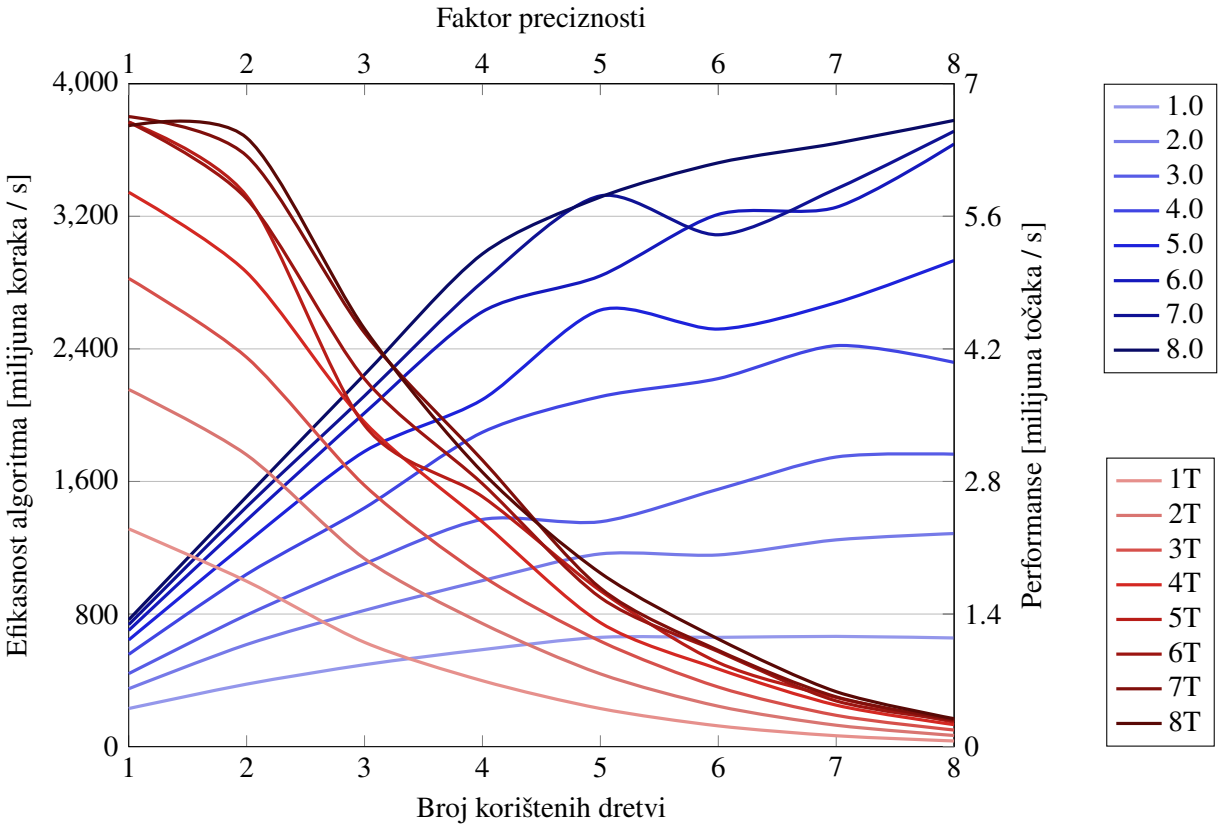
Slika 7: Prikaz performansi spore metode za izračun vektorskog potencijala (računalo A Win10)



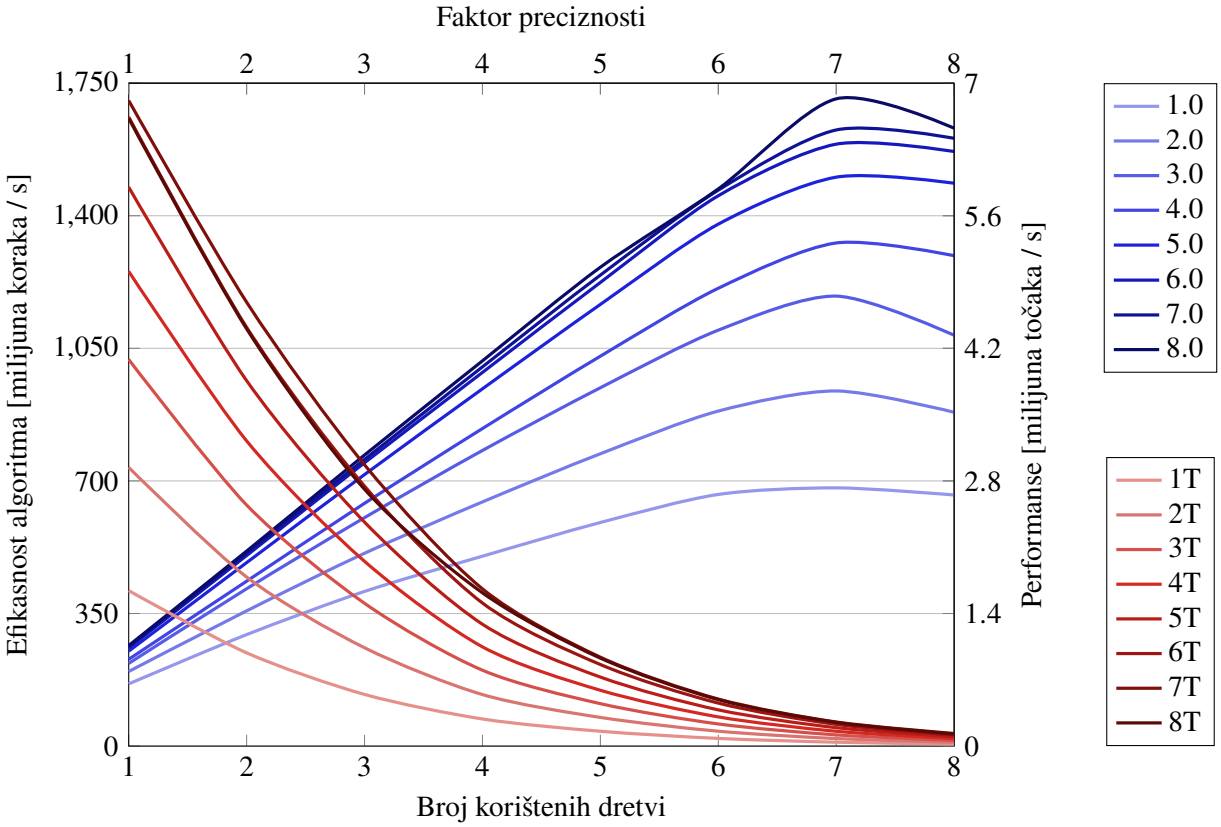
Slika 8: Prikaz performansi spore metode za izračun magnetskog polja (računalo A Win10)



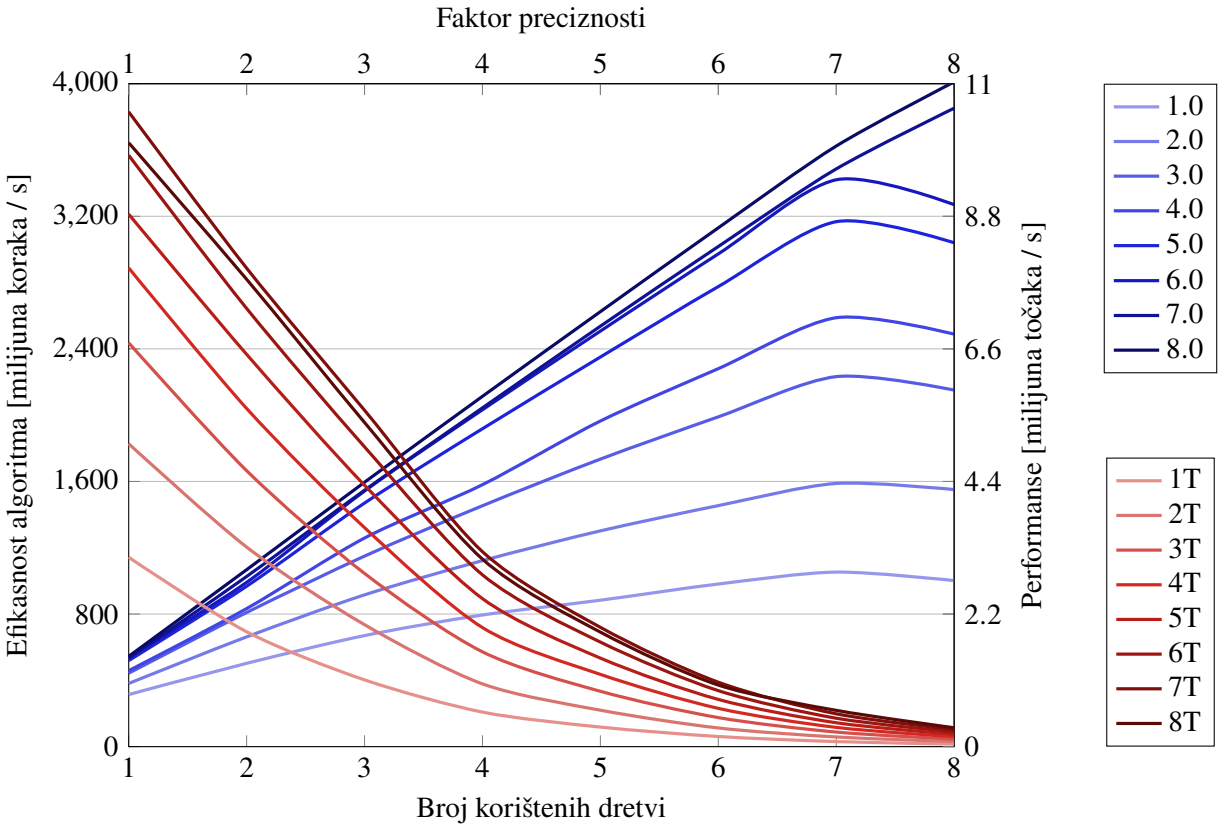
Slika 9: Prikaz performansi spore metode za izračun gradijenta magnetskog polja (računalo A Win10)



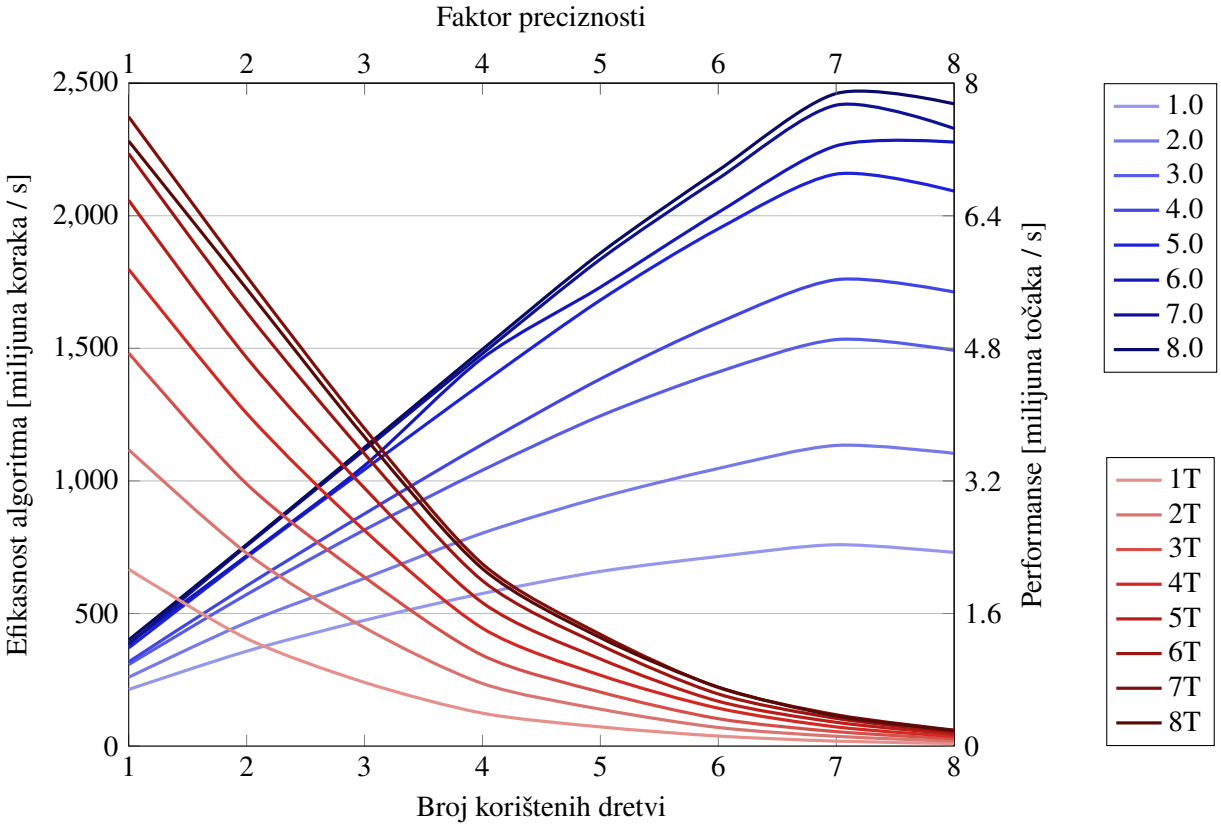
Slika 10: Prikaz performansi brze metode za izračun vektorskog potencijala (računalo A Ubu22.04)



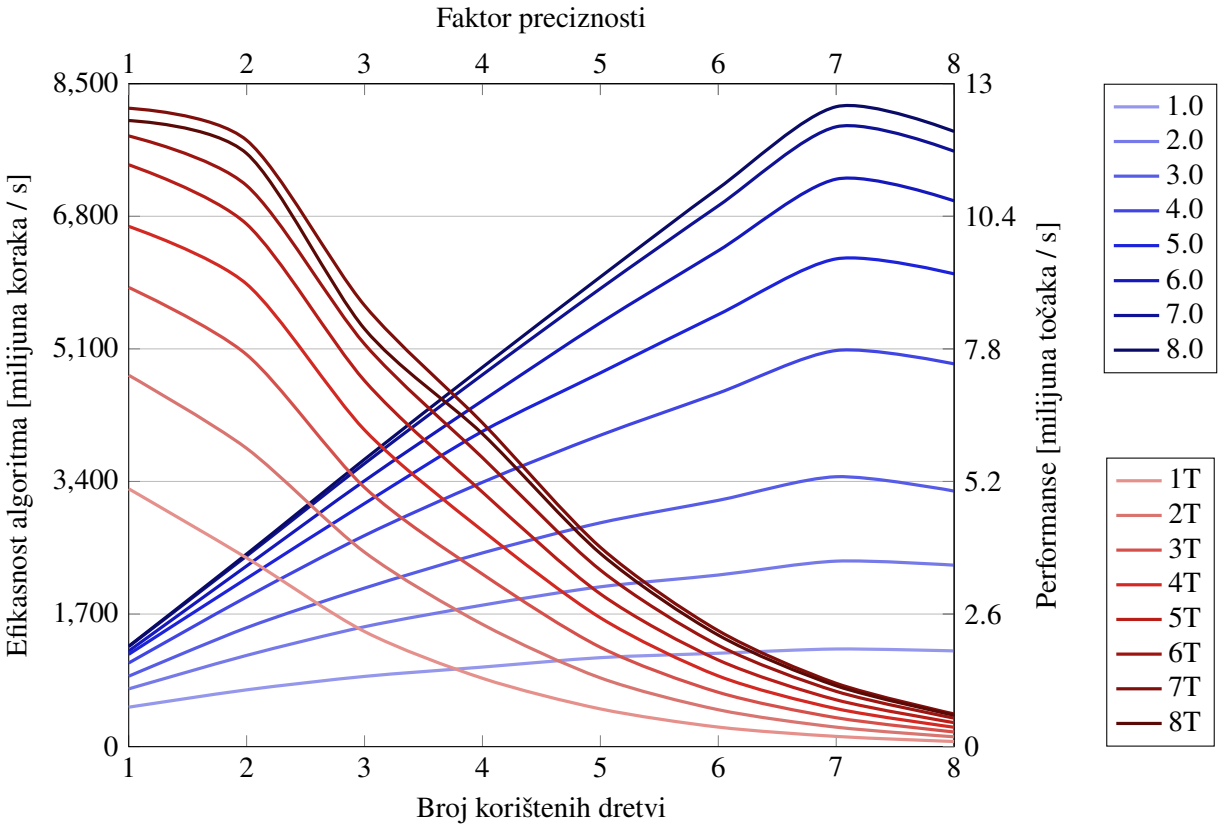
Slika 11: Prikaz performansi brze metode za izračun magnetskog polja (računalo A Ubu22.04)



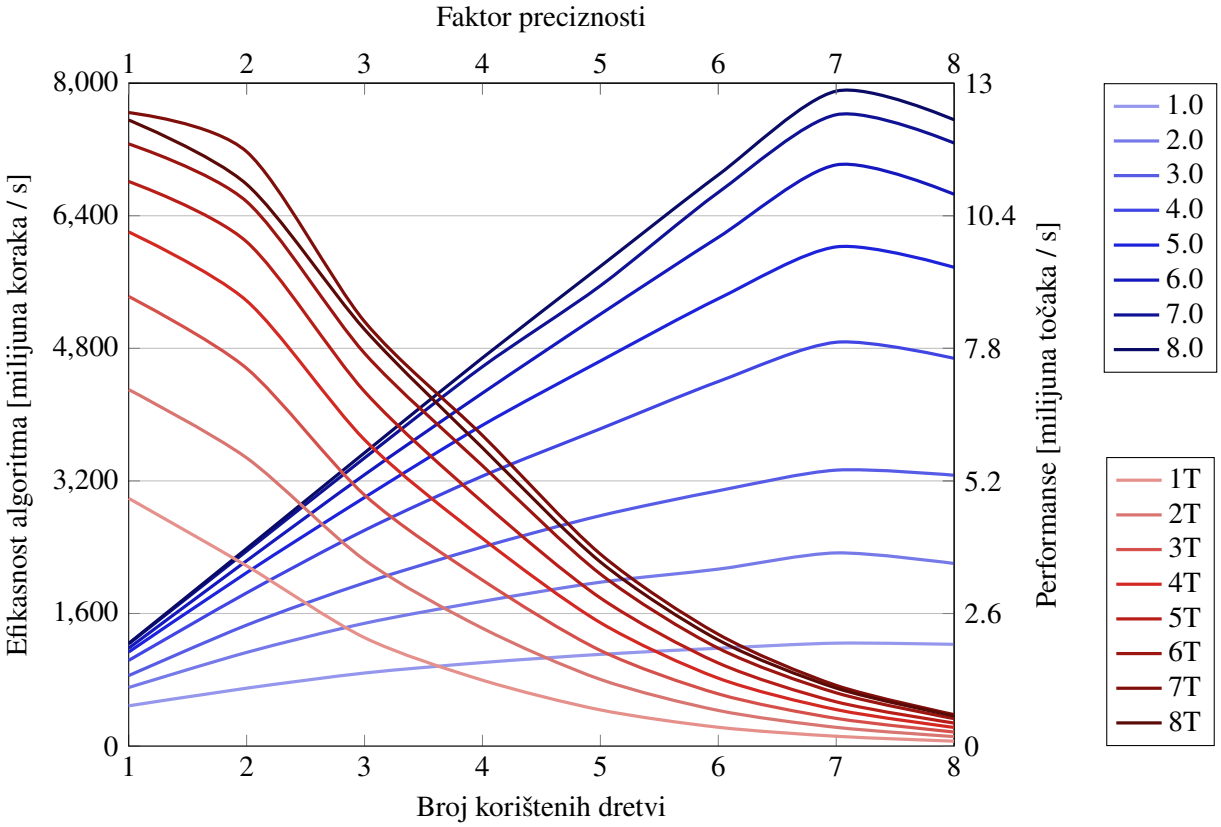
Slika 12: Prikaz performansi brze metode za izračun gradijenta magnetskog polja (računalo A Ubu22.04)



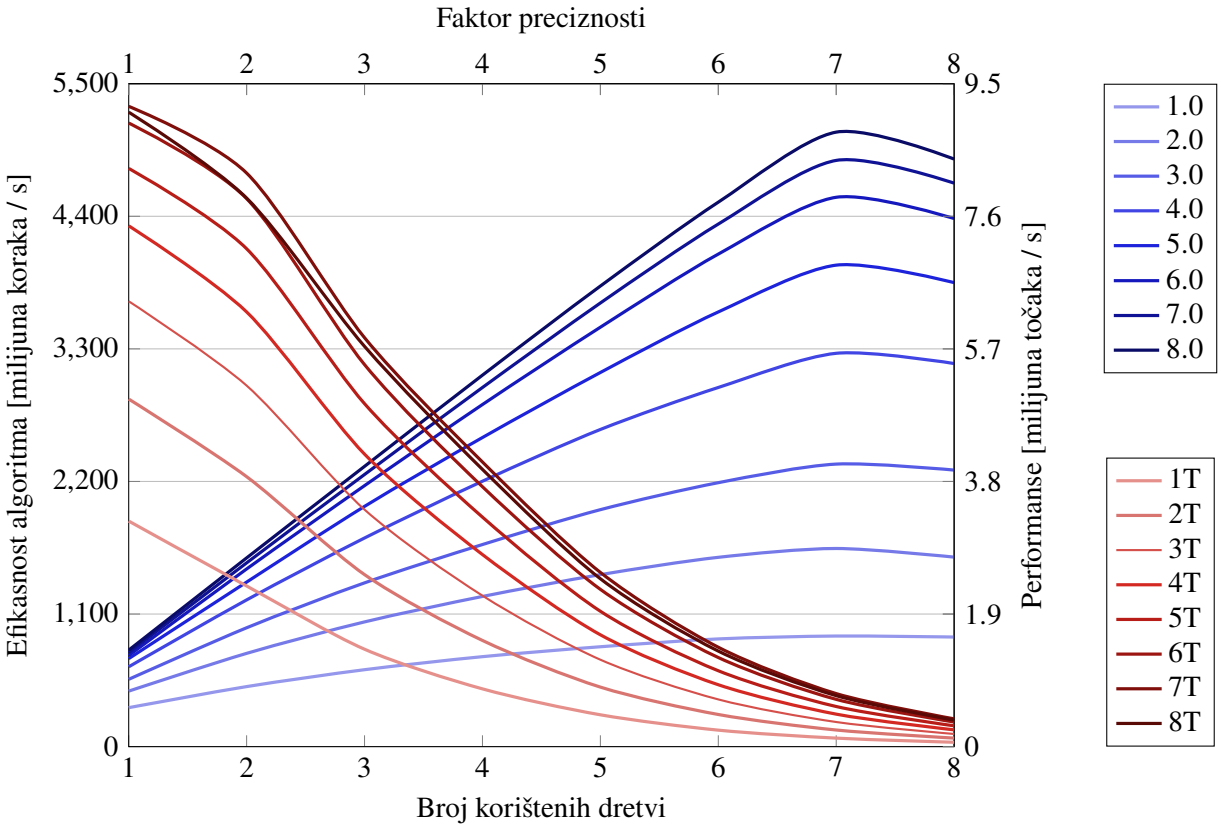
Slika 13: Prikaz performansi spore metode za izračun vektorskog potencijala (računalo A Ubu22.04)



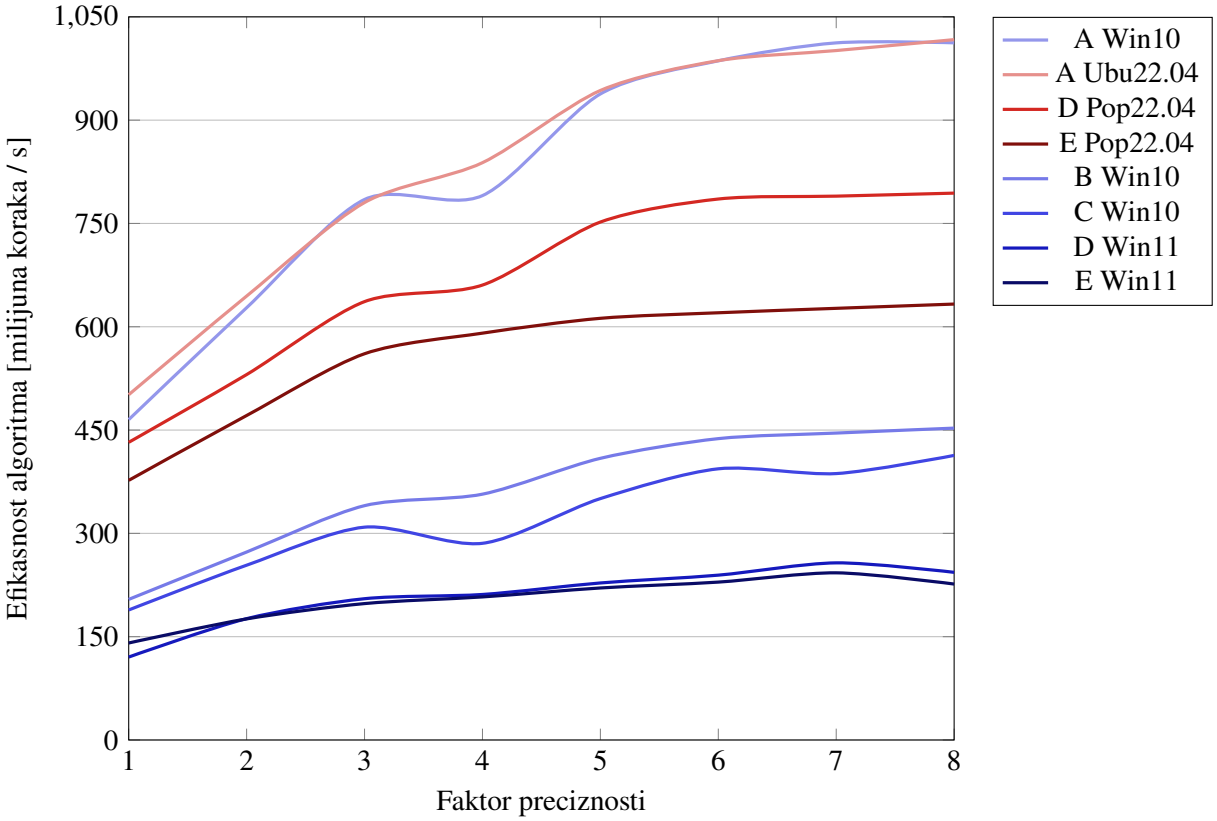
Slika 14: Prikaz performansi spore metode za izračun magnetskog polja (računalo A Ubu22.04)



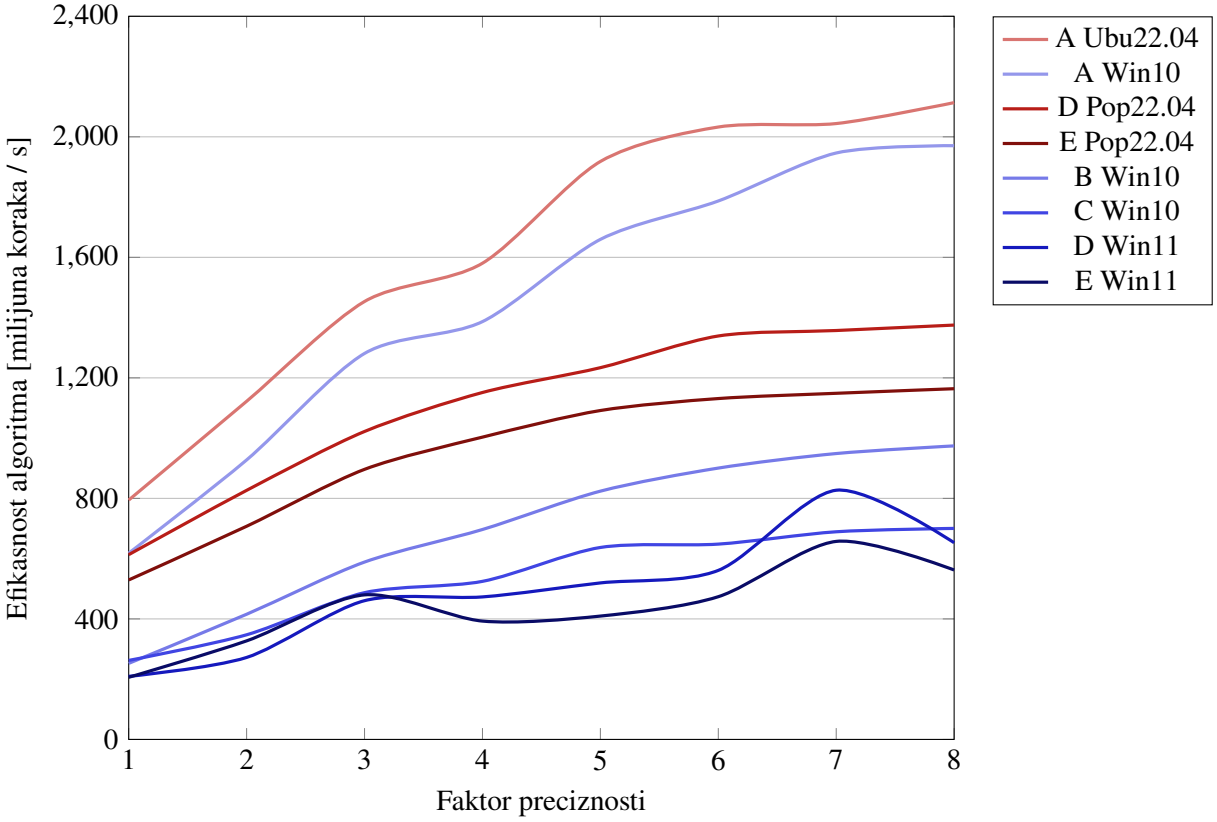
Slika 15: Prikaz performansi spore metode za izračun gradijenta mag. polja (računalo A Ubu22.04)



Slika 16: Usporedba performansi brze metode za izračun vektorskog potencijala (4T)



Slika 17: Usporedba performansi brze metode za izračun magnetskog polja (4T)



Prikazane su performanse svih brzih i sporih metoda za izračune polja na dvama operacijskim sustavima (prvih 12 grafova) te usporedba performansi između različitih računala korištenih na dvama operacijskim sustavima (zadnja 2 grafa). Performanse se testiraju tako da se stvori novi objekt zavojnice, postavi željeni broj dretvi, generira lista vrijednosti za koje se računa polje, pokrene mjerenje vremena te poziva prikladna metoda za izračun polja nad definiranom zavojnicom (za potencijal konkretno `computeAllAPotentialComponents`). Nakon završetka izvođenja metode, vrijeme se zaustavlja te se ispisuju performanse i efikasnost algoritma (uz upis u .txt datoteku). Lista vrijednosti sadrži 200 003 točke za naše testiranje. Ovo je srednji broj točaka; nešto više nego što bi koristile metode za izračun međuindukcije, sile i zakretnog momenta, a manje nego što bi se koristilo za generiranje prikaza polja.

Prva očita razlika između prvih 6 i drugih 6 grafova je glatkoća krivulja - na operacijskom sustavu Ubuntu ponašanje metoda je konzistentnije, a najveće performanse su znatno veće nego na operacijskom sustavu Windows. Efiksnost metoda je također poboljšana za otprilike 20 %. Razlike se smanjuju pri većim faktorima preciznosti, dakle pri većem opterećenju. Navedeno sugerira da Ubuntu bolje koristi resurse koje ima na raspolaganju uz zanimljiv fenomen smanjenja performansi pri korištenju 8 dretvi. Uzrok ovoga je vrlo vjerojatno automatsko usporavanje procesora zbog velikog opterećenja - navedene metode dovoljno opterećuju procesor da doseže temperature bliske maksimalnoj. Kako na Ubuntu nije uspješno instaliran softver za regulaciju brzine ventilatora, dolazi do pregrijavanja pri najvećem opterećenju. Ovo slijedi iz činjenice da na Windowsu procesor doseže 93 stupnja Celzijeva (maksimum je 100) uz ventilatore na maksimalnoj brzini. Razumno je zaključiti da program u dobroj mjeri koristi AVX2 instrukcije jer one redovito dovode do velikog zagrijavanja. Na Windowsu je ponašanje vrlo nekonzistentno i krivulje svaki put izgledaju drugačije.

Nadalje, uočavamo da su performanse i brzih i sporih metoda odlične - moguće je izračunati milijune točaka po sekundi s nižom i srednjom preciznosti i stotine tisuća s visokom preciznosti. Vidimo da je od brzih metoda vektorski potencijal daleko najsporiji i to upravo zato što koristi logaritam. Magnetsko polje je najbrže, a gradijent je iznenađujuće brz s obzirom na broj aritmetičkih operacija potreban za njegov izračun. Ovo pokazuje punu moć AVX instrukcija. Uočava se da povećanjem faktora preciznosti efikasnost algoritma raste zato što je sve veći udio vremena proveden u unutarnjoj petlji algoritma sličnog onomu prikazanom u odjeljku 4.2. Ovo ublažava smanjenje performansi pri većim faktorima preciznosti. Vidimo da je kod brzih metoda u slučaju jedne jezgre i visokog faktora preciznosti broj milijuna koraka po sekundi između 250 i 500. Kod sporih metoda performanse su još više, što je očekivano s obzirom na to da su i sami izrazi jednostavniji. Efikasnost sporih metoda iznimno je visoka, između 750 i 1250 milijuna koraka po sekundi što je jednako ili brže od osnovnih funkcija. Ovo je vrlo važno jer pokazuje da je efikasnost implementacije metoda vrlo visoka. Prethodni izračun vrijednosti kosinusa je znatno ubrzao i brze i spore metode, a korištenje dvostrukog vectora je eliminiralo veliko usporenje u izvođenju,

vjerojatno zbog pristupa L2 cacheu (usporenje je bilo ekvivalentno dodavanju 6 ciklusa unutarnjoj petlji). Iz ovoga također zaključujemo da je prevoditelj uspješno prepoznao statičke matrice i vrlo vjerojatno zadane redove učitava direktno u L1 cache za maksimalnu brzinu izvođenja.

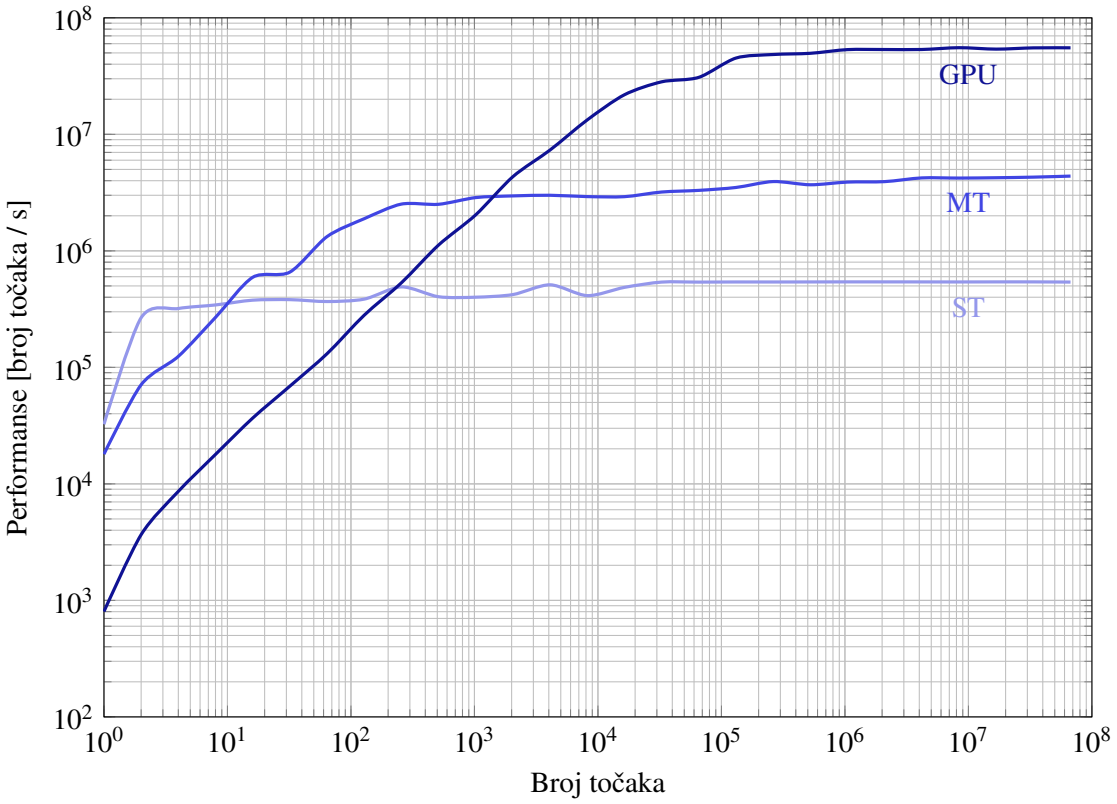
Uočavamo da se performanse ne povećavaju linearno s brojem korištenih dretvi. Na Ubuntu je rast znatno linearniji i izostaju znatne varijacije, dok na Windowsu vidimo ponašanje koje sugerira da dretve nisu prikladno raspoređene među jezgrama. Veličina bloka kod testiranog broja točaka i dalje nije dovoljno velika za ostvarenje punih performansi procesora. Analiza koliko je točaka potrebno za ostvarenje punih performansi provodi se u nešto kasnije u odjeljku.

Promatranjem grafova 16 i 17 uočava se da računalo A na Windows operacijskom sustavu ostvaruje performanse nekoliko puta veće od računala B i C. Zanimljivo je uočiti da računala D i E na Windowsu pokazuju performanse slabije od računala C, a utvrđeno je da se radi o odabiru specifičnog MSVC prevoditelja. Naime, dostupno je više različitih verzija prevoditelja: za 32-bitne x86 sustave, za 64-bitne x64 sustave, za 32-bitne x64 sustave, te za 64-bitne x64 sustave. Ako nije odabrana ispravna arhitektura procesora (64-bitni x86) performanse su znatno snižene kao što vidimo na grafu 16. Na operacijskom sustavu Ubuntu (Pop je zasnovan na Ubuntu), performanse na računalima D i E znatno su brže i usporedive s performansama računala A koje je također nešto brže nego na Windowsu. Zaključujemo da korištenje Ubuntu operacijskog sustava, ili neke druge slične distribucije Linuxa, omogućava iskorištenje punih performansi metoda.

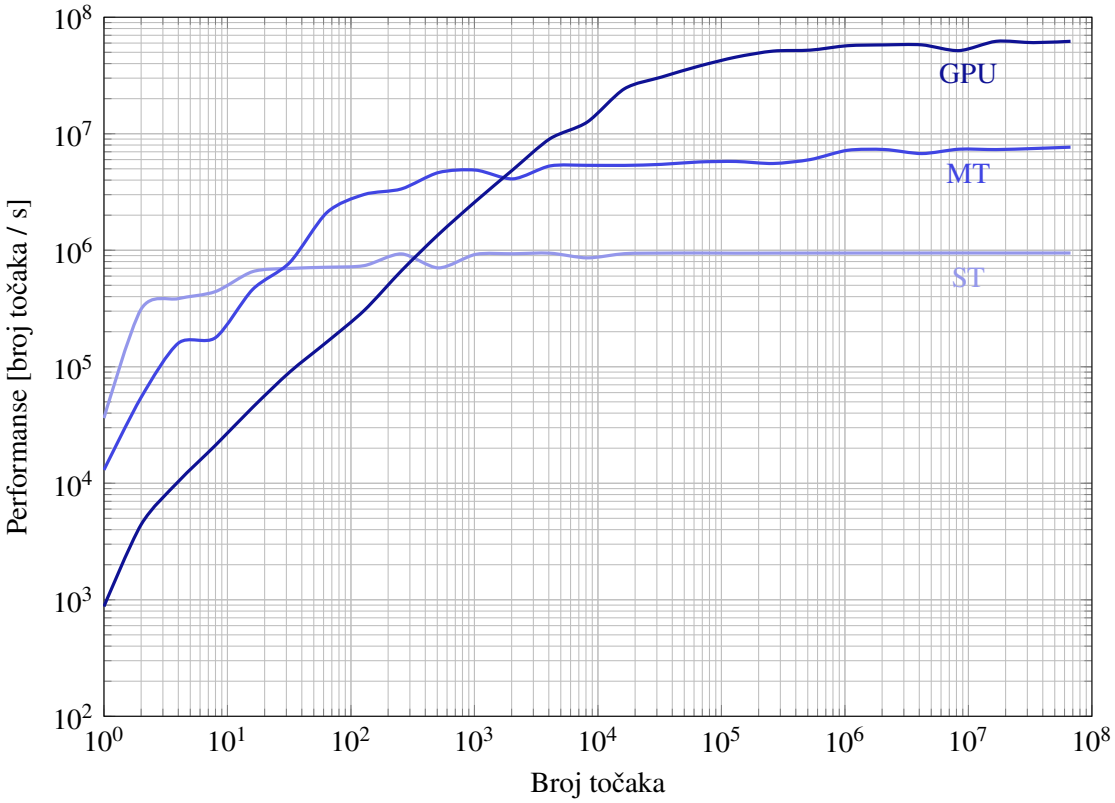
Tijekom testiranja je uočeno da korištenje broja softverskih dretvi jednakog broju procesorskih jezgri koristi svega 75 % teoretskih performansi procesora na sustavu Windows 10. Tek kada je broj dretvi jednak broju procesorskih dretvi i broj točaka dovoljno velik, moguće je dobiti povećavanje performansi približno proporcionalno broju jezgri procesora. Za određivanje ovisnosti performansi o veličini izračuna, potrebno je odabrati jedan faktor preciznosti i broj dretvi te mijenjati broj točaka s kojim se računa. Ovo se redovito naziva skaliranjem performansi i važan je pokazatelj kvalitete izvedbe paralelizacije. U našemu slučaju broj dretvi je jednak 16 jer je toliko procesorskih dretvi na raspolaganju i u tom slučaju očekujemo maksimalne performanse. Faktor preciznosti je postavljen na 3.0 zbog usporedbe s grafičkom karticom. Kao što je već spomenuto, preciznost na grafičkoj kartici je trenutno fiksna pa je potrebno odabrati ekvivalentni faktor preciznosti. Mi smo odabrali fiksni red kvadrature 20 za obje dimenzije integracije, a navedeno je najbliže faktoru preciznosti 3.0 po ukupnom broju inkremenata ($20 \cdot 20 = 400 = 100 \cdot 2^{3-1}$). Uspoređujemo performanse u tri slučaja: kada se koristi samo jedna jezgra, kada se koriste sve jezgre (16 softverskih dretvi za 16 procesors kih), te kada se koristi grafička akceleracija. Testirane su performanse za 1, 2, 4, 8, 16,..., i 67108864 točaka i izražene kao broj točaka po sekundi koji je ostvaren prilikom računanja s danim brojem točaka. Korišteno je logaritamsko mjerilo za obje osi jer je znatno pogodnije za prikaz velikog raspona broja točaka i performansi.

Koristi se oznaka ST (*single thread*) za jednojezgreno izvođenje, oznaka MT (*multithreading*) za potpuno opterećen procesor, a GPU za grafički akcelerirano izvođenje. Uočavamo da jedna jezgra vrlo brzo dostiže svoje maksimalne performanse, no ipak ne odmah. Izgledno je da se radi o trošku pokretanja funkcija i učitavanja podataka za izračun. Nismo sigurni zašto se pojavljuje određena valovitost kod manjeg broja točaka, a potpuno nestane tek na 32768 točaka. Najveće ostvarene performanse su redom 540 tisuća točaka po sekundi za potencijal, 950 tisuća točaka po sekundi za magnetsko polje i 650 tisuća točaka za gradijent. Kao što je očekivano, višedretvenost nije efikasna kada je broj točaka malen, no važno je uočiti da donosi poboljšanje čim je broj točaka veći od dvostrukog broja korištenih dretvi, 32 u ovome slučaju. Kod vektorskog potencijala, koji je najsporiji, znatno poboljšanje performansi vidi se već na 16 točaka. Najveće performanse se dostižu tek kod najvećeg broja točaka i iznose 4.4 milijuna točaka po sekundi za potencijal, 7.7 milijuna točaka po sekundi za polje i 4.9 milijuna točaka po sekundi za gradijent. Približno 70 % performansi postiže se pri izračunu 20 000 točaka za sve 3 metode. Grafička kartica kreće vrlo sporo i pokazuje vrlo niske performanse kada je broj točaka malen, no kako raste broj točaka tako rastu i performanse pa zaključujemo da se radi o visokom trošku pokretanja. Grafička kartica postaje brža od procesora na približno 2000 točaka. Najveće performanse koje postiže iznose 55 milijuna točaka po sekundi za potencijal, 62 milijuna točaka po sekundi za magnetsko polje i 29 milijuna točaka po sekundi za gradijent.

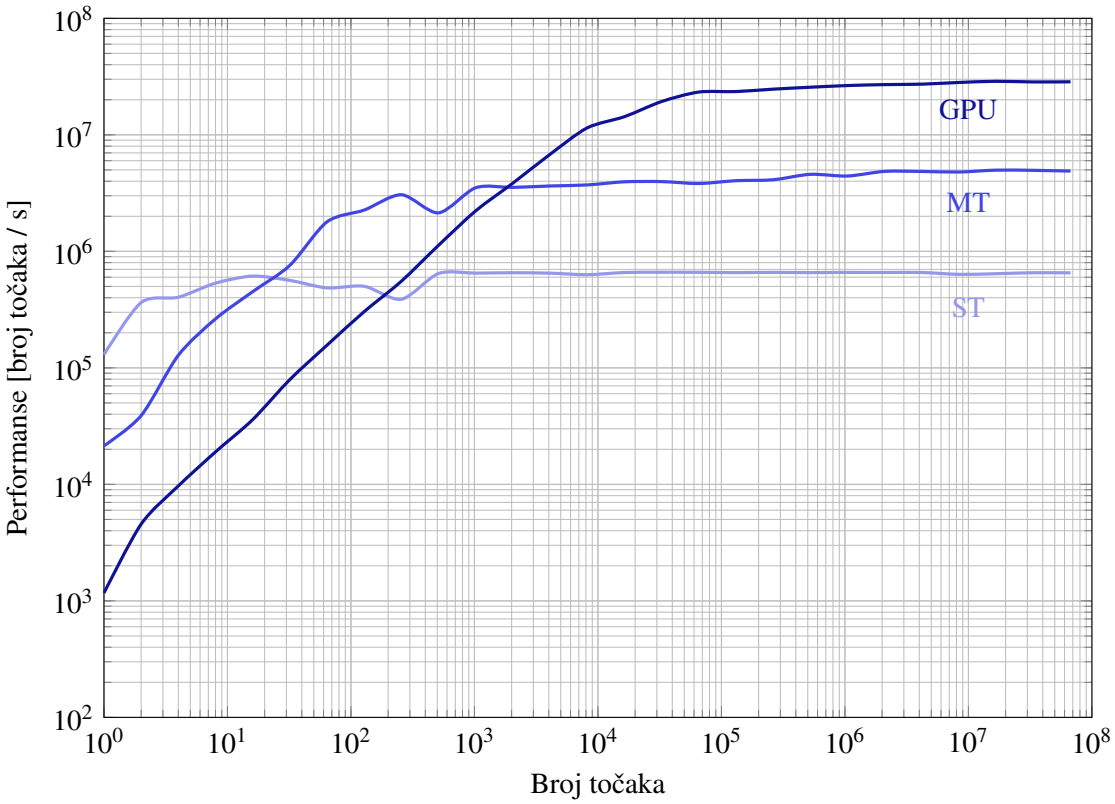
Slika 18: Skaliranje performansi brze metode za izračun potencijala (računalo A Win10)



Slika 19: Skaliranje performansi brze metode za izračun magnetskog polja (računalo A Win10)



Slika 20: Skaliranje performansi brze metode za izračun gradijenta polja (računalo A Win10)



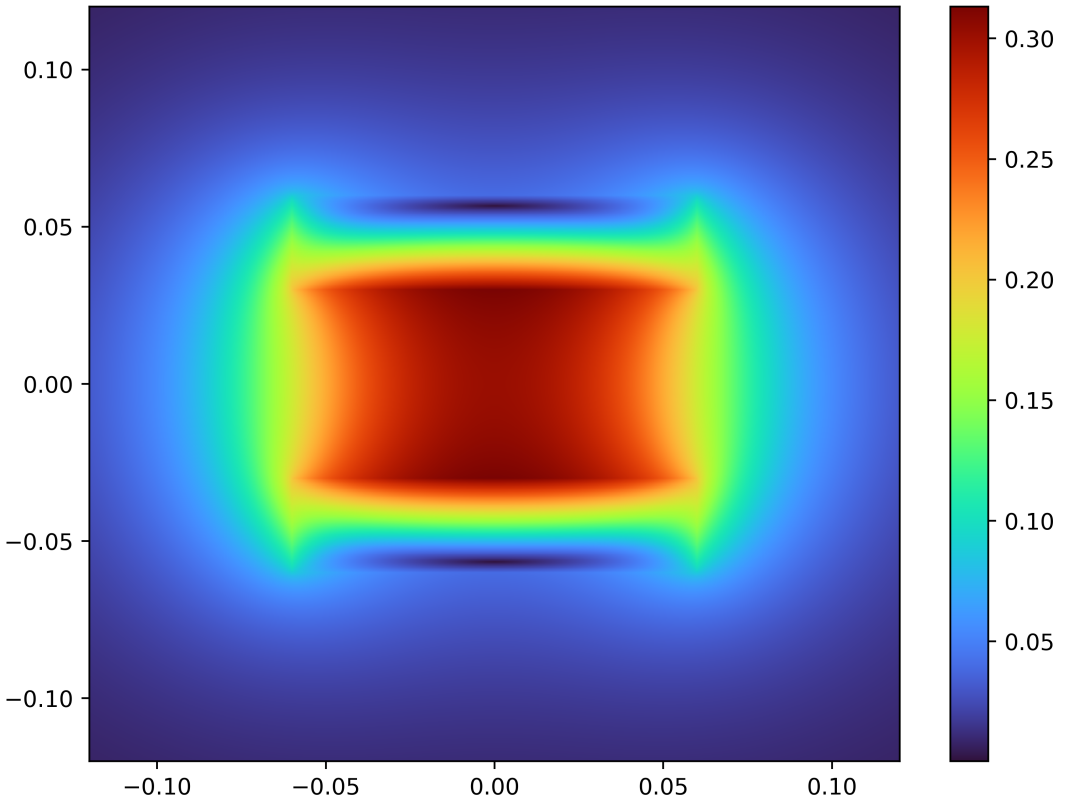
Kako bismo bolje razumjeli ponašanje performansi grafičke kartice potrebno je razmotriti osnovne korake koji se koriste pri implementaciji grafičke akceleracije u kontekstu naše metode, a to su: alokacija radne memorije, pripremanje podataka za grafičku karticu, prijenos podataka u radnu memoriju grafičke kartice, pokretanje i provođenje izračuna, prijenos rezultata izračuna natrag u radnu memoriju procesora i pretvorbe izračunatih vrijednosti u povratni tip podatka funkcije. Priprema i pretvorba približno linearno ovise o broju točaka, a pokazalo se da koriste čak polovicu vremena ukupnog izračuna pa zaključujemo da brzina memorije procesora predstavlja problem. Prijenos podataka s procesora na grafičku karticu, pokretanje izračuna i prijenos podataka natrag u radnu memoriju imaju određenu (vrlo visoku) cijenu pokretanja koja čini korištenje neefikasnim ako je broj točaka malen. Prijenosi podataka su se pokazali iznimno sporima - zauzimaju 60-75 % ukupnog vremena korištenja grafičke kartice. Pokazalo se također da RTX 2080 Ti ne pokazuje bolje performanse od RTX 3080 Mobile zbog sporijeg prijenosa podataka. Brzina prijenosa je na obje kartice iznosila 4-6 GB/s. Ovo je vrlo zanimljivo ponašanje i uočava se na oba računala s tom grafičkom karticom (računala A i D).

Dakle, izračun je daleko najbrži dio grafički akcelerirane metode i približno osmina vremena se provede na njemu, od ukupnog vremena koje se koristi u metodi za izračun. Kada je kod testiran na računalu C i kartici GTX 1050, ukupne performanse su bile svega 2.5 puta niže, djelomično zbog niže brzine prijenosa. Starije i slabije grafičke kartice također mogu znatno povećati dostupne performanse. Brzina memorije i prijenosa je glavno ograničenje koje sprječava bolje performanse. Ovo potvrđuje da je procjena teoretske brzine grafičke kartice bila ispravna i da je osmišljeni pristup pogodan za hardversku akceleraciju. U slučaju istovremenog izračuna polja koje stvara više zavojnica, memorija bi trebala biti znatno manji problem zahvaljujući superpoziciji koja omogućava zbrajanje doprinosa svih zavojnica polju u danoj točki. Zavojnica zauzima znatno manje memorije (oko 1kB) od nekoliko milijuna točaka (desetci MB memorije) pa je prijenos brži. Najveće ostvarive performanse za izračun potencijala bi tada iznosile približno 400 milijuna točaka po sekundi (oko 80 puta više od procesora). Zaključujemo da u budućnosti treba što više proračuna raditi direktno na grafičkoj kartici i izbjegavati bilo kakve konverzije tipova podataka. Ovdje su one bile nužne jer, kao što je već rečeno, ovaj pristup nije specifično osmišljen za grafičku akceleraciju. Glavna prepreka je što komercijalne grafičke kartice ne mogu efikasno koristiti double tip podataka, a upravo on se koristi u ostatku koda. Upravo zato je pretvorba i potrebna.

Ako usporedimo najveće ostvarene performanse, možemo uočiti da korištenje višedretvenosti povećava performanse malo više od 8 puta pa zaključujemo da SMT donosi malena unaprijeđenja performansi. Nadalje, grafička kartica može poboljšati performanse još 6 do 12 puta, a pritom valja imati na umu da je preciznost smanjena na najviše 7 značajnih znamenki. Dakle, već ova osnovna implementacija grafički akceleriranog pristupa pokazuje koliko je grafička kartica moćnija u usporedbi s procesorom i njen velik potencijal za budući rad.

Za kraj ovog dugačkog i pomnog razmatranja, prikazuje se slika apsolutne vrijednosti magnetskog polja odabrane zavojnice. Postavljena je u ishodište i orijentirana tako da joj presjek leži u xOy ravnini. Slika 21 generirana je korištenjem našeg Python modula i modula Matplotlib [25] za generiranje slike. Rezolucija je postavljena na 3000x3000, efektivno 9 milijuna točaka, i faktor preciznosti je postavljen na 9.0 za detaljan prikaz polja unutar zavojnice. Za izračun i generiranje slike trebalo je oko 3 minute. Naravno, moguće je dobiti znatno bolje performanse korištenjem grafičke kartice, no cilj je bio ispitati konvergenciju izraza za polja i odgovor je - uglavnom konvergiraju. Potencijal i magnetsko polje konvergiraju, a jedna komponenta gradijenta divergira u rubnim točkama zavojnice (na desnom i lijevom rubu zavojnice) pa je potrebno dodatno filtrirati abnormalno visoke vrijednosti. Kod nižih faktora preciznosti vidjele bi se paralelne crte unutar zavojnice, jer pravokutna zavojnica se zapravo sastoji od niza specifično odabranih tankih zavojnica. Kod vanjskog polja se ne uočava razlika.

Ovime se zaključuju sva relevantna razmatranja performansi metoda polja i može se nastaviti s detaljnim testiranjem preciznosti.



Slika 21: Prikaz apsolutne vrijednosti magnetskog polja u T. Zavojnica ima dimenzije $R = 0.03$ m, $a = 0.03$ m, $b = 0.12$ m te 3600 navoja kroz koje teče struja od 10 Ampera

5.3 Međuindukcija

5.3.1 Slučaj zajedničke osi

Najbolji test preciznosti pristupa za računanje polja je njegova primjena na slučajevima izračuna međuindukcije, sile i zakretnog momenta. Očekujemo da će naš pristup imati problema kad su zavojnice jako blizu jedna drugoj, odnosno da se neće dobiti svih 15 značajnih znamenaka preciznosti. Naime, za ostvariti željenu preciznost, potreban je sve veći red Gauss-Legendre kvadrature što su zavojnice bliže jedna drugoj. Smanjivanje tog razmaka stoga povećava pogrešku u slučaju konstantnog faktora preciznosti. Prvo ćemo proći slučajeve sa zajedničkom z-osi.

Tablica 2: Testni podatci za međuindukciju u slučaju zajedničke z-osi, preuzeti iz [7] te pretvoreni u veličine i koordinate korištene za ovaj pristup, $x_1,\ y_1, z_1, x_2, y_2$ su postavljeni na 0

slučaj	R_1 (m)	a_1 (m)	b_1 (m)	N_1 ()	R_2 (m)	a_2 (m)	b_2 (m)	N_2 ()	z_2 (m)
1	0.1	0.1	0.1	100	0.3	0.1	0.1	100	0.2
2	0.5	1	1	1	0.5	1	1	1	1.001
3	0.0875	0.025	0.025	200	0.0875	0.025	0.025	200	0.06
4	0.2	0.2	0.2	500	0.6	0.2	0.2	500	0
5(a)	0.0287	0.022	0.0302	10	0.0287	0.022	0.0302	10	0.03021
5(b)	0.0287	0.022	0.0302	10	0.0287	0.022	0.0302	10	0.04
5(c)	0.0287	0.022	0.0302	10	0.0287	0.022	0.0302	10	0.065
6	0.1	0.1	0.1	100	0.1	0.1	0.1	100	0.2
7	1.2	0.05	0.25	100	1	0.05	0.2	10	0.525

Tablica 3: Usporedba rezultata izračuna međuindukcije u slučaju zajedničke z-osi, podaci iz tablice 2, faktor preciznosti 5.0, vrijeme računanja je manje od 0.2 milisekunde (Računalo A Win10)

slučaj	$M_{z,12}$ iz [7]	$M_{z,12}$ naš pristup	apsolutna relativna pogreška
1	0.8454457615296840 mH	0.8454457615296840 mH	0.0
2	0.539456018627887 µH	0.5394558463523180 µH	3.19E-07
3	3.737280536931003 mH	3.737280536931000 mH	8.03E-16
4	70.77152436821374 mH	70.77152436821373 mH	1.41E-16
5(a)	3.108645299521619 µH	3.108644959497476 µH	1.09E-07
5(b)	2.117124447694834 µH	2.117124447694834 µH	0.0
5(c)	0.9163970216044788 µH	0.9163970216044781 µH	7.64E-16
6	0.4920371955116460 mH	0.4920371955116457 mH	6.10E-16
7	1.309176459700494 mH	1.309176459700493 mH	7.638E-16

U tablici 3 možemo vidjeti da se rezultati našeg pristupa izvrsno slažu s onima iz [7] u većini testiranih slučajeva te da se odstupanja pojavljuju kad su zavojnice jako blizu jedna drugoj. Preciznost se dalje može poboljšati korištenjem većeg faktora preciznosti, no odabrani je dobra kombinacija preciznosti i brzine. U dodatnom testiranju vidjeli smo da čak i faktor preciznosti 1.0 može dati razumne rezultate s pogreškama od 1.43E-5 do 1.04E-15 u otprilike 0.04 ms (milisekundi). Dakle u slučajevima u kojima su zavojnice nešto udaljenije već niski faktori preciznosti omogućavaju vrlo precizan izračun zbog brze konvergencije.

Budući da je broj inkremenata konstantan za zadani faktor preciznosti, očekujemo da će vremena izvođenja biti slična bez obzira na raspored i dimenzije zavojnica. Mala odstupanja mogu se pojaviti zbog prekoračenja broja inkremenata uzrokovanih algoritmom za balansiranje broja inkremenata te razlika u performansama metode za izračun potencijala koje se mogu vidjeti na grafu 4. Rezultati su prikazani u tablici 4. Važno je uočiti da broj korištenih dretvi nije fiksan već se mijenja ovisno o procesoru i metodi. Prilikom testiranja je odlučeno da se koristi najveći broj dretvi koji donosi konzistentno poboljšanje performansi na danome sustavu.

Razumna preciznost (greška reda 10^{-7}) može se postići u 0.2 ms na računalu A, dok pristup prikazan u [7] navodi vrijeme izvođenja od 30 ms. Kako je rad napisan 2014. godine, zaključujemo da bi na modernom sustavu vrijeme izvođenja vjerojatno bilo dosta kraće i iznosilo bliže 10 ms. Višedretvenost ima malen utjecaj na performanse pri nižim faktorima preciznosti, a on raste kako se faktor preciznosti povećava. Skaliranje performansi je dosta loše i najbolje je na računalu C koje ima samo 4 jezgre a ostvaruje 3 puta veće performanse. Na operacijskom sustavu Ubuntu ova metoda je znatno brža nego na Windowsu, do te mjere da paralelizacija ne donosi nikakvo unaprjeđenje performansi.

Tablica 4: Brzine izvođenja za međuindukciju u slučaju zajedničke z-osi s obzirom na faktor preciznosti na više različitih računala i operacijskih sustava, slovo predstavlja korišteno računalo, a broj uz T broj korištenih dretvi, sva vremena su u milisekundama

faktor preciznosti	inkrementi	A Windows		A Ubuntu		B Windows		C Windows	
		1T	12T	1T	4T	1T	8T	1T	8T
1.0	1 000	0.039	0.022	0.014	0.019	0.057	0.040	0.075	0.044
2.0	2 000	0.076	0.033	0.028	0.026	0.111	0.059	0.128	0.068
3.0	4 000	0.160	0.053	0.057	0.050	0.228	0.104	0.238	0.109
4.0	8 000	0.306	0.085	0.091	0.083	0.444	0.189	0.457	0.191
5.0	16 000	0.630	0.178	0.174	0.167	0.913	0.333	0.925	0.390
6.0	32 000	1.252	0.343	0.349	0.303	1.811	0.705	1.849	0.683
7.0	64 000	2.430	0.608	0.695	0.673	3.536	1.268	3.579	1.204
8.0	128 000	4.797	1.318	1.427	1.176	6.980	2.622	6.946	2.377

5.3.2 Općeniti slučaj

Sljedeći primjer, koji se često pronalazi u literaturi, su zavojnice koje leže na paralelnim osima. Rad [8] nudi mnogo primjera i prikazat ćemo njih 20. Fokusiramo se na one koji za faktor preciznosti 5.0 daju apsolutnu relativnu pogrešku veću od 0. Svi slučajevi iz [8] su testirani, a svi koji nisu prikazani ovdje imaju relativnu pogrešku 0 za faktor preciznosti veći ili jednak 3.0. U skladu s njihovim primjerima, središte prve zavojnice se nalazi u ishodištu, a središte druge u xOz ravnini. Uz to, svi kutovi rotacije su postavljeni na 0.

Tablica 5: Testni podatci za međuindukciju u slučaju paralelnih osi, preuzeti iz [8] te pretvoreni u veličine i koordinate korištene za ovaj pristup: x_1,y_1, z_1, y_2 su postavljeni na 0

slučaj	R_1 (m)	a_1 (m)	b_1 (m)	N_1 ()	R_2 (m)	a_2 (m)	b_2 (m)	N_2 ()	z_2 (m)	x_2 (m)
1	0.1	0.1	0.65	1	0.3	0.1	0.36	1	0.005	0.01
2	0.1	0.1	2.5	1	0.3	0.1	0.4	1	0.25	0.05
3	0.1	0.1	2.2	1	0.3	0.1	0.5	1	0.35	0.08
4	0.1	0.1	1.8	1	0.3	0.1	0.3	1	0.15	0.1
5	0.1	0.1	1.5	1	0.3	0.1	1.5	1	0.0	0.02
6	0.1	0.1	0.4	1	0.3	0.1	0.27	1	0.035	1.2
7	0.1	0.1	1.2	1	0.3	0.1	0.7	1	0.25	2.2
8	0.1	0.1	1.3	1	0.4	0.1	1.5	1	1.0	0.05
9	0.4	0.1	2.2	1	0.1	0.1	1.7	1	1.15	0.02
10	0.4	0.1	2.1	1	0.1	0.1	1.0	1	0.55	0.15
11	0.1	0.1	2.8	1	0.4	0.1	2.8	1	0.0	0.12
12	0.2	0.2	1.3	1	0.3	0.2	0.9	1	2.2	0.15
13	0.2	0.1	0.3	1	0.7	0.8	0.15	1	0.025	0.2
14	0.1	0.2	0.2	1	0.9	0.4	0.13	1	0.015	0.3
15	0.6	0.1	0.3	1	0.9	0.5	0.2	1	0.05	0.1
16	0.4	0.2	0.4	1	1.0	0.2	0.3	1	0.05	0.2
17	0.7	0.2	3.2	1	1.1	0.1	3.0	1	0.0	2.7
18	0.2	0.2	1.7	1	0.3	0.2	0.6	1	1.15	1.6
19	0.5	0.2	2.7	1	0.9	0.4	2.4	1	0.05	2.4
20	0.3	0.2	1.4	1	0.4	0.2	0.8	1	0.3	1.5

Većina pogrešaka rezultata u tablici 6 smanjuje se povećanjem faktora preciznosti te relativne pogreške padaju na vrijednosti ispod 1.32E-11, odnosno u većini slučajeva iščeznu. Neki posebni slučajevi imaju pored rezultata „(?)”. U 7. slučaju, dobili smo vrijednost točno dva puta manju od one koju su autori dobili. U 20. slučaju se razlikuju tri znamenke: posljednje dvije (89 umjesto 90, ali postanu iste na većoj preciznosti) i 6. značajna znamenka (1 umjesto 7). Stoga je razumno pretpostaviti da se radi o pogrešci unosa podataka. Za 8. slučaj nemamo dobro objašnjenje, vjerojatno je neki parametar pogrešno upisan. Moguće je da smo pogrešno pretvorili dimenzije, ali nakon mnogo provjera pogreška

nije pronađena.

Tablica 6: Usporedba rezultata izračuna međuindukcije u slučaju paralelnih osi, podaci iz tablice 5, faktor preciznosti 5.0, vrijeme računanja je 4.5 milisekunde (Računalo A Win10)

slučaj	M_{12} (nH) iz [8]	M_{12} (nH) ovog pristupa	apsolutna relativna pogreška
1	95.153726129186	95.153726129186	0.0
2	35.306858110658	35.306858110658	0.0
3	39.227167276179	39.227167276028	3.85E-12
4	47.500980052677	47.500980078214	5.37E-10
5	49.154516729840	49.154516729848	1.63E-13
6	-1.731092052113	-1.731092052113	0.0
7	-0.455473787425	-0.227736893712	0.500 (?)
8	24.40371190625	19.996924468515	0.181 (?)
9	19.377233946763	19.377233946175	3.03E-11
10	35.677454532503	35.677454532502	2.80E-14
11	28.224715973988	28.224715973976	4.25E-13
12	3.290633000207	3.290633000207	0.0
13	124.431891194311	124.431891194311	0.0
14	84.962100847746	84.962100847746	0.0
15	849.545393862007	849.545393862005	2.35E-15
16	496.558909768082	496.558909768083	2.01E-15
17	-25.207800573761	-25.207800573761	0.0
18	-0.396557065745	-0.396557065745	0.0
19	-20.342933936402	-20.342933936402	0.0
20	-8.583874278590	-8.583814278589	6.99E-06 (?)

Tablica 7: Brzine izvođenja za općenite slučajeve međuindukcije s obzirom na faktor preciznosti na više računala i operacijskih sustava, slovo označava korišteno računalo, a broj uz T broj korištenih dretvi, sva vremena su u milisekundama

faktor preciznosti	inkrementi	A Windows		A Ubuntu		B Windows		C Windows	
		1T	12T	1T	12T	1T	12T	1T	8T
1.0	100 000	0.636	0.116	0.587	0.113	1.269	0.201	1.101	0.390
2.0	200 000	1.390	0.226	1.160	0.200	2.714	0.346	2.251	0.785
3.0	400 000	2.315	0.384	2.023	0.334	4.711	0.540	3.835	1.413
4.0	800 000	4.034	0.679	3.892	0.637	8.477	0.965	6.843	2.436
5.0	1 600 000	8.666	1.372	7.938	1.261	18.31	1.863	14.72	4.889
6.0	3 200 000	14.96	2.369	14.00	2.220	32.28	3.082	24.91	7.808
7.0	6 400 000	30.50	4.727	29.00	4.492	66.22	6.241	51.96	14.46
8.0	12 800 000	52.97	8.042	53.58	8.453	116.9	10.75	89.88	25.07

Podatci za performanse mogu se vidjeti u tablici 7. U kontekstu naše metode slučaj paralelnih osi je programski ostvaren jednako kao općeniti slučaj. Stoga performanse koje su prikazane u tablici 7 su zapravo performanse u općenitom slučaju. Ovo je važno jer opisana metoda za izračun u slučaju paralelnih osi ima vremena izvođenja od 30-ak ms do otprilike 4 sekunde, ovisno o tome kakav je međusobni položaj zavojnica. Razlog leži u činjenici da metoda [8] ima dva slučaja, brži slučaj za određene konfiguracije i sporiji slučaj koji se temelji na poboljšanju pristupa prikazanog u radu [4]. Naš pristup pokazuje odlične performanse i vrlo dobro skaliranje performansi, pogotovo kod računala B koje je dobilo priliku pokazati da raspolaže sa 16 jezgara. Vrlo je zanimljivo da je ovo jedini slučaj u kojemu je Ubuntu pokazao performanse malo lošije od Windowsa. Već niži faktori preciznosti daju vrlo precizne rezultate s vremenima izračuna reda 1 ms.

Dalje se prikazuju rezultati iz rada [3] kako bismo demonstrirali mogućnost obrade rotacije. Preciznost podataka iz spomenutog rada je ograničena, ali i dalje vrlo korisna. Budući da je tanka zavojnica zapravo skup navoja, središta svih navoja neće ležati na z-osi kada se zavojnica zarotira. To znači da istovremeno imamo slučaj rotacije i paralelnih osi, što je po karakteru vrlo slično općenitom slučaju. Rotacija opisana u [3] je zapisana s θ_2 - ovdje ćemo promatrati samo jedan od jedanaest kutova. Ostali će biti postavljeni na 0. Za neke testne slučajeve dani su precizni teoretski podaci. Uspoređivat ćemo rezultate s tim podacima kad je to moguće, inače s rezultatima pristupa opisanog u radu. Budući da je vrijeme izračuna u ovim slučajevima vrlo kratko, za faktor preciznosti se odabire 8.0. Prije računanja relativnih pogrešaka, rezultate ćemo zaokružiti na jednak broj decimala kao u [3]. Također, kao i u tom radu, središte prve zavojnice je ishodište, a središte druge nalazi se na z-osi.

Tablica 8: Testni slučajevi međuindukcije s rotacijom oko paralelnih osi, preuzeti iz [3] te pretvoreni u veličine i koordinate korištene za ovaj pristup: x_1, y_1, z_1, x_2 i y_2 postavljeni na 0

slučaj	R_1 (m)	a_1 (m)	b_1 (m)	N_1 ()	R_2 (m)	a_2 (m)	b_2 (m)	N_2 ()	z_2 (m)	$\cos\theta_2$ ()
1	0.06	0.0	0.12	120	0.05	0.0	0.0	1	0.0	0.5
2	0.06	0.0	0.12	120	0.05	0.0	0.0	1	0.03	0.4
3	0.06	0.0	0.12	120	0.05	0.0	0.0	1	0.06	0.3
4	0.06	0.0	0.12	120	0.05	0.0	0.0	1	0.12	0.2
5	0.06	0.0	0.12	120	0.05	0.0	0.04	60	0.0	0.6
6	0.06	0.0	0.12	120	0.05	0.0	0.04	60	0.0	0.5
7	0.06	0.0	0.12	120	0.05	0.0	0.04	60	0.0	0.4
8	0.06	0.0	0.12	120	0.05	0.0	0.04	60	0.0	0.3
9	0.04	0.02	0.0	200	0.015	0.01	0.0	100	0.05	0.1
10	0.04	0.02	0.01	100	0.02	0.0	0.0	1	0.0	0.9

Tablica 9: Usporedba rezultata izračuna međuindukcije s rotacijom oko paralelnih osi, podaci iz tablice 8, faktor preciznosti 8.0, vrijeme računanja je uvijek manje od 1 milisekunde (Računalo A Win10)

slučaj	M_{12} (µH) iz []	M_{12} (µH) ove metode	apsolutna relativna pogreška
1	3.2871	**3.28707**633977599	0.0
2	2.3353	**2.33528**204096537	0.0
3	1.3053	**1.30532**695112295	0.0
4	0.2996	**0.29963**7197170154	0.0
5	243.5900	**243.58998**3113468	0.0
6	179.8060	**179.80601**7611640	0.0
7	104.9544	**104.95435**9808496	0.0
8	25.4067	**25.40673**74343466	0.0
9	12.0800	**12.08005**84796149	4.84E-06
10	1.4792	**1.47918**909915092	0.0

Rezultati u tablici 9 odlično se slažu s onima iz [3]. Nećemo detaljnije komentirati performanse jer ovaj tip zavojnice nije fokus našeg rada. One će ipak biti barem red veličine brže nego što je prikazano u tablici 7. Korištenje višedretvenosti u ovim slučajevima samo smanjuje performanse i općenito se ne preporuča ako primarna zavojnica nije pravokutna.

Posljednji dio testiranja uključuje dva potpuno općenita slučaja koje smo uspjeli pronaći u literaturi, a oni su iz [9]. Za razliku od prijašnjih slučajeva, ovi uključuju i rotaciju i pomak osi dviju pravokutnih zavojnica. Preciznost izračuna razmatra se detaljno za različite faktore preciznosti. Prikazat će se samo relativne pogreške kako tablica ne bi zauzimala odviše prostora. U [9], udaljenost između središta zavojnica je definirana preko cilindričnih koordinata, a pritom je jedna zavojnica nužno u ishodištu. Ovdje su umjesto cilindričnih parametara (z,r) uzete koordinate x i z. Općeniti slučaj karakteriziraju te dvije udaljenosti i dva kuta rotacije. Ovo su zapravo relativni položaji dviju zavojnica. Cilj našeg rada je mogućnost izračuna međuindukcije između proizvoljnog broja zavojnica u proizvoljnim položajima i orijentacijama pa su položaj i orijentacija svake zavojnice definirani s 5 parametara. Zato se naše koordinate moraju prilagođavati dostupnim testnim podacima, a prilagođene koordinate su prikazane u tablici 10.

Tablica 10: Općeniti slučajevi iz [9] pretvoreni u mjere i koordinate korištene za ovaj pristup; x_1, y_1, z_1 i y_2 postavljeni na 0

Slučaj	R_1 (m)	a_1 (m)	b_1 (m)	N_1 ()	R_2 (m)	a_2 (m)	b_2 (m)	N_2 ()	z_2 (m)	x_2 (m)
1	0.1	0.02	0.24	1200	0.04	0.01	0.06	1200	0.0	0.0
2	0.1	0.02	0.24	1200	0.04	0.01	0.06	1200	0.1	0.18

Tablica 11: Rezultati primjene našeg pristupa na podacima iz tablice 10, decimalni brojevi u gornjem retku predstavljaju faktore preciznosti, apsolutne relativne pogreške su prikazane za određeni slučaj i kut rotacije, ostala 3 kuta su postavljena na 0

slučaj	$\cos\theta_2$	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0
1	1.0	1.7E-15	0.0	2.8E-16	2.8E-16	2.8E-16	2.8E-16	2.8E-16	2.8E-16
1	0.9	6.2E-16	6.2E-16	9.4E-16	3.1E-15	1.2E-15	0.0	1.2E-15	0.0
1	0.8	2.5E-15	3.9E-15	0.0	1.8E-15	3.5E-16	0.0	5.3E-15	3.9E-15
1	0.7	6.0E-15	6.0E-15	6.0E-15	6.0E-15	6.8E-15	4.8E-15	6.0E-15	3.6E-15
1	0.6	5.2E-15	3.8E-15	4.7E-15	2.8E-15	2.8E-15	2.8E-15	9.4E-16	5.2E-15
1	0.5	5.7E-15	3.4E-15	6.2E-15	5.1E-15	4.0E-15	4.0E-15	5.7E-15	6.2E-15
1	0.4	8.5E-15	5.0E-15	5.0E-15	5.7E-15	5.0E-15	5.0E-15	5.0E-15	4.2E-15
1	0.3	1.0E-14	4.7E-15	2.8E-15	4.7E-15	5.7E-15	1.9E-15	2.8E-15	1.9E-15
1	0.2	1.7E-14	9.2E-15	4.8E-15	6.1E-15	2.8E-15	5.8E-15	6.0E-15	4.7E-15
1	0.1	3.6E-14	2.6E-14	2.0E-14	1.9E-14	2.1E-14	1.9E-14	2.0E-14	1.5E-14
2	1.0	4.4E-11	4.0E-13	4.6E-14	1.1E-14	8.7E-15	1.2E-14	1.1E-14	8.2E-15
2	0.9	1.7E-11	8.5E-12	1.5E-14	6.5E-14	9.5E-15	1.1E-14	7.1E-15	3.9E-15
2	0.8	1.9E-10	6.0E-11	8.3E-12	1.5E-12	2.0E-14	4.9E-15	6.9E-15	5.8E-15
2	0.7	1.5E-10	4.7E-11	5.3E-10	4.7E-12	8.3E-13	7.4E-13	6.9E-14	2.2E-14
2	0.6	1.4E-08	4.9E-09	3.4E-09	2.2E-10	9.1E-11	1.3E-14	4.2E-12	1.2E-13
2	0.5	1.6E-08	1.5E-09	1.6E-09	9.4E-11	3.2E-12	1.5E-12	4.2E-13	4.2E-14
2	0.4	8.3E-08	5.9E-09	5.9E-09	4.6E-10	9.5E-11	1.6E-11	2.2E-12	4.5E-13
2	0.3	7.0E-08	5.7E-10	5.5E-10	3.5E-10	5.0E-12	1.2E-11	3.3E-12	2.6E-13
2	0.2	4.2E-08	1.1E-08	1.1E-08	5.6E-10	3.8E-11	1.3E-11	1.0E-12	1.3E-13
2	0.1	3.5E-08	9.2E-09	9.2E-09	5.1E-10	1.9E-11	7.2E-12	2.1E-12	8.3E-14

Tablica 11 pokazuje jako dobro slaganje rezultata te se u mnogo slučajeva postiže najbolja preciznost od 15 značajnih znamenaka. Ipak, u najzahtjevnijem slučaju, s velikim kutom rotacije, dio preciznosti je izgubljen, ali je najveća pogreška uvijek manja od 1E-07. Ako se faktor preciznosti postavi na 5.0, osigurava se preciznost na 10 značajnih znamenaka s vremenom računanja od 1.3 do 5 milisekundi, ovisno o korištenom računalu. Ovo je veliko poboljšanje u usporedbi s vremenom izvođenja od 8 sekundi za 8 značajnih znamenki prikazanim u radu [9]. Pritom je ostvarena veća ili jednaka preciznost jer metoda [9] postiže preciznost od 15 značajnih znamenki tek kada je vrijeme izvođenja približno 10 minuta. Izračun rezultata korištenih za usporedbu trajao je između 15 i 120 minuta. Faktori preciznosti veći ili jednaki 8.0 neznatno povećavaju preciznost i potrebno im je više vremena. Ipak, vrijeme izračuna tada iznosi 8-25 milisekundi, odnosno performanse su i dalje dovoljno visoke da je korištenje višeg faktora preciznosti opravdano, pogotovo ako su zavojnice vrlo blizu. Mogu se dobiti još veće performanse korištenjem grafičke kartice (vrijeme izračuna manje od 0.1 ms), no pogreške su tada uvijek reda 10^{-6} ili 10^{-7}.

Posljednji i najzahtjevniji slučaj koji će se razmotriti je izračun samoindukcije.

5.3.3 Samoindukcija

Slučaj proračuna samoindukcije je složen i predstavlja rubni slučaj za odabranu numeričku metodu integracije. Naime, suma za međuindukciju pravokutne zavojnice (81) u srži nije stabilna. Zahvaljujući balansiranju inkremenata ipak funkcionira s ograničenom preciznošću za pravokutnu i tanku zavojnicu. Kod plosnate zavojnice su pogreške reda 0.1 %, a samoindukciju beskonačno tanke petlje je nemoguće izračunati jer se analitičkim proračunom dobije da integral divergira. Uzeti su primjeri iz [4].

Kako je i pretpostavljeno, podaci za samoindukciju u ovim slučajevima pokazuju ograničenja naše metode. Pogreška manja od 0.01 % u mnogim je slučajevima prihvatljiva za aproksimaciju samoindukcije. Konvergencija je prilično spora, a pogreška oscilira s porastom faktora preciznosti i zadržava se na približno 3E-07. Uz više inkremenata uvijek postaje manja od 0.001 %. Ipak, ne preporučujemo korištenje ove metode za samoindukciju ako je preciznost od velike važnosti. Ovo je najgori slučaj za ovaj pristup, ali također pokazuje da može tolerirati zavojnicu unutar zavojnice - možemo računati potencijal unutar zavojnice, ali s ograničenom preciznošću. Performanse su dane u tablici 14. Metoda nije paralelizirana jer se pokazalo da paralelizacija smanjuje performanse. Ovo omogućava korištenje relativno visokog faktora preciznosti uz brzo vrijeme izvođenja, svega 1.5 ms za faktor preciznosti 10.0.

Tablica 12: Slučajevi za samoindukciju iz [4], relativne dimenzije definirane u radu su $\alpha = (R+a)/R$ i $\beta = b/(2R)$, odabran je $R = 0.1$ m

slučaj	α	β	R (m)	a (m)	b (m)	N
1	1.5	0.25	0.1	0.05	0.05	1
2	3.0	1.0	0.1	0.2	0.2	1
3	4.0	3.0	0.1	0.3	0.6	1
4	7.0	6.0	0.1	0.6	1.2	1
5	9.0	4.0	0.1	0.8	0.8	1

Tablica 13: Rezultati testiranja na podacima iz tablice 12 s dva faktora preciznosti: 5.0 i 10.0, vrijednosti samoindukcije dane su u μH/m)

slučaj	L/N^2R	L/N^2R 5.0	L/N^2R 10.0	Aps. relativna pogreška 5.0	Aps. relativna pogreška 10.0
1	2.8693034861	2.869339925	2.869304167	1.27E-05	2.37E-07
2	2.5330065469	2.533284451	2.533005576	1.10E-04	3.83E-07
3	1.9012957998	1.901393247	1.901295747	5.13E-05	2.77E-08
4	2.4472978794	2.447271850	2.447296959	1.06E-05	3.76E-07
5	4.2674017935	4.267334461	4.267385419	1.58E-05	3.74E-06

Tablica 14: Brzina izvođenja za samoindukciju s obzirom na faktor preciznosti na više različitih računala i operacijskih sustava, slovo predstavlja korišteno računalo, a broj uz T broj korištenih dretvi, sva vremena su u milisekundama

faktor preciznosti	inkrementi	A Windows	A Ubuntu	B Windows	C Windows
1.0	1 000	0.0050	0.0049	0.0099	0.0097
2.0	2 000	0.0088	0.0087	0.0179	0.0173
3.0	4 000	0.0155	0.0162	0.0333	0.0296
4.0	8 000	0.0284	0.0302	0.0624	0.0537
5.0	16 000	0.0519	0.0567	0.1164	0.0966
6.0	32 000	0.0992	0.1070	0.2214	0.1853
7.0	64 000	0.2106	0.2230	0.4766	0.4179
8.0	128 000	0.4059	0.4312	0.9285	0.7278
9.0	256 000	0.7500	0.8269	1.7377	1.2220
10.0	512 000	1.4883	1.6342	3.4611	2.4883
11.0	1 024 000	3.1670	3.3768	7.2762	5.5088
12.0	2 048 000	5.9734	6.5103	13.865	9.7900

5.4 Sila i zakretni moment

Testiranje metoda za Amperovu silu bit će manje detaljno opisano zato što su metode u srži jednako implementirane pa ako su one za međuindukciju vrlo precizne, razumno je pretpostaviti da su i one za silu. Konvergencija je utvrđena na temelju slike na kraju odjeljka 5.2. Dodatno, nismo uspjeli pronaći mnogo relevantne literature za generalni slučaj dviju pravokutnih zavojnica što je donekle neobično jer poznavanje sile i zakretnog momenta između dviju supravodljivih magneta može biti vrlo korisno. U našem slučaju, sila i zakretni moment se računaju brže od međuindukcije zato što je računanje magnetskog polja brže od računanja vektorskog potencijala. Potencijal u implementaciji sadrži logaritam koji znatno usporava izvođenje kao što je diskutirano u 4.2 te prikazano u odjeljku 5.2.

5.4.1 Slučaj zajedničke osi

Prvi test je računanje sile između dviju tankih i dviju plosnatih zavojnica. Slučajevi su preuzeti iz [12]. Budući da su svi izračuni iznimno brzi, odabrali smo faktor preciznosti 8.0 za prikazane podatke.

Naš pristup ovdje jako dobro funkcionira - sila tankih zavojnica izračuna se u približno 0.02 ms, a plosnatih u približno 0.3 ms (računalo A Win10). Tablica 16 pokazuje da se svi rezultati podudaraju. U slučaju br. 2 zavojnice su jako blizu (što smo spomenuli da je problematično za ovu metodu) pa je bio potreban faktor preciznosti 9.0 da bi rezultat dobro konvergirao.

Tablica 15: Testni slučajevi za Amperovu silu u slučaju z-osi preuzeti iz [12], mjere i koordinate su pretvorene u one korištene u ovome pristupu, x_1, y_1, z_1, x_2 i y_2 su 0

slučaj	R_1 (m)	a_1 (m)	b_1 (m)	N_1I_1 (A)	R_2 (m)	a_2 (m)	b_2 (m)	N_2I_2 (A)	z_2 (m)
1	0.5	0.0	0.3	200	0.5	0.0	0.2	100	0.4
2	0.1	0.0	0.15	225	0.09	0.0	0.15	300	0.05
3	0.25	0.0	0.06	40	0.2	0.0	0.04	120	0.3
4	0.762	0.0832	0.0	732.72	0.762	0.0832	0.0	732.72	0.0468
5	0.16	0.12	0.0	1000	0.11	0.15	0.0	1000	0.05
6	0.12	0.11	0.0	1000	0.12	0.11	0.0	1000	0.02
7	0.1	0.0	0.2	100	0.2	0.2	0.0	100	0.6

Tablica 16: Rezultati pristupa na podacima iz tablice 15, faktor preciznosti je postavljen na 8.0, sila je uvijek privlačna (u smjeru negativne z-osi)

slučaj	$F_{z,12}$ iz [12]	$F_{z,12}$ ove metode	apsolutna relativna pogreška
1	23.83657567265133 mN	23.83657567265137 mN	1.68E-15
2	45.80457156637387 mN	45.80457156637362 mN	5.46E-15
3	1.869154790142725 mN	1.869154790142795 mN	3.75E-14
4	1.050694343958323 N	1.050694343958318 N	4.76E-15
5	2.586692824396309 N	2.586692824396240 N	2.67E-14
6	4.1507739 N	**4.15077392**3097935 N	0.0
7	0.23120076 mN	**0.2312007758**282652 mN	0.0

Detaljnija analiza performansi za silu između dvije pravokutne zavojnice na istoj osi može se vidjeti u tablici 17. Svako uklanjanje sloja integracije poboljšava performanse za red veličine. Vidimo da je ova metoda nešto sporija od one za međuindukciju u slučaju zajecničke osi, no valja imati na umu da ima jedan sloj integracije više. Korištenje višedretvenosti je djelotvorno, ali samo pri manjem broju dretvi.

Tablica 17: Brzine izvođenja za Amperovu silu u slučaju zajedničke osi s obzirom na faktor preciznosti na više računala i operacijskih sustava, slovo označava korišteno računalo, a broj uz T broj korištenih dretvi, sva vremena su u milisekundama

faktor preciznosti	inkrementi	A Windows		A Ubuntu		B Windows		C Windows	
		1T	6T	1T	6T	1T	8T	1T	8T
1.0	10 000	0.028	0.015	0.022	0.014	0.044	0.032	0.053	0.039
2.0	20 000	0.055	0.020	0.043	0.018	0.094	0.040	0.105	0.054
3.0	40 000	0.115	0.031	0.093	0.026	0.213	0.059	0.222	0.096
4.0	80 000	0.263	0.060	0.220	0.051	0.500	0.096	0.521	0.179
5.0	160 000	0.559	0.110	0.512	0.104	1.159	0.189	1.119	0.384
6.0	320 000	1.332	0.248	1.240	0.395	2.800	0.404	2.676	0.748
7.0	640 000	3.496	0.652	3.210	0.632	6.923	0.957	7.143	1.988
8.0	1 280 000	8.186	1.557	7.640	1.353	16.58	2.231	16.48	4.302

5.4.2 Općeniti slučaj

Sada ćemo proći kroz generalni slučaj dviju petlji u proizvoljnim pozicijama. Ovaj slučaj nije pogodan za naš pristup, kao što je prije spomenuto, te predviđamo probleme zbog odabrane metode integracije kada su petlje vrlo blizu jedna drugoj - najmanja udaljenost između petlji je znatno manja od njihovih radijusa. Testovi su preuzeti iz [14], a od 10 zasebnih primjera prikazanih u radu, prikazano je 7 koji pokrivaju općeniti slučaj. U radu [14] se pretpostavlja da jedna petlja leži u ravnini xOy a druga u ravnini zadanoj jednadžbom $Ax + By + Cz + D = 0$. Kako mi ne koristimo ovakvu definiciju orijentacije i položaja, pretvorit ćemo ju u pripadne kutove preko formula $\tan\theta_2 = \sqrt{A^2 + B^2}/C$ i $\tan\vartheta_2 = B/A$. Budući da je izvođenje u slučaju dviju petlji iznimno brzo, odabran je faktor preciznosti 8.0. Preračunate vrijednosti su dane u tablici 18. Tablica 19 pokazuje izvrsno slaganje rezultata uz zanemarive relativne pogreške.

Važno je napomenuti da metoda prikazana u ovome radu demonstrira vrlo impresivnu preciznost u slučajevima u kojima naš pristup jednostavno ne konvergira. U radu [14] prikazano je još testova koje ovdje ne razmatramo, no valja napomenuti da u testu u kojemu su dvije petlje međusobno razmaknute za 10^{-18} m (tisućinu debljine protona), naš pristup ne konvergira bez obzira koji faktor preciznosti odabrali. Radi se o ograničenju metode koje je u ovom slučaju zorno prikazano. Općenito, taj test nas je potaknuo da formuliramo koliko zavojnice moraju biti razmaknute da bi naša metoda brzo konvergirala. Utvrđeno je da je najmanji prihvatljivi razmak jednak približno desetini radijusa petlje. Kod pravokutnih zavojnica ovoga ograničenja nema dokle god jedna zavojnica nije unutar druge, a vrijedi jednako za međuindukciju, silu i zakretni moment, u slučaju z-osi ili općenito. Preciznost je pritom uvijek bolja od 10^{-6}. Najlošija preciznost dobila se u slučaju dvije vrlo tanke zavojnice (debljina 1000 puta veća od duljine) koje se dodiruju i nisu na istoj osi (pogreška reda 10^{-5}). Valja napomenuti da se konvergencija ne postiže u još dva rubna slučaja: dvije tanke zavojnice jedna unutar druge od kojih jedna ima neznatno manji radijus od druge i dvije plosnate zavojnice kojima je razmak zanemariv. Iz tablice 19 i prethodnih testova jasno se vidi da čim su zavojnice dostatno razmaknute, metode pokazuju vrlo brzu konvergenciju i postižu punu preciznost od 15 značajnih znamenaka. Navedeni nedostatci pristupa su neizbježni i potječu od znatnog korištenja numeričke integracije. Vrlo visoka preciznost u rubnim slučajevima je zamijenjena za veliku brzinu izvođenja. Prikladno je napomenuti da u rubnim slučajevima ne poštujemo pretpostavke modela definirane u uvodu 1 te da bi vjerojatno nesavršenost zavojnica i nezanemarive dimenzije navoja rezultirale znatno većim pogreškama.

Tablica 20 prikazuje brzine izvođenja za općeniti slučaj za silu i veće su od brzina za međuindukciju. Izračuni dobre preciznosti mogu se dobiti u nekoliko milisekundi. Skaliranje performansi je vrlo dobro.

Tablica 18: Općeniti slučaj za Amperovu silu između dvije petlje, podatci iz [14] su pretvoreni u mjere i koordinate korištene u našemu pristupu, debljina i duljina obje zavojnice je 0, x_1, y_1, i z_1 su 0, primarna zavojnica ima sve kutove rotacije jednake 0)

slučaj	R_1 (m)	I_1(A)	R_2 (m)	I_2 (A)	x_2 (m)	y_2 (m)	z_2 (m)	$\tan\theta_2$ ()	$\tan\vartheta_2$ ()
1	0.2	1	0.1	1	0.1	0.1	0.1	$\sqrt{2}$	1.0
2	0.4	1	0.05	1	0.1	0.15	0.0	$\sqrt{12}$	2/3
3	0.9	1	0.6	1	0.3	0.2	0.5	$\sqrt{2}$	1.0
4	0.005	1	0.001	1	0.003	0.001	0.0005	$\sqrt{2.5}$	1/3
5	0.3	1	0.3	-1	0.1	-0.3	0.2	$\sqrt{5}$	-2.0
6	1.0	1	0.5	1	1.0	2.0	3.0	∞	0.0
7	1.0	1	0.5	1	2.0	2.0	2.0	∞	∞

Tablica 19: Rezultati pristupa na podacima iz tablice 18, faktor preciznosti 8.0

slučaj		F_{12} iz [14]	F_{12} ovog pristupa	apsolutna relativna pogreška
1	x	-0.1080729656128444 µN	-0.1080729656128443 µN	9.25E-16
	y	-0.1080729656128444 µN	-0.1080729656128432 µN	1.11E-14
	z	-1.407372060313650 µN	-1.407372060313623 µN	1.92E-14
2	x	4.171776672650815 nN	4.171776672650817 nN	4.79E-16
	y	6.523855691357912 nN	6.523855691357856 nN	8.58E-15
	z	27.71549975211961 nN	27.71549975211960 nN	3.61E-16
3	x	0.5228604018646984 µN	0.5228604018646962 µN	4.21E-15
	y	0.4983356050923922 µN	0.4983356050923907 µN	3.01E-15
	z	-0.6364927281992902 µN	-0.6364927281992890 µN	1.89E-15
4	x	0.1370009982312461 µN	0.1370009982312457 µN	2.92E-15
	y	0.04566699941041536 µN	0.04566699941041518 µN	3.94E-15
	z	0.09856738399856347 µN	0.09856738399856331 µN	1.62E-15
5	x	0.2292455704933025 µN	0.2292455704933022 µN	1.31E-15
	y	-0.5621415690326643 µN	-0.5621415690326637 µN	1.07E-15
	z	-0.09249247340323912 µN	-0.09249247340323904 µN	8.65E-16
6	x	1.939241379554508 nN	1.939241379554505 nN	1.55E-15
	y	-1.861181718234281 nN	-1.861181718234280 nN	5.37E-16
	z	-2.202382194552672 nN	-2.20238219455267 nN	9.08E-16
7	x	-4.901398177052367 nN	-4.901398177052338 nN	5.92E-15
	y	-1.984872313200162 nN	-1.984872313200134 nN	1.41E-14
	z	-2.582265710169297 nN	-2.582265710169337 nN	1.55E-14

Tablica 20: Brzine izvođenja za općeniti slučaj sile i zakretnog momenta s obzirom na faktor preciznosti prikazan na više računala i operacijskih sustava, slovo označava korišteno računalo, a broj uz T broj korištenih dretvi, sva vremena su u milisekundama

faktor preciznosti	inkrementi	A Windows		A Ubuntu		B Windows		C Windows	
		1T	12T	1T	12T	1T	12T	1T	8T
1.0	100 000	0.433	0.102	0.337	0.093	0.753	0.178	0.794	0.300
2.0	200 000	0.875	0.167	0.630	0.148	1.439	0.276	1.594	0.560
3.0	400 000	1.446	0.266	1.029	0.225	2.500	0.415	2.644	0.914
4.0	800 000	2.442	0.473	1.992	0.407	4.526	0.641	4.604	1.431
5.0	1 600 000	5.122	0.871	3.950	0.759	9.371	1.163	9.897	2.950
6.0	3 200 000	8.707	1.515	6.844	1.283	16.23	1.894	16.84	5.051
7.0	6 400 000	17.32	3.060	14.10	2.561	32.77	3.599	34.49	9.354
8.0	12 800 000	29.49	4.896	26.58	4.698	58.61	6.620	57.09	15.38

Pronalaženje testnih slučajeva za potpuno općeniti slučaj bilo je teško, ali uspjeli smo ih pronaći u radovima [13] i [15]. Potonji je zanimljiviji jer sadrži slučajeve i za silu i za zakretni moment. Radi se o tri supravodljive zavojnice za koje se utvrđuju sile i zakretni momenti u slučaju malih pomaka po zajedničkoj osi, malih pomaka između zajedničkih paralelnih osi i malih promjena kuta između zajedničkih osi. Koristit ćemo faktor preciznosti 8.0 jer preciznost izračuna u [14] nije iznimno velika s očekivanom razlikom reda 0.01% i više. Rezultate ćemo dati na 15 značajnih znamenaka jer je to najveća teoretska preciznost za ovaj pristup. Različiti testni slučajevi iz rada stavljeni su u jednu tablicu. Neće se prikazati sve kombinacije jer bi tablica bila prevelika. Dovoljno mali brojevi (manji od 10^{-10}) zaokruženi su na 0. Svi smjerovi sila i zakretnih momenata dani su onako kako se vide *izvan* koordinatnog sustava.

Tablica 21: Tri slučaja supravodljivih zavojnica preuzeta iz [15] i pretvorena u mjere i koordinate korištene za ovaj pristup, prve dvije zavojnice su nepomične - treća se pomiče, dimenzije su u metrima

zavojnica	R (m)	a (m)	b (m)	N ()	I (A)	x	y	z	θ	ϑ
1	0.168	0.0285	0.552	1890	725	0.0	0.0	0.0	0.0	0.0
2	0.1965	0.04365	0.552	3792	725	0.0	0.0	0.0	0.0	0.0
3	0.0602	0.0728	0.5292	126	16500	varijabilno				

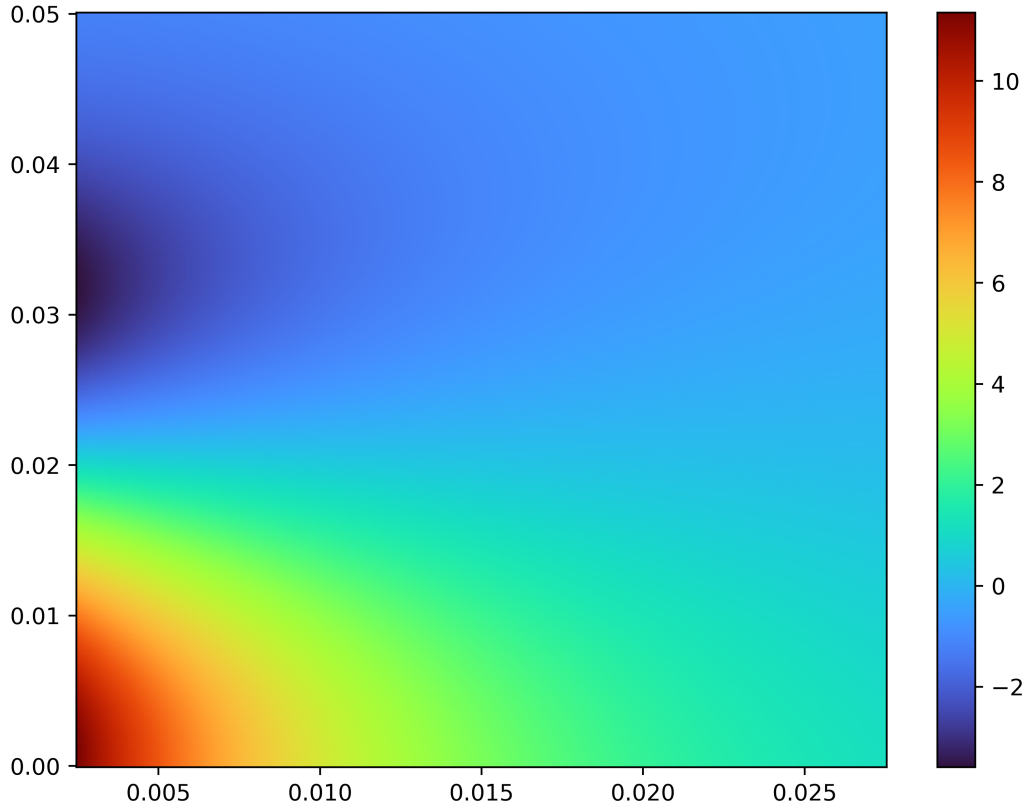
Iz tablice 22 možemo zaključiti da se svi rezultati slažu. Mala vrijednost 5.90E-10 N može se interpretirati kao apsolutna pogreška i time ukazuje da je i relativna pogreška malena. Pri računanju pogrešaka, vrijednosti su zaokružene na jednak broj značajnih znamenaka. Pogreška većinom iznosi približno 0.13% što je znatno veće od 0.01% koliko tvrdi autor rada [15]. Autor rada je objasnio da je odabrao zadanu preciznost zato što proizvodi greške reda 0.01%, no kao što se vidi iz ove usporedbe, pogreška njegovog pristupa je bila podcijenjena. Ćelije označene s (-) nisu prisutne u [15], no ovdje su dodane zbog potpunosti. Zaključujemo da su svi izrazi precizni unutar ograničenja modela.

Tablica 22: Rezultati našeg pristupa na podacima iz tablice 21, prikazana je sila koju zavojnica 1 i zavojnica 2 stvaraju na zavojnici 3, pozicija zavojnice 3 se može mijenjati, faktor preciznosti je 8.0, pomaci su dani u milimetrima, kutovi u stupnjevima, neki predznaci su promijenjeni radi konzistentnosti

Parametri slučaja		Rezultat iz [15]	Rezultat ove metode	apsolutna relativna pogreška
1:	F_x (N)	0	-7.24E-10	0.0
$x_3 = 0$	F_y (N)	6012	6004.41942497219	1.33E-03
$y_3 = 2$	F_z (N)	0	0.0	0.0
$z_3 = 0$	T_x (Nm)	-	0.0	-
$\theta_3 = 0°$	T_y (Nm)	-	0.0	-
$\vartheta_3 = 0°$	T_z (Nm)	-	0.0	-
2:	F_x (N)	0	-7.50E-10	0.0
$x_3 = 0$	F_y (N)	6011	6004.01409121734	1.16E-03
$y_3 = 2$	F_z (N)	-6012	-6004.68954391638	1.16E-03
$z_3 = 1$	T_x (Nm)	-	-4.78210752786357	-
$\theta_3 = 0°$	T_y (Nm)	-	0.0	-
$\vartheta_3 = 0°$	T_z (Nm)	-	0.0	-
3:	F_x (N)	0	-5.78E-10	0.0
$x_3 = 0$	F_y (N)	6008	6000.77380059494	1.17E-03
$y_3 = 2$	F_z (N)	18033	-18010.8256986322	1.22E-03
$z_3 = 3$	T_x (Nm)	-	-14.3279132416142	-
$\theta_3 = 0°$	T_y (Nm)	-	0.0	-
$\vartheta_3 = 0°$	T_z (Nm)	-	0.0	-
4:	F_x (N)	0	-7.56E-10	0.0
$x_3 = 0$	F_y (N)	12023	12008.0252060256	1.25E-03
$y_3 = 4$	F_z (N)	-12028	-12013.4329524184	1.25E-03
$z_3 = 2$	T_x (Nm)	-	-19.1468483177684	-
$\theta_3 = 0°$	T_y (Nm)	-	0.0	-
$\vartheta_3 = 0°$	T_z (Nm)	-	0.0	-
5:	F_x (N)	0	-5.58E-10	0.0
$x_3 = 0$	F_y (N)	12013	11998.2980105876	1.25E-03
$y_3 = 4$	F_z (N)	-24050	-24020.364174005	1.25E-03
$z_3 = 4$	T_x (Nm)	-	-38.2197734078408	-
$\theta_3 = 0°$	T_y (Nm)	-	0.0	-
$\vartheta_3 = 0°$	T_z (Nm)	-	0.0	-
6:	F_x (N)	3006.0	3002.28321703283	1.23E-03
$x_3 = 1$	F_y (N)	-43.00	-41.9145257595907	0.0253
$y_3 = 0$	F_z (N)	-6013.0	-6006.52090590201	1.08E-03
$z_3 = 1$	T_x (Nm)	-2138.2	-2137.42208461393	3.74E-04
$\theta_3 = 1°$	T_y (Nm)	2.4630	2.40128157271086	0.0251
$\vartheta_3 = 270°$	T_z (Nm)	0.0430	0.0419145257644353	2.56E-03
7:	F_x (N)	3006.1	3002.32375130920	1.26E-03
$x_3 = 1$	F_y (N)	2964.6	2961.98718511205	8.77E-04
$y_3 = 1$	F_z (N)	-6057.1	-6048.62973912744	1.40E-03
$z_3 = 1$	T_x (Nm)	-2138.4	-2139.73178025414	6.08E-04
$\theta_3 = 1°$	T_y (Nm)	2.3755	2.31088032980234	0.0272
$\vartheta_3 = 270°$	T_z (Nm)	0.0414	0.0403365661995316	0.0266
8:	F_x (N)	-	5.90E-10	-
$x_3 = 0$	F_y (N)	-	0.0	-
$y_3 = 0$	F_z (N)	-	1.24E-09	-
$z_3 = 0$	T_x (Nm)	-11020.3	-11005.9323137128	1.31E-03
$\theta_3 = 5°$	T_y (Nm)	-	0.0	-
$\vartheta_3 = 270°$	T_z (Nm)	-	0.0	-

5.5 Dodatne mogućnosti

Na kraju ovog poglavlja prikladno je još kratko komentirati neke testove koji su provedeni a nisu eksplicitno prikazani. Performanse MTD pristupa nisu detaljno diskutirane no dostatno je reći da je ponašanje jednako kao kod metoda za izračun polja kod velikog broja točaka. Posebno veliko unaprjeđenje vidi se kod metoda za izračun velikog broja konfiguracija zavojnica. Na slici 22 prikazan je iznos međuindukcije za različite relativne položaje dviju zavojnica. Dimenzije zavojnica su preuzete iz [11], rezolucija slike je 300x300, dakle 90 000 konfiguracija, a vrijeme izračuna iznosi svega 6 minuta na računalu B. U slučaju da su zavojnice znatno udaljene moguće je izračunati vektor magnetskog momenta sekundarne zavojnice te s preciznošću reda 0.1 % odrediti silu i zakreni moment. Nismo još prikazali slučajeve korištenja razreda `CoilGroup` koji omogućava brze izračune kod korištenja velikog broja zavojnica - recimo prikaz magnetskog polja toroidne zavojnice koja je modelirana kao skup petlji. Naprednija hardverska akceleracija metoda ove klase je zanimljiv predmet budućeg rada.



Slika 22: Prikaz vrijednosti međuindukcije (μH) dviju zavojnica dimenzija $R_1 = R_2 = 0.01022$ m, $a_1 = a_2 = 0.011$ m, $b_1 = b_2 = 0.0022$, a broj navoja $N_1 = N_2 = 20$, prva zavojnica je u ishodištu, a drugoj je $y_2 = 0$ dok x-os određuje z_2, a y-os određuje x_2 u vektoru položaja, svi kutovi rotacije su 0

6 Zaključak i daljnji rad

U ovome radu je prikazan novi pristup izračuna vektorskog potencijala, magnetskog polja i gradijenta magnetskog polja u slučaju kružne zavojnice te primjena navedenog pristupa na izračun interakcije parova kružnih zavojnica u obliku međuindukcije, sile i zakretnog momenta. Za razliku od ostalih metoda pronađenih u znanstvenoj literaturi, naš pristup temelji se na korištenju što jednostavnijih izraza s ciljem poboljšanja performansi i implementacije pristupa u kontekstu višedretvenosti i hardverske akceleracije. Pristup je osmišljen u prvome redu za kružne zavojnice pravokutnoga presjeka, dok su ostala 3 slučaja, petlja, tanka zavojnica i plosnata zavojnica, uključena zbog dodatne pogodnosti kod korištenja i lakše usporedbe pristupa s trenutnom znanstvenom literaturom. Za navedena 3 slučaja postoje brojne metode razvijene specifično za njih koje pokazuju znatno bolja svojstva u rubnim slučajevima poput malog razmaka. Naš pristup je implementiran u programskom jeziku C++ uz Python modul koji je omotač oko C++ koda. Ovo je još jedna posebnost našeg pristupa jer u ostalim radovima ili implementacijski detalji nisu dostupni ili je metoda implementirana u višim programskim jezicima poput platforme MatLab ili sustava Wolfram Mathematica.

Kako je cilj izbjeći korištenje velikog broja kompliciranih funkcija, trostruki integrali koji se pojavljuju pri raspisu izraza za polja pravokutne zavojnice kružnog presjeka ne mogu se znatno pojednostaviti pa se pristup ekstenzivno oslanja ne Gauss-Legendre metodu numeričke integracije. Reduciran je jedan sloj integracije po duljini zavojnice što dovodi do podjele metoda za izračun polja na brze metode, primjenjive na tanku i pravokutnu zavojnicu, te spore metode, primjenjive na plosnatu zavojnicu i petlju. Performanse ovih metoda su opsežno testirane te je utvrđeno da su implementirane na vrlo efikasan način koji može potpuno opteretiti moderni višejezgreni procesor. Za ostvarenje visokih performansi pristup se primarno pouzdaje u AVX2 instrukcije, optimizacije C++ prevoditelja te korištenje višedretvenosti. Višedretvenost može poboljšati performanse proporcionalno broju korištenih jezgara kada je opterećenje dovoljno veliko. Moguće je izračunati milijune vrijednosti polja svake sekunde. Pristup se također pokazao pogodnim za hardversku akceleraciju te je prikazano da već osnovna implementacija može poboljšati performanse za jedan red veličine u usporedbi s potpuno opterećenim procesorom. Pritom je važno imati na umu da je preciznost smanjena. Testiranje je provedeno na 5 računala različitih osobina koja efektivno pokrivaju performanse procesora od 2013. godine do danas. Također su detaljno uspoređivane performanse operacijskih sustava Windows 10 i Ubuntu 22.04 te je utvrđeno da Ubuntu pokazuje bolje i konzistentnije performanse.

Metode za izračun vektorskog potencijala i magnetskog polja su zatim iskorištene za direktan izračun međuindukcije, sile i zakretnog momenta. Razlikuje se slučaj zajedničke osi i općeniti slučaj. Kako je u općenitom slučaju potrebno provesti numeričku integraciju nad ukupno 5 slojeva, razvijen je algoritam za efikasnu raspodjelu fiksnog broja inkremenata između tih 5 slojeva. Za izračun međuindukcije

u slučaju zajedničke osi, napravljena je posebna metoda koja se koristi u slučaju tanke zavojnice i pravokutne zavojnice te osnovna metoda za slučaj tanke zavojnice i petlje. Performanse su otprilike dva reda veličina bolje od prijašnjih metoda uz preciznost od 15 značajnih znamenaka, osim u slučajevima kada su dvije zavojnice jako blizu. Poseban slučaj je samoindukcija i za nju je također implementirana posebna metoda. Konvergencija samoindukcije je dosta spora i ima slabiju preciznost reda 10^{-6} pa se ne preporuča upotreba ako je potrebna velika preciznost. Općenita metoda je vrlo precizna ako se zavojnice ne dodiruju te može ostvariti punih 15 značajnih znamenki preciznosti u manje od 10 milisekundi, dok je najsličnijoj pronađenoj metodi potrebno nekoliko minuta. Sila i magnetski moment rijetko se analiziraju u literaturi i pronađen je samo jedan rad koji se bavi općenitim slučajem za dvije kružne zavojnice pravokutnog presjeka, ali s izračunima niske preciznosti. Preciznost u slučaju zajedničke osi i petlji u proizvoljnom položaju opet iznosi 15 značajnih znamenki. Vrijeme izračuna sile i zakretnog momenta nešto je kraće od izračuna međuindukcije. Raspis gradijenta magnetskog polja nije pronađen u literaturi, a ovdje je tretiran jednako kao magnetsko polje i potencijal. Može se koristiti kao aproksimacija sile na udaljenu ili malenu zavojnicu te za izračun sile na česticu s dipolnim momentom.

Pristup opisan u ovom radu pokazao se iznimno brzim i preciznim u većini slučajeva. Performanse metoda za izračun polja su dostatne za brz izračun vrijednosti polja u cijelom 3D prostoru, pogotovo ako se koristi grafička akceleracija. Velika brzina ovih metoda također omogućava brz izračun interakcije zavojnica u slučaju varijacije parametara, poput prikaza vrijednosti međuindukcije kod različitih prostornih konfiguracija. Sve opisane metode rade i na sustavima od više zavojnica te su implementirane u sveobuhvatnom okviru koji se može koristiti iz programskog jezika više razine, poput Pythona.

Predstavljeni pristup pokazuje moć korištenja hardvera kojeg imamo na raspolaganju u moderno vrijeme, a pogotovo je zanimljivo iskorištenje potencijala grafičke kartice. U budućem radu ćemo se baviti efikasnijim korištenjem grafičke kartice te primjenom opisane paradigme na druge probleme iz ovoga područja. Također je važno detaljnije promotriti izraze polja u slučaju da je moguće reducirati još jedan sloj integracije uz implementaciju neke posebne funkcije. U planu je također proširenje programske potpore na platformu MatLab kako bi metoda bila pristupna još većem broju inžinjera.

7 Sažetak - Abstract

Hrvatska inačica - sažetak:

Općeniti pristup za paralelizirani izračun polja kružnih zavojnica pravokutnog presjeka

Davor Dobrota, Nikola Sočec, Lara Vrabac

Većina literature u području istraživanja kružnih zavojnica temelji se na smanjenju broja slojeva integracije te korištenju programskih jezika visoke razine poput programske platforme MatLab i Wolfram Mathematica sustava. Stoga je cilj ovoga rada prikazati alternativni pristup izračunu vektorskog potencijala, magnetskog polja i gradijenta magnetskog polja u slučaju kružnih zavojnica. Pristup je temeljen na Gauss-Legendre kvadraturi za numeričku integraciju, a naglasak je na efikasnoj implementaciji metoda za izračun polja s ciljem ostvarenja visokih performansi. Pristup je pogodan za paralelizaciju i hardversku akceleraciju. Korištenje paralelizacije omogućava izračun više od milijun vrijednosti polja svake sekunde, a u slučaju korištenja hardverske akceleracije izračunava se i preko dvadeset milijuna vrijednosti po sekundi. Navedeno omogućava brz izračun međuindukcije, sile i zakretnog momenta između dvije ili više kružnih zavojnica u proizvoljnom položaju i orijentaciji. Za efikasno korištenje računalnih resursa osmišljen je algoritam za automatsku optimizaciju preciznosti numeričke integracije. Rezultat je brz, precizan i općenit pristup koji se odlično slaže s rezultatima drugih objavljenih radova uz dva do tri reda veličine kraće vrijeme izvođenja. Puna preciznost od petnaest značajnih znamenaka ostvariva je u većini slučajeva. Pristup je implementiran u programskom jeziku C++ uz pripadni omotač koji omogućava korištenje iz programskog jezika Python.

Ključne riječi:
kružna zavojnica, magnetsko polje, međuindukcija, paralelizacija, hardverska akceleracija

English version - abstract:

General parallelized field calculation approach for circular coils with rectangular cross section

Davor Dobrota, Nikola Sočec, Lara Vrabac

Relevant literature on circular coils is primarily focused on decreasing the number of layers of integration, and using high-level programming languages on platforms such as MatLab and Wolfram Mathematica. The goal of this paper is to present an alternative to existing approaches for calculating the magnetic vector potential, magnetic field, and magnetic field gradient of circular coils. The approach is based on the Gauss-Legendre quadrature for numerical integration with an emphasis on efficient implementation and, consequently, high performance. It is suitable for parallelization and hardware acceleration. Employing parallelization leads to more than a million field calculations each second, while hardware acceleration increases this number to over twenty million. This enables fast calculations of mutual inductance, force, and torque between two circular coils in an arbitrary arrangement. An increment-balancing algorithm has been added to efficiently utilize available computational resources. The result is a performant and accurate general approach which is in excellent agreement with previously published methods. Peak precision of fifteen significant digits is attained in most cases. The method is implemented in the C++ programming language with a dedicated interface to the Python programming language.

Keywords:
circular coil, magnetic field, mutual inductance, parallelization, hardware acceleration

8 Literatura

[1] S. Babic, C. Akyel, “Improvement in calculation of the self- and mutual inductance of thin-wall solenoids and disk coils”, IEEE Trans. Magn., vol. 36, no. 4, pp. 1970-1975, July 2000.

[2] S. Babic, C. Akyel, S. J. Salon, “New procedures for calculating the mutual inductance of the system: filamentary circular coil-massive circular solenoid”, IEEE Trans. Magn., vol. 30, no. 3, pp. 1131-1134, May 2003.

[3] S. I. Babic, C. Akyel, “Calculating mutual inductance between circular coils with inclined axes in air”, IEEE Trans. Magn., vol. 40, no. 7, pp. 1743-1750, July 2008.

[4] J. T. Conway, “Inductance calculations for circular coils of rectangular cross section and parallel axes using Bessel and Struve Functions”, IEEE Trans. Magn., vol. 36, no. 1, pp. 75-81, January 2010.

[5] S. Babic, F. Sirois, C. Akyel, G. Lemerquand, V. Lemerquand, R. Ravaud, “New formulas for mutual inductance and axial magnetic force between a thin wall solenoid and a thick circular coil of rectangular cross-section”, IEEE Trans. Magn., vol. 47, no. 8, pp. 2034-2044, August 2011.

[6] S. Babic, C. Akyel, “New formulas for mutual inductance and axial magnetic force between magnetically coupled coils: thick circular coil of the rectangular cross-section-thin disk coil (pancake)”, IEEE Trans. Magn., vol. 49, no. 2, pp. 860-868, February 2013.

[7] T. Župan, Ž. Štih, Bojan Trkulja, “Fast and precise method for inductance calculation of coaxial circular coils with rectangular cross section using the one-dimensional integration of elementary functions applicable to superconducting magnets”, IEEE Trans. Appl. Supercond., vol. 24, no. 2, seq. num. 4901309, April 2014

[8] Y. Luo, X. Wang, X. Zhou, “Inductance calculations for circular coils with rectangular cross section and parallel axes using inverse Mellin transform and generalized hypergeometric functions”, IEEE Trans. Power Electron., vol. 32, no. 2, pp. 1367-1374, February 2017.

[9] J. T. Conway, “Mutual inductance of thick coils for arbitrary relative orientation and position”, Proceedings of 2017 PIERS – FALL, Singapore, 19-22 November 2017.

[10] K. Song, J. Feng, R. Zhao, X. Wu, “A general mutual inductance formula for parallel non-coaxial circular coils”, ACES Journal, vol. 34, no. 9, pp. 1385-1390, September 2019.

[11] Y. Wang, X. Xie, Y. Zhou, W. Huan, "Calculation and modeling analysis of mutual inductance between coreless circular coils with rectangular cross section in arbitrary spatial position", Proceedings of ITOEC 2020, pp. 1258-1267, Chongqing, 12-14 June 2020

[12] S. I. Babic, C. Akyel, "Magnetic force calculation between thin coaxial circular coils in air", IEEE Trans. Magn., vol. 44, no. 4, pp. 445-452, April 2008.

[13] Y. Ren, "Magnetic force calculation between misaligned coils for a superconducting magnet", IEEE Trans. Appl. Supercond., vol. 20, no. 6, pp. 2350-2353, December 2010.

[14] S. Babic, C. Akyel, "Magnetic force between inclined circular filaments placed in any desired position", IEEE Trans. Magn., vol. 48, no. 1, January 2012.

[15] Z. J. Wang, Y. Ren, "Magnetic force and torque calculation between circular coils with nonparallel axed", IEEE Trans. Appl. Supercond., vol. 24, no. 4, seq. num. 4901505, August 2014.

[16] Feynman, Richard P., Robert B. Leighton, and Matthew L. Sands, "The Feynman Lectures on Physic", Reading, Mass: Addison-Wesley Pub. Co, 1963.

[17] Rainer Kress, „Numerical Analysis", Springer New York, 1998
https://doi.org/10.1007/978-1-4612-0599-9

[18] Nodes and Weights of Gauss-Legendre Calculator, accessed on: May 25, 2022
Available: https://keisan.casio.com/exec/system/1280624821

[19] Intel Advanced Vector Extensions Programming Reference, 2011, accessed on: May 25, 2022
https://www.intel.com/content/dam/develop/external/us/en/documents/36945

[20] Nvidia Cuda Toolkit Documentation, accessed on: May 27, 2022
https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[21] AnandTech: Hardware News and Tech Reviews Since 1997, accessed on: May 27, 2022
https://www.anandtech.com/

[22] R. Dolbeau, "Theoretical peak FLOPS per instruction set: a tutorial", J Supercomput, vol. 74, pp. 1341–1377, 2018, available: https://doi.org/10.1007/s11227-017-2177-5

[23] Wenzel Jakob, 2016, pybind11 (Version 2.9.2), accessed on: May 25, 2022
Source code: https://github.com/pybind/pybind11

[24] Github user vit-vit, 2004, accessed on: May 25, 2022
Source code: https://github.com/vit-vit/CTPL

[25] J. D. Hunter, “Matplotlib: A 2D Graphics Environment”, Computing in Science And Engineering, vol. 9, no. 3, pp. 90-95, 2007.