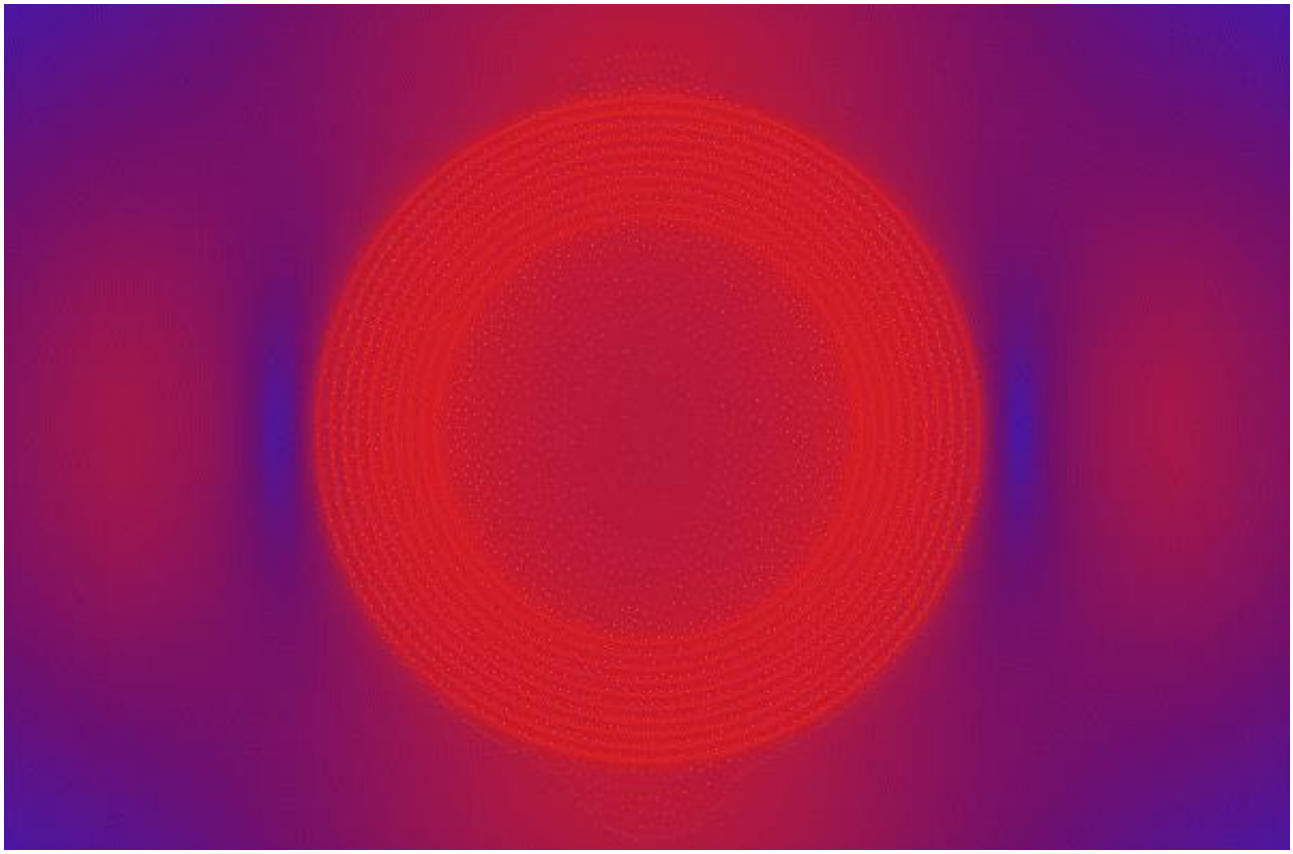
COPPER

Coil OPeration Projection with Expedited Reckoning

Autori: Nikola Sočec, Davor Dobrota

Mentor: prof. Nikola Dmitrović



Zagreb, 2019.

Sadržaj

1. Uvod

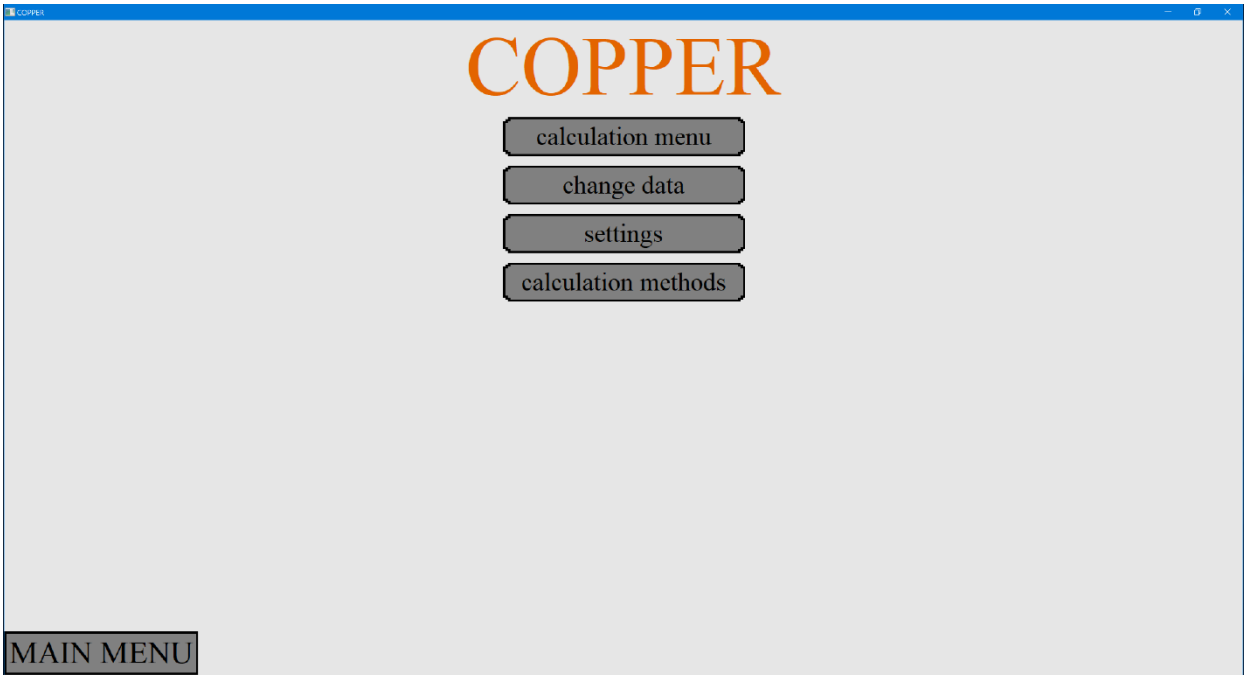
Potreba za izradu programa koji vizualizira magnetsko polje nastala je prilikom istraživanja primjene magnetskog polja zavojnice. Tražeći brojne programe po internetu, nismo uspjeli pronaći nikakav adekvatan alat koji bi nam omogućio fleksibilno prikazivanje magnetskog polja jedne ili više zavojnica. Kao rješenje toga problema nastao je COPPER (Coil OPeration Projection with Expedited Reckoning – Projekcija rada zavojnice s ubrzanim računanjem).

Iako je ovaj projekt inicijalno zamišljen samo kao jedna od podjedinica projekta iz fizike, uskoro smo shvatili da je sama ideja i njena implementacija dostojna vlastitog projekta. Dodatni razlog postojanja projekta je i složenost same funkcije koja leži o njegovoj osnovi. Radi se nedefiniranom eliptičnom integralu trećeg reda, čije rješavanje zahtijeva popriličnu količinu resursa. Pri prosječnoj preciznosti, jednoj procesorskoj jezgri i7 procesora 7. generacije potrebno je preko 2 sekunde za izračunati jednu točku. Ovakve performanse su se pokazale neprihvatljivima za crtanje cijelih slika i 3D prostora pa smo zato odlučili algoritam paralelizirati, a računanje obavljati na grafičkoj kartici. Jednako kompliciranim se pokazalo i prikazati izračunate vrijednosti polja na informativan i elegantan način. Za tu svrhu, koristili smo se visual libraryjem WHS (Windows Handling Simplified) kojeg je samostalno razvio Nikola Sočec.

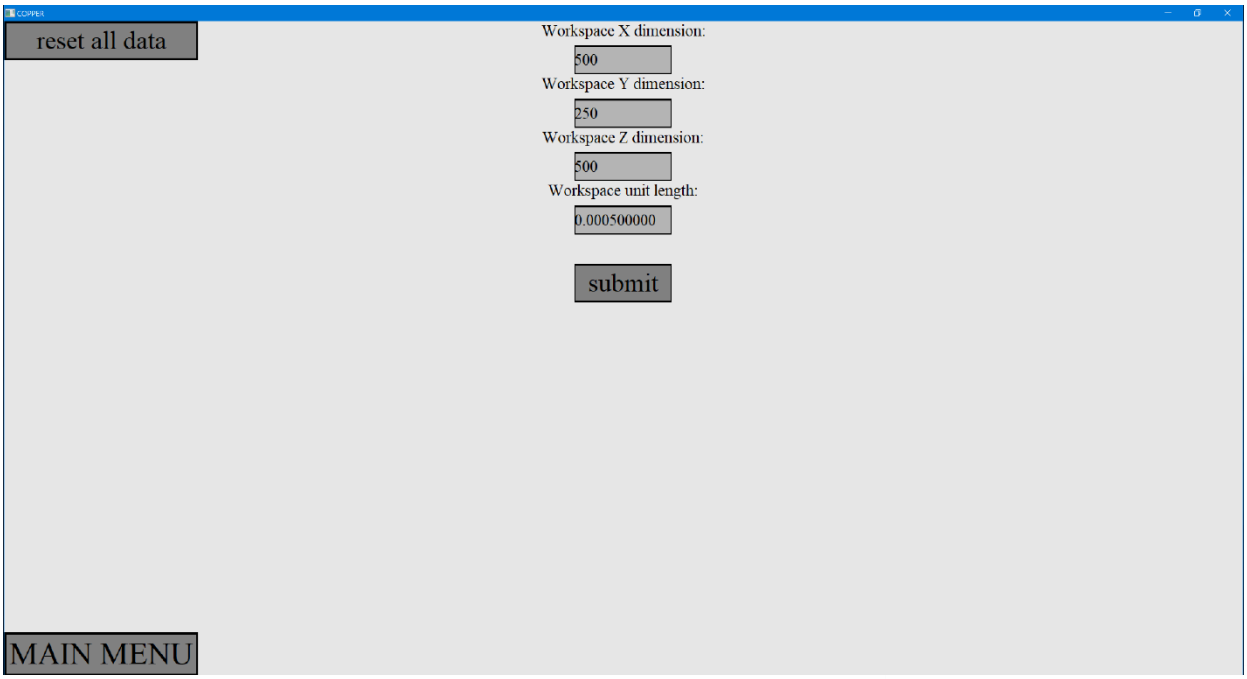
Cijeli projekt prikaza polja kružne zavojnice i njegove uporabe, kojega je COPPER dio, nastao je kao posljedica manjka dobrih informacija na internetu o samom magnetskom polju za ovakav tip zavojnice te njegovim dodatnim efektima. Davor Dobrota je originalno osmislio ideju te je „fizičar“ iza projekta. On se pretežno bavio fizikalnim dijelom projekta, tj. matematičkom razradom same funkcije, ali i razradom funkcionalnosti projekta i njegovih ciljeva. Nikola Sočec bavio se većinom programskim dijelom projekta, tj. implementacijom funkcije. Valja napomenuti da je rad slične teme s naslovom „Magnetsko polje kružne zavojnice i njegova primjena“ (istih autora) također ostvario plasman na državno natjecanje iz fizike.

2. Uporaba programa

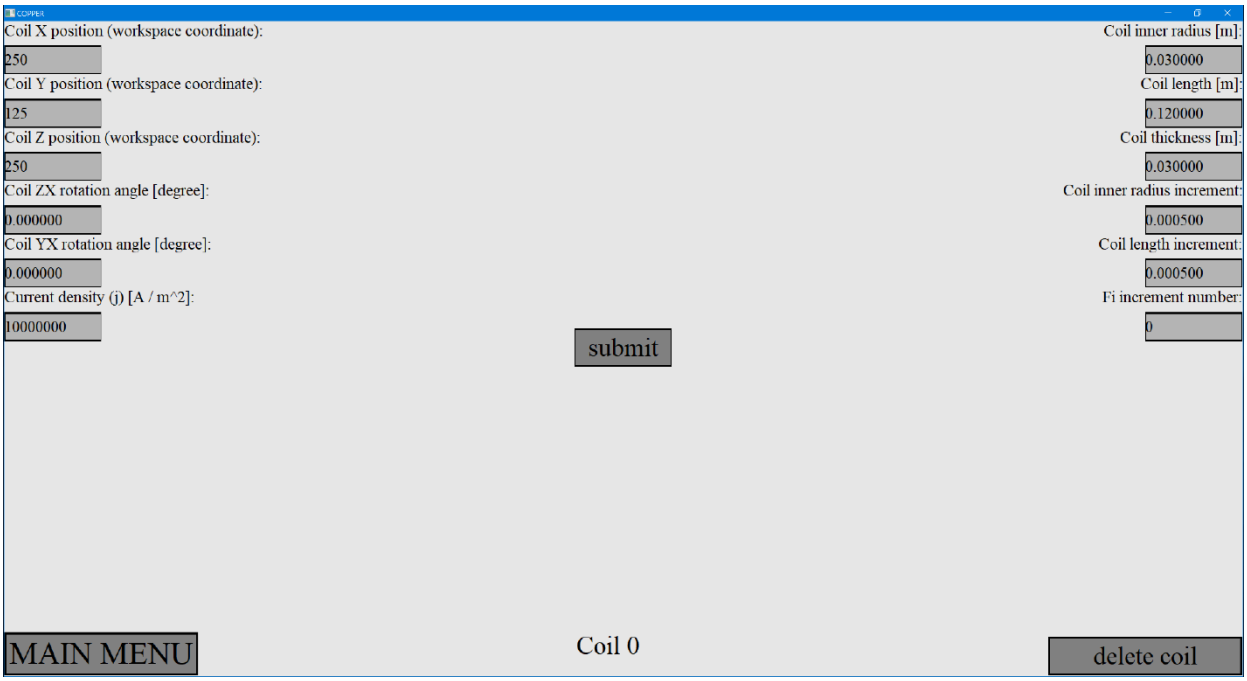
Kao što smo već naglasili u uvodu, primarna uloga ovoga programa je prikazati magnetsko polje u prostoru, njegov oblik i iznos. Za prikaz uporabe uzet je skup podataka koji će sigurno raditi, dok je također natprosječno zahtijevan. Program je podijeljen na glavni izbornik te četiri podizbornika.



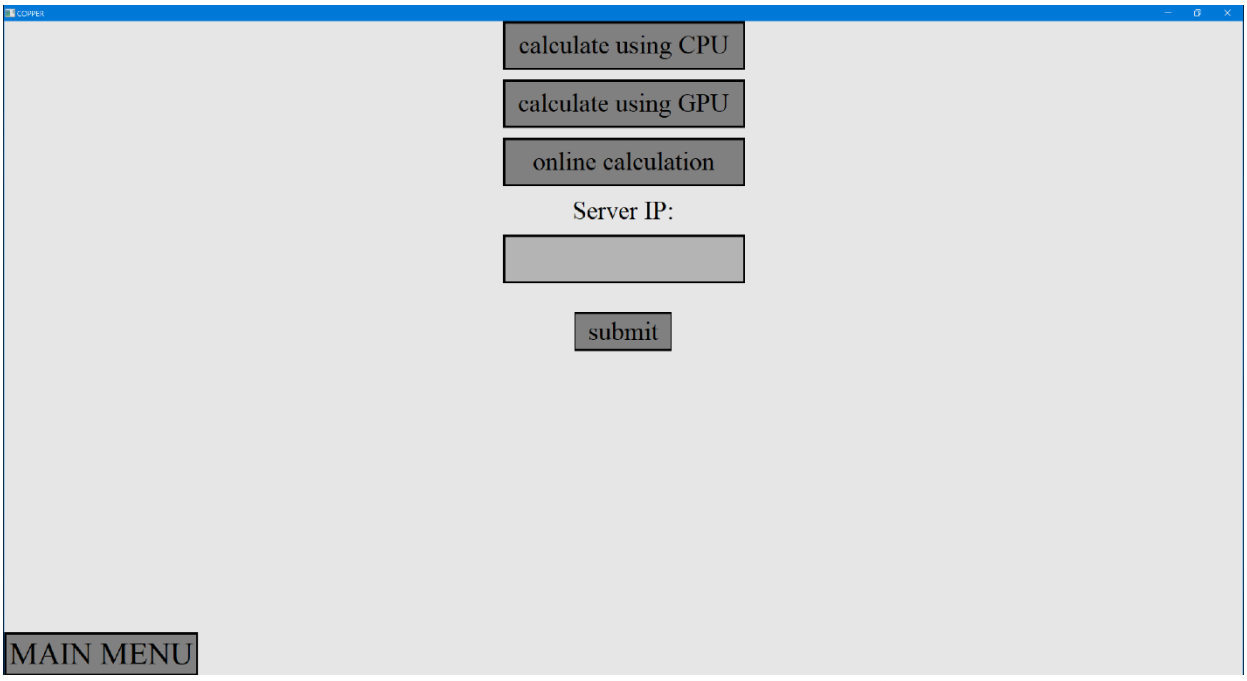
Preporučljivo je započeti od podizbornika settings gdje se nalaze osnovne postavke prostora koji se promatra. Prostor je zapravo 3D matrica te je definirana trima dimenzijama X, Y i Z. Te vrijednosti su ujedno i broj pixela te ih je nuzno unositi kao integere. Valja naglasiti da se svi podatci pohranjuju direktno u RAM te da treba paziti na količinu memorije (X*Y*Z*16 B) Četvrta bitna odrednica je činjenica da svaki taj pixel zapravo predstavlja jako mali kubični odsječak prostora, a ova varijabla definira upravo duljinu stranice kocke koja ga definira. Unos je u metrima [m], a za većinu primjena je prikladno uzimati vrijednosti u intervalu $[0.0001, 0.01]$. U gornjem lijevom rubu ekrana se nalazi gumb za resetiranje svih podataka (uključuje matricu i zavojnice).



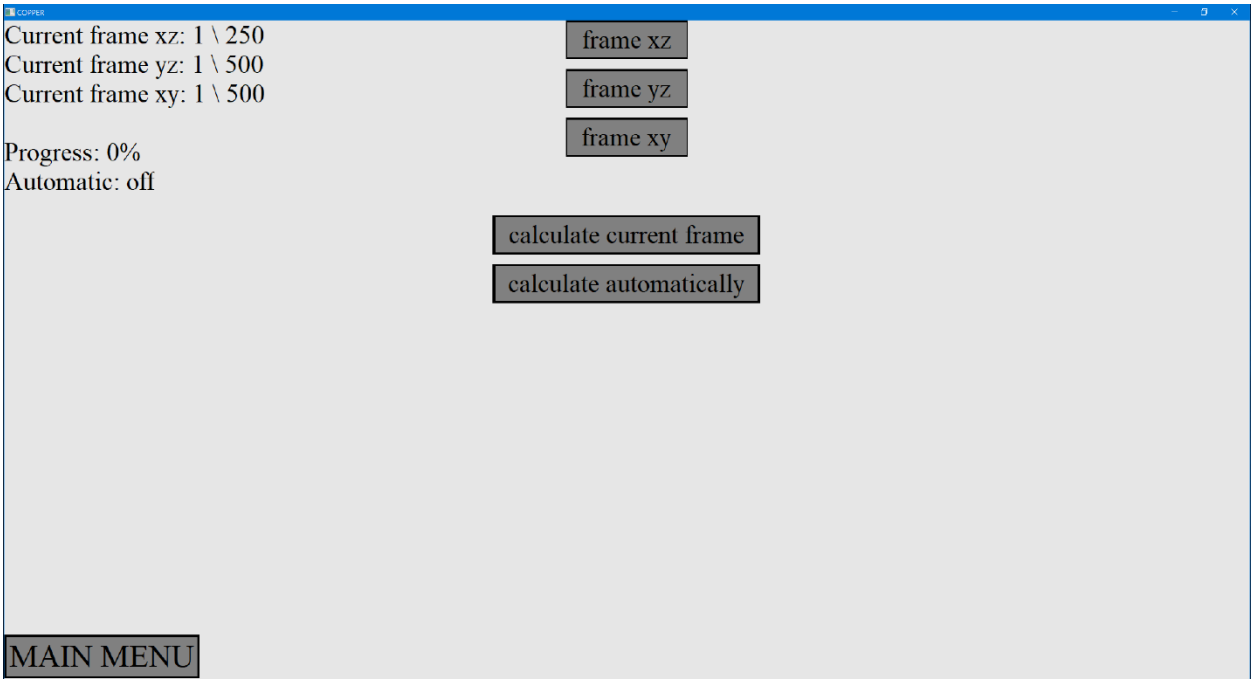
Nakon klikanja na submit možete se vratiti u main menu (vaše vrijednosti su pohranjene i biti će primjenjene u računanju). Nadalje prelazimo na podatke koje je potrebno unijeti za samu zavojnicu. Podatke unosimo prema parametrima iz 2. poglavlja. Pritiskom na gumb u samoj sredini ekrana se zavojnica mijenja ako već postoji ili se stvara nova. Pritiskom na gumb u donjem desnom kutu se trenutna zavojnica briše. Tekst na sredini dna ekrana pokazuje trenutnu zavojnicu. Strelicama na tipkovnici se mijenja trenutna zavojnica (strelica prema gore donosi sljedeću, a strelica prema dolje prethodnu zavojnicu). Zavojnice se kreću od broja 0 do broja koji odgovara ukupnom broju zavojnica. Mijenjanje te zadnje zavojnice ustvari dodaje novu zavojnicu i otvara se novo mjesto za unos.



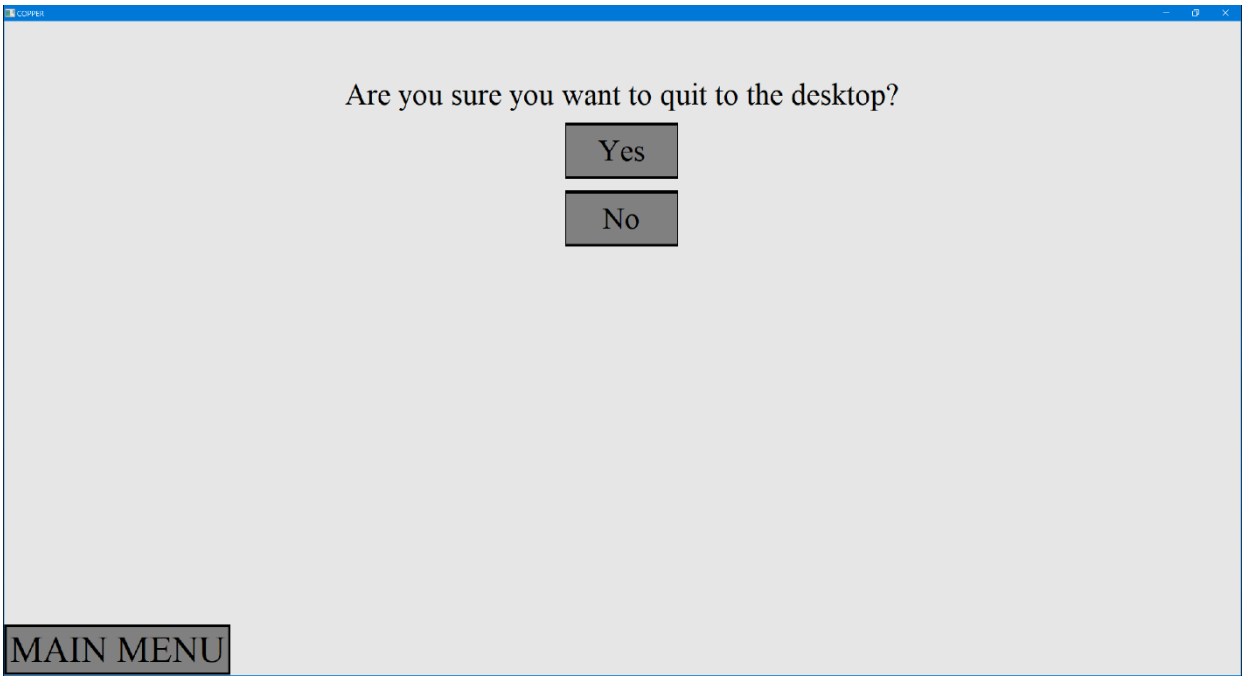
U segmentu „calculation methods“ se bira način na koji će se polje računati. Program podržava računanje pomoću procesora (bira se pomoću prvog gumba s vrha) i pomoću grafičke kartice (drugi gumb s vrha). Konačno ukoliko se koristi slabo računalo ili je kalkulacija jednostavno prezahtijevna, moguće se spojiti na server poznate IP adrese koji koristi Host verziju programa.



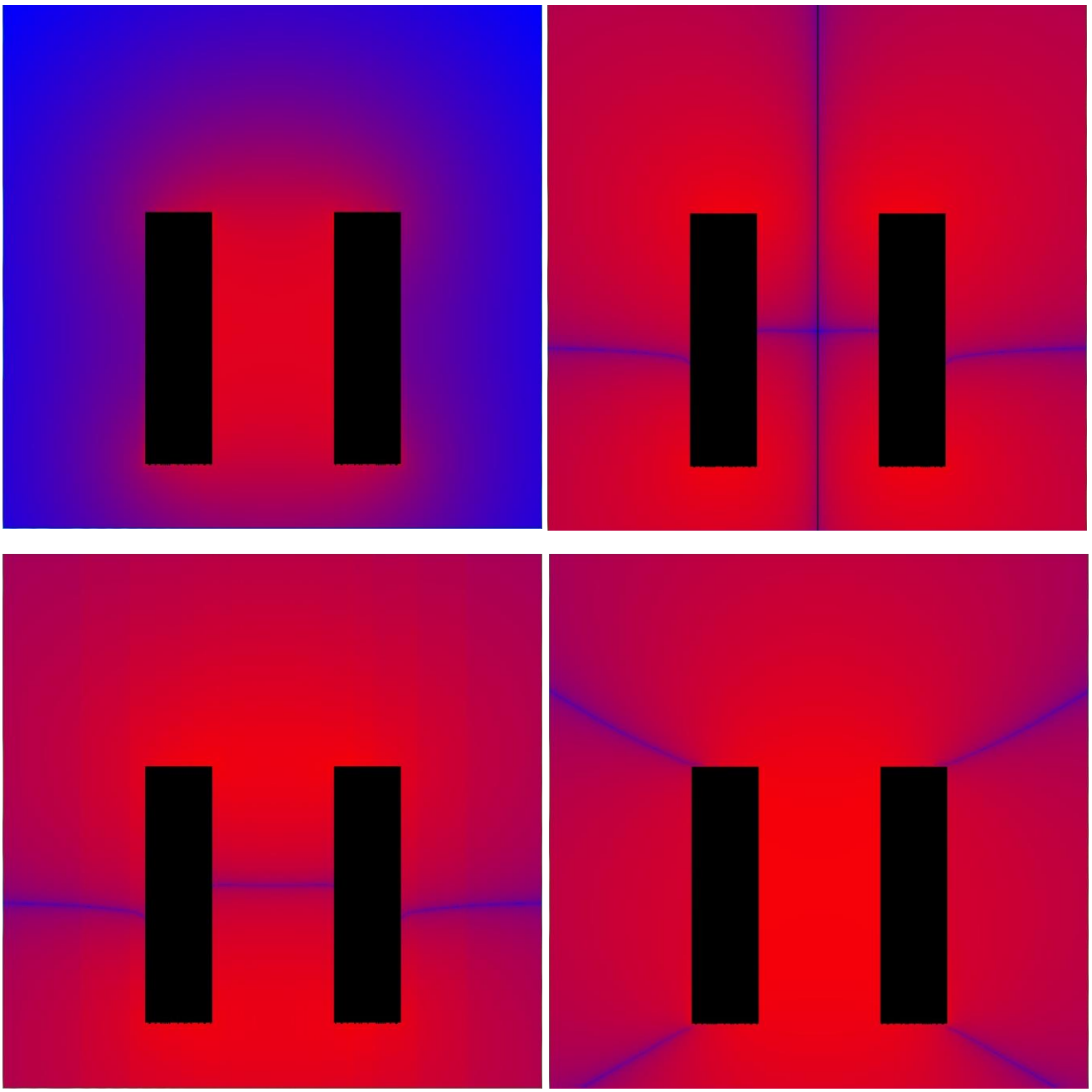
Dolazimo do samog segmenta računanja. 3 su moguće perspektive, xy, xz i xy te svaka ima određeni broj slojeva. Kliknemo na perspektivu koju želimo računati, a potom sa scroll kotačićem (±1 sloj), strelicama lijevo i desno (±1 sloj) i strelicama gore i dolje (±10 slojeva) mijenjamo sloj koji računamo. Kad ste spremni započeti kliknite calculate. Ovisno o rezoluciji i postavljenoj preciznosti očekujemo različito vrijeme računanja, ono linearno ovisi o rezoluciji, linearno o broju inkremenata po fi te je obrnuto proporcionalno inkrementima po duljini i debljini. Moguće je mijenjati veličinu svih prozora razvlačenjem, uključujući i veličinu elemenata na slici s + i -. Ako želite redom računati kliknite na calculate automatically koji funkcionira kao toggle, tj. kada ugasite možete vidjeti sve renderano i samo nastaviti naknadno. Možete pritom raditi stvari u pozadini no to može produžiti vrijeme renderanja



Ukoliko želimo izaći iz programa kliknemo Esc kada treba još jednom potvrditi da stvarno želimo izaći. Pritom se svi podatci automatski spremaju.



Kao output možemo očekivati nešto slično ovome



3. Fizikalna pozadina (Davor Dobrota)

3.1. Izvod korištene funkcije

Započnimo sa osnovom samoga programa, a to je teoretska funkcija koja leži u njegovoj srži. Izvod naveden ovdje je čisto fizikalne naravi.

Prvo je potrebno definirati za kakav tip zavojnice je naše rješenje valjano. U obzir uzimamo kružnu zavojnicu pravoktunog presjeka. Za razliku od solenoida kojemu je debljina zanemariva naspram duljine, u ovakvoj zavojnici postoji više slojeva namotaja pa ju se može promatrati kao veći broj solenoida, svaki radijusa većeg za debljinu žice d. Ovakav tip zavojnice karakteriziramo trima veličinama – njezinom debljinom a (u programu Coil Thickness), duljinom b (Coil Length) i njezinim unutarnjim raijusom (Inner Radius). Ukoliko nam je poznat broj namotaja po debljini i duljini možemo pisati

$$(1a)\ M = \frac{a}{d}$$

$$(1b)\ L = \frac{b}{d}$$

$$(1c)\ N = ML$$

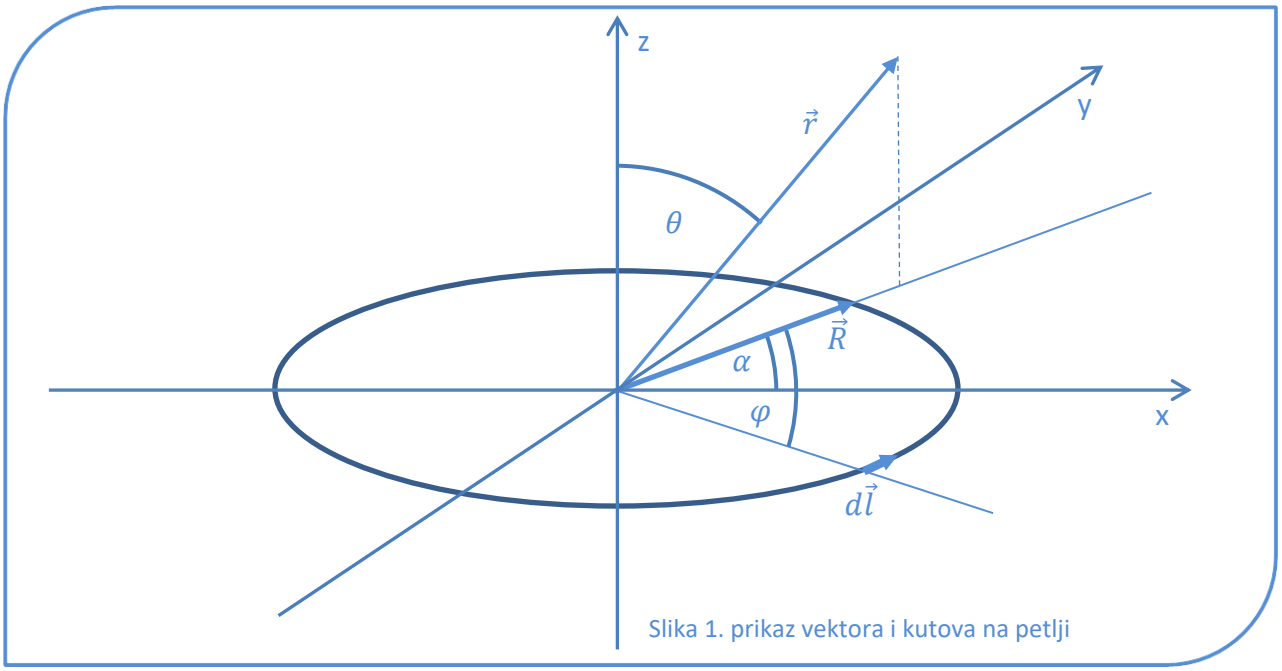
Gdje je M broj namotaja po debljini, L po duljini, a N ukupni broj namotaja. Nadalje potrebno je upoznati se s konceptom gustoće struje. Gustoća struje je struja podijeljena s površinom kroz koju prolazi. Iako je žica koja čini namotaje zapravo okrugla, vrlo je korisno uzeti da je ona zapravo pravokutna jer u tom slučaju dobivamo mnogo fleksibilnosti uz minimalno smanjenje preciznosti. U ovakvom slučaju možemo pisati

$$(1d)\ J = \frac{I}{d^2} = \frac{NI}{ab}$$

Gdje je I struja koja protječe kroz zavojnicu. Ukoliko nas zanima neki realni materijal poput bakra, onda imamo određena ograničenja vezana uz gustoću struje no u programu ne postoje ograničenja toga (osim toga da stane u double format).

Osnovni zakon koji koristimo za evaluaciju bilo kakvog magnetnog statičnog polja je Biot-Savart-ov[1] zakon. Zavojnicu možemo promatrati kao mnogo kružnih petlji s različitim radijusima i pozicijama središta. Dakle prvo je potrebno pronaći jednadžbu kojom možemo opisati magnetsko polje jedne petlje radijusa R za bilo koju točku u prostoru, a zatim jednadžbu relativno jednostavno proširimo na više petlji. S obzirom na kružni oblik petlje korisno je koristiti se polarnim koordinatnim sustavom u kojemu su koordinate definirane udaljenošću od središta te dvama kutovima. Središte petlje postavljamo u ishodište polarnog sustava. Uzimamo proizvoljnu točku u prostoru danu koordinatama

$A(r, \alpha, \beta)$. Polarne koordinate jednostavno je prevesti u kartezijanske koordinate i iz toga saznati iznos svake zasebne komponente magnetskog polja. Uvodimo kut φ koji predstavlja kut između polupravca koji izlazi iz središta petlje i prolazi kroz mali segment $d\vec{l}$ na kružnici čiji doprinos promatramo, te polupravca kojeg definira kut α sa središtem. Također uvodimo kut θ definiran okomicom kroz središte petlje (z-os kartezijanskog sustava) i vektorom $\vec{r}$ povučenim od ishodišta do točke A koju promatramo (iznosnom je $\theta = \beta - \pi/2$). Ovo se lijepo vidi na shemi prikazanoj na Slici 1.



Slika 1. prikaz vektora i kutova na petlji

Za svaku točku vrijedi formula

$$(2a)\ \ \vec{B}(r) = \frac{\mu_0}{4\pi}\int_C \frac{I\left(d\vec{l} \times \vec{r}'\right)}{|\vec{r}'|^3}$$

Iz skice vidimo da vektor $\vec{r}'$ poprima vrijednost $\vec{r} - \vec{R}$ pa pišemo

$$(2b)\ \ \vec{B} = \frac{\mu_0}{4\pi}\int_C \frac{Id\vec{l} \times \left(\vec{r} - \vec{R}\right)}{\left|\vec{r} - \vec{R}\right|^3}$$

Raspisujemo pripadne vektore na kartezijanske koordinate. U ovoj fazi izvoda zanemarit ćemo kut α zato što sam kut isključivo služi razdvajanju x i y komponente polja

$$(2c)\ \ \vec{R} = R\cos\varphi\,\hat{\imath} + R\sin\varphi\,\hat{\jmath}$$

$$(2d)\ \ \vec{R} = r\sin\theta\,\hat{\imath} + R\cos\theta\,\hat{k}$$

Diferencijal $d\vec{l}$ postaje $d\vec{R}$ pa ga zapisujemo preko kružnog luka za neki mali kut $d\varphi$

$$(2e)\ \ d\vec{l} = d\vec{R} = \vec{R}d\varphi$$

$$(2f)\ \ d\vec{R} = (-R\sin\varphi\,\hat{\imath} + R\cos\varphi\,\hat{\jmath})d\varphi$$

$$(2g)\ \ \left|\vec{r}-\vec{R}\right| = \sqrt[2]{r^2 + R^2 - 2\left(r_x R_x + r_y R_y + r_z R_z\right)}$$

$$\left|\vec{r}-\vec{R}\right| = \sqrt[2]{r^2 + R^2 - 2rR\sin\theta\cos\varphi}$$

$$(2h)\ \ \left(\vec{r}-\vec{R}\right) = (r\sin\theta - R\cos\varphi)\hat{\imath} - R\sin\varphi\hat{\jmath} + r\cos\theta\,\hat{k}$$

$$(2i)\ \ \frac{1}{d\varphi}d\vec{R}\times\left(\vec{r}-\vec{R}\right) = (-R\sin\varphi\,\hat{\imath} + R\cos\varphi\,\hat{\jmath})\times((r\sin\theta - R\cos\varphi)\hat{\imath} - R\sin\varphi\hat{\jmath} + r\cos\theta\,\hat{k})$$

$$\frac{1}{d\varphi}d\vec{R}\times\left(\vec{r}-\vec{R}\right) = R^2\sin^2\varphi\,\hat{k} + Rr\cos\theta\sin\varphi\,\hat{\jmath} - Rr\sin\theta\cos\varphi\,\hat{k} + R^2\cos^2\varphi\,\hat{k} + Rr\cos\theta\cos\varphi\,\hat{\imath}$$

$$\frac{1}{d\varphi}d\vec{R}\times\left(\vec{r}-\vec{R}\right) = Rr\cos\theta\cos\varphi\,\hat{\imath} + Rr\cos\theta\sin\varphi\,\hat{\jmath} + (R^2\sin^2\varphi + R^2\cos^2\varphi - Rr\sin\theta\cos\varphi)\hat{k}$$

$$(2j)\ \ \frac{1}{d\varphi}d\vec{R}\times\left(\vec{r}-\vec{R}\right) = Rr\cos\theta\cos\varphi\,\hat{\imath} + Rr\cos\theta\sin\varphi\,\hat{\jmath} + (R^2 - Rr\sin\theta\cos\varphi)\hat{k}$$

Dobivene izaze uvrštavamo u formulu (7a) no umjesto po $d\vec{l}$ integriramo po $d\varphi$ u intervalu $[0, 2\pi]$

$$(3a)\ \ \vec{B} = \frac{\mu_0}{4\pi}I\int_0^{2\pi} d\varphi\,\frac{Rr\cos\theta\cos\varphi\,\hat{\imath} + Rr\cos\theta\sin\varphi\,\hat{\jmath} + (R^2 - Rr\sin\theta\cos\varphi)\hat{k}}{(r^2 + R^2 - 2rR\sin\theta\cos\varphi)^{\frac{3}{2}}}$$

S obzirom na granice integrala uočavamo da je $\sin 2\pi = \sin 0 = 0$ pa član $Rr\cos\theta\sin\varphi\,\hat{\jmath}$ integriranjem otpada.

$$(3b)\ \ \vec{B} = \frac{\mu_0}{4\pi}I\int_0^{2\pi} d\varphi\,\frac{Rr\cos\theta\cos\varphi\,\hat{\imath} + (R^2 - Rr\sin\theta\cos\varphi)\hat{k}}{(r^2 + R^2 - 2rR\sin\theta\cos\varphi)^{\frac{3}{2}}}$$

Sada kada smo pronašli izraz vraćamo kut α kao vrijednost koja rastavlja izraz $Rr\cos\theta\cos\varphi$ na x i y članove. Konačno dobivamo

$$(4)\ \ \vec{B} = \frac{\mu_0}{4\pi}I\int_0^{2\pi} d\varphi\,\frac{Rr\cos\theta\cos\varphi\cos\alpha\,\hat{\imath} + Rr\cos\theta\cos\varphi\sin\alpha\,\hat{\jmath} + (R^2 - Rr\sin\theta\cos\varphi)\hat{k}}{(r^2 + R^2 - 2rR\sin\theta\cos\varphi)^{\frac{3}{2}}}$$

No ovakav izraz ima jedan nedostatak – ne može se integrirati. Naime radi se o definiranom eliptičnom integralu trećeg reda, kojemu nismo pronašli prikladnu aproksimaciju. To povlači da je potrebno manualno integrirati na način da postavimo vrlo mali $d\varphi$ te sumiramo u intervalu $[0, 2\pi]$. To ćemo učiniti programski, no prije nego to učinimo pronađimo izraz za cijelu zavojnicu.

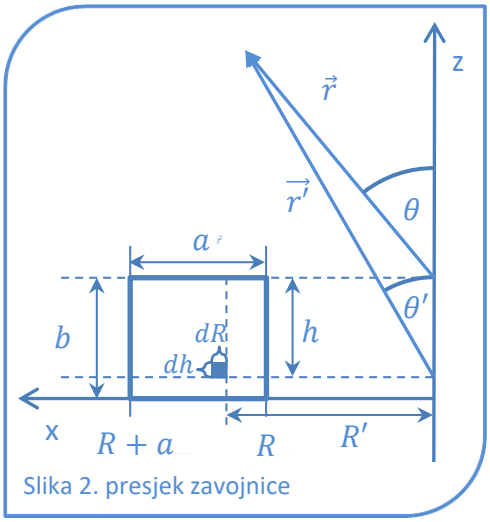
Za zavojnicu smo prethodno rekli da se sastoji od više individualnih petlji. Za zavojnicu pravokutnog presjeka vrijede izrazi (4a), (4b) i (4c) iz uvoda. Dakle potrebno je integrirati u dodatne dvije dimenzije, a pritom struja I više nije dostatna nego nam je potrebna gustoća struje J

$$(5a)\ \ dI = J dS$$

$$(5b)\ \ dS = dh\, dR$$

Gustoća struje je konstantna i iznosi

$$(5c)\ \ J = \frac{NI}{ab} = I\frac{ML}{ab} = \frac{I}{d^2}$$



Slika 2. presjek zavojnice

Uvedimo dvije varijable koje nam olakšavaju daljnji izvod

$$(6a)\ \ x = r\sin\theta$$

$$(6b)\ \ z = r\cos\theta$$

Presjek zavojnice prikazan je na Slici 4. S obzirom da je integral (9) nema antiderivat, niti integrali njega samoga nemaju antiderivat. Dakle opet promatramo vrlo male odsječke površine zavojnice. Uočavamo da se radius mijenja u intervalu $[R, R+a]$. Visina se mijenja u intervalu $[0, b]$. Promjena visine kao što vidimo utječe na $\vec{r}$. Njegova x komponenta ostaje očuvana kako vrijedi (11a), dok se komponenta z mijenja i postaje z'

$$(7a)\ \ x = r\sin\theta = r'\sin\theta'$$

$$(7b)\ \ z' = z + h$$

Sada možemo zapisati volumni integral. Dobivamo

$$(8a)\ \ \vec{B} = \frac{\mu_0}{4\pi} J \int_R^{R+a}\int_0^b dR\, dh \int_o^{2\pi} \frac{R'z'\cos\alpha\cos\varphi\,\hat{\imath} + R'z'\sin\alpha\cos\varphi\,\hat{\jmath} + (R'^2 - R'x\cos\varphi)\hat{k}}{(R'^2 + r'^2 - 2R'x\cos\varphi)^{\frac{3}{2}}} d\varphi$$

$$(8b)\ \ r'^2 = x^2 + (z+h)^2$$

Sada konačno možemo uvrstiti konačnu formulu za magnetsko polje točke. Kao unose funkcija uzima polarne koordinate prilagođene zavojnici (r, α, θ), jakost struje koja prolazi zavojnicom I, broj namotaja N, unutarnjeg promjera R te debljinu a i dužinu b. Zapisujemo ju fragmentirano zbog prostora i razumijevanja same funkcije.

$$(9)\ \vec{B} = \frac{\mu_0}{4\pi}\frac{NI}{ab}\int_R^{R+a}\int_0^b dR\,dh\int_o^{2\pi}\frac{G_x\hat{\imath} + G_y\hat{\jmath} + G_z\hat{k}}{D^{1.5}}d\varphi$$

$$(9a)\ G_x = R'(r\cos\theta + h)\cos\alpha\cos\varphi$$

$$(9b)\ G_y = R'(r\cos\theta + h)\sin\alpha\cos\varphi$$

$$(9c)\ G_z = R'^2 - R'r\sin\theta\cos\varphi$$

$$(9d)\ D = R'^2 + r^2\sin^2\theta + (r\cos\theta + h)^2 - 2R'r\sin\theta\cos\varphi$$

Ukoliko nas zanima samo određena komponenta polja možemo samo nju uvrstiti, no kako god potreban nam je računalni program ukoliko bismo uopće mogli izračunati polje zavojnice. Za izračun ukupnog polja u točki koristimo modul

$$(9e)\ B = \sqrt[2]{{B_x}^2 + {B_y}^2 + {B_z}^2}$$

Indirektnu potvrdu valjanosti ovih jednadžbi pronašli smo u NASA-inu članku na temu sile i magnetskog polja[2] u kojemu je određen vektorski potencijal polja dan jednadžbom

$$(10a)\ A = \frac{\mu_0 R}{4\pi}\int_0^{2\pi}\frac{\cos\varphi\,d\varphi}{\sqrt{r^2 + R^2 - 2rR\sin\theta\cos\varphi}}$$

Kako je vektorski potencijal ništa drugo nego divergencija magnetskog polja dana jednadžbom

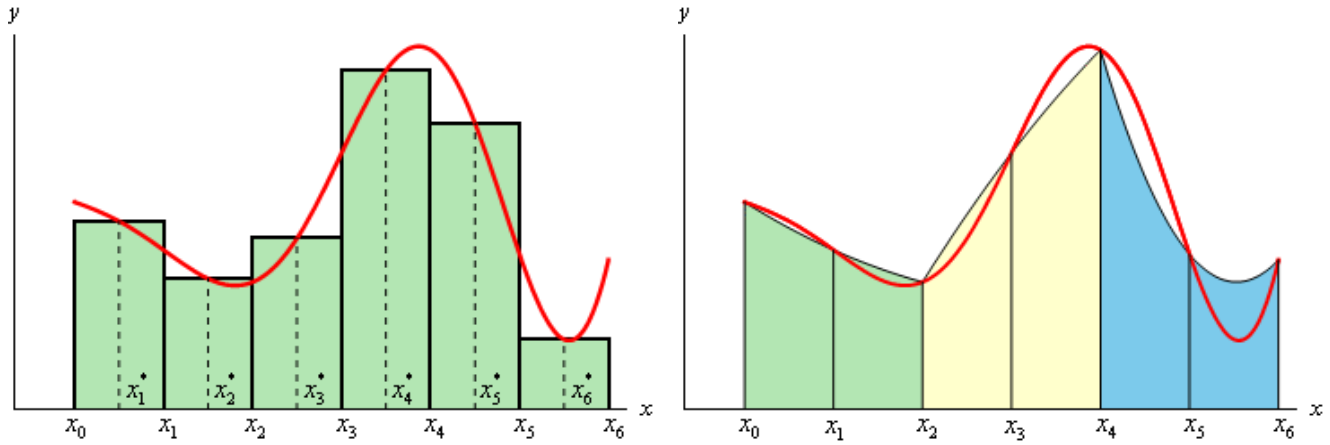
$$(10b)\ \vec{A} = \nabla\times\vec{B}$$

No valja naglasiti da se jednadžba polja (4) ne može dobiti samo iz vektorskog potencijala, već vektorski potencijal više služi kao provjera valjanosti jednadžbe.

3.2. Integralna aproksimacija

Ključnu ulogu u određivanju preciznosti igra metoda aproksimacije. Dvije metode aproksimacije u našem slučaju se preko „pravila srednje točke“ (slika 9. lijevo) te preko Simpsonovog pravila (slika 9. desno). Iako smo započeli sa metodom srednje točke uočili smo da ona proizvodi znatne pogreške u računanju te da joj je potreban poprilično velik broj inkremenata. Za razliku od nje Simpsonovo pravilo uzima u obzir 3 varijable x koje se međusobno razlikuju za dx te s obzirom da u tom slučaju aproksimiramo površinu kvadratne funkcije radije nego kvadrata (konstante) dobivamo znatnu količinu preciznosti. Formula za prosjek u intervalu $2dx$.) je

$$(11)\ f(x_0, x_1, x_2) = \frac{dx}{3}(x_0 + 4x_1 + x_2)$$



Slika 3. Prikaz dviju metode aproksimacije integrala za proizvoljnu krivulju

Slika 3. i formula su preuzete sa web stranice [3]

Ključna komponenta za uspješnu evalucaiju pogreške funkcije je pronalazak nekakve reference za koju znamo da je precizna. Ako pogledamo malo pomnije formulu (4) možemo vidjeti da bi se mogla integrirati ukoliko bismo uzeli da točka leži isključivo na z osi. Kako vrijedi $\theta = 0$ tada formula postaje

$$(12a)\ \vec{B} = \frac{\mu_0}{4\pi} I \int_0^{2\pi} d\varphi \frac{Rr\cos\varphi\cos\alpha\,\hat{\imath} + Rr\cos\varphi\sin\alpha\,\hat{\jmath} + R^2\hat{k}}{(r^2 + R^2)^{1.5}}$$

Integral nezavisnog $\cos\varphi$ u intervalu $[0, 2\pi]$ jednak je 0 pa se komponente $\hat{\imath}$ i $\hat{\jmath}$ ne pojavljuju

$$(12b)\ \vec{B} = \frac{\mu_0}{4\pi} I \int_0^{2\pi} d\varphi \frac{R^2\hat{k}}{(r^2 + R^2)^{1.5}}$$

$$\vec{B} = \frac{\mu_0}{4\pi} I \frac{2\pi R^2\hat{k}}{(r^2 + R^2)^{1.5}}$$

$$(12c)\ \vec{B} = \frac{\mu_0}{2} I \frac{R^2\hat{k}}{(r^2 + R^2)^{1.5}}$$

Poznavanje ovakve relacije vrlo nam je važno zato što nam omogućuje direktnu usporedbu vrijednosti B_z dobivenih manualnom integracijom u usporedbi s funkcijom (9). Funkciju možemo prilagoditi za izračun B_z komponente zavojnice a u tom slučaju vrijedi

$$(13)\ \vec{B} = \frac{\mu_0}{4\pi} J \int_R^{R+a} \int_0^b dR\, dh \frac{R'^2 \hat{k}}{((z+h)^2 + R'^2)^{1.5}}$$

Sada imamo sredstvo uspoređivanja funkcija uz uvjet da je $\theta = 0$. U računalu se realni brojevi zapisuju na 2 glavna načina odnosno tipa podataka: kao „single-precision floating point“ te kao „double-precision floating point“. Glavna razlika među njima je preciznost: Single-precision (nadalje u radu ćemo zvati ovakvu preciznost samo single) precizan je na 6 značajnih znamenki dok je Double-precision precizan na 15 značajnih znamenki (double). Ne očekujemo programsku pogrešku manju od 0.001% pa nema pretjerane poante koristiti double, osim ako nije cilj dobiti maksimalnu preciznost. No kako računamo koristeći CUDU, a nemamo na raspolaganju Titan V ili neku noviju Quadro karticu, uopće nećemo razmatrati double preciznost.

S obzirom da je manualna integracija ništa drugo nego zbrajanje malih inkremenata, potrebno je definirati i njihovu preciznost. Integriramo po 3 dimenzije: duljini, širini i kutu. Broj inkremenata kuta označavamo sa IncFi, a tipičan interval u kojemu se on nalazi je $[16, 64]$ te je uvijek nužno paran broj. Za duljinu i širinu broj inkremenata određujemo preko same dimenzije podijeljene s korištenim inkrementom. To zapravo definira broj kalkulacija potreban za svaku zavojnicu, odnosno ako pomnožimo broj inkremenata po kružnici, po duljini i po debljini možemo dobiti uvid u vrijeme računanja. Ako razmatramo zavojnice fizikalno u tom slučaju je dio koji nas najviše zanima na koliko je dijelova podijeljena žica. Kako je žica nužno kružnog presjeka nužno je da veličina inkrementa po duljini jednaka veličini inkrementa po debljini. Ovo donosi najbolje rezultate i preporučeno je koristiti to. Na temelju ovih parametara preciznost ćemo nadalje označavati u formatu single 24 x4. Prvi dio se referira na preciznost (single ili double, gotovo uvijek korišten single), drugi dio na broj inkremenata IncFi (bilo koji paran broj veći od 12), a treći na koliko kvadratnih dijelova smo podijelili površinu jedne žice (dio skupa $\{1,4,9,16,25 \dots\}$). Standardna preciznost programa je upravo single 24 x4, a pritom je debljina žice 1mm. Valja naglasiti da su ovo više upute za uspješnije korištenje nego pravila.

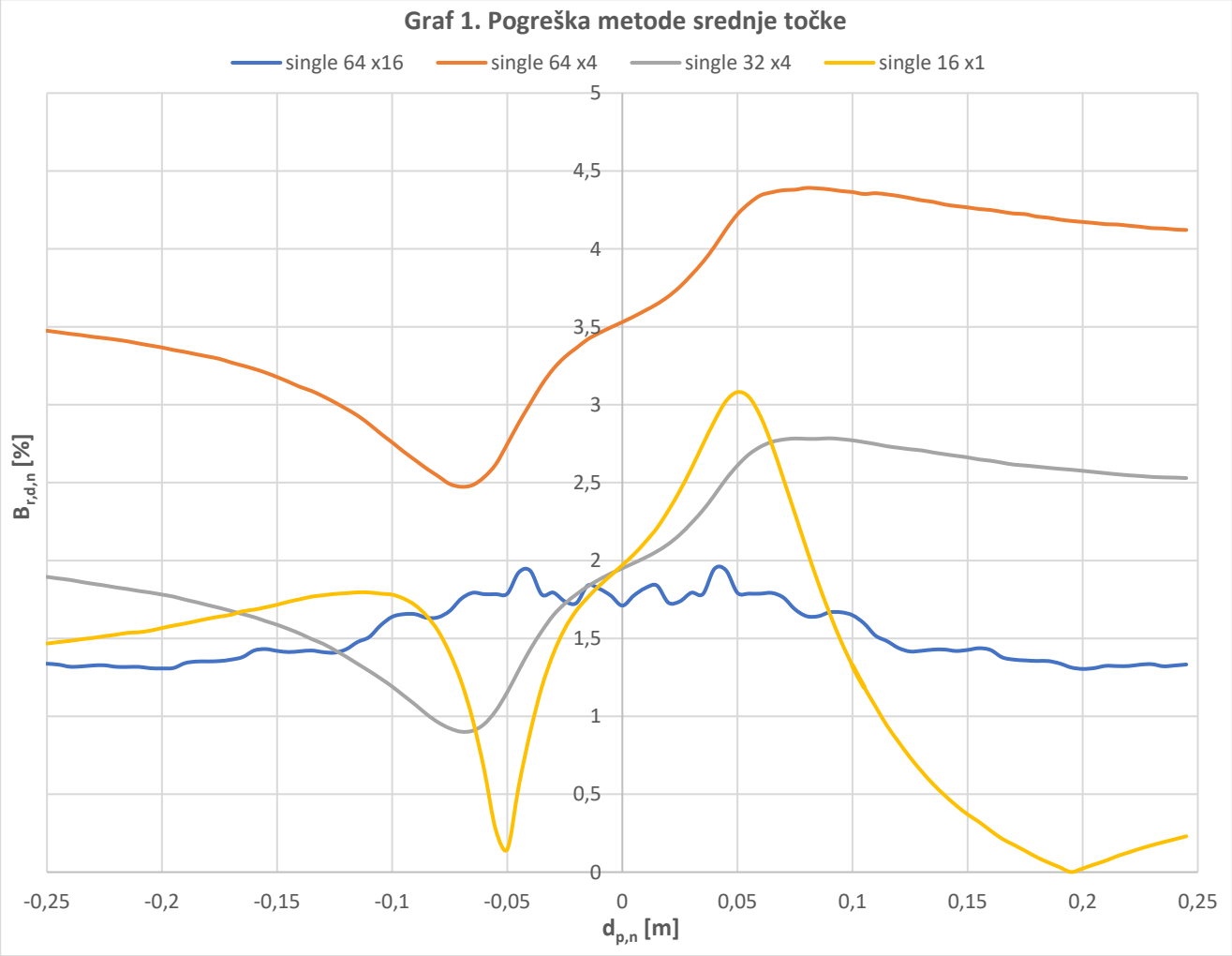
Pogrešku općenito računamo kao

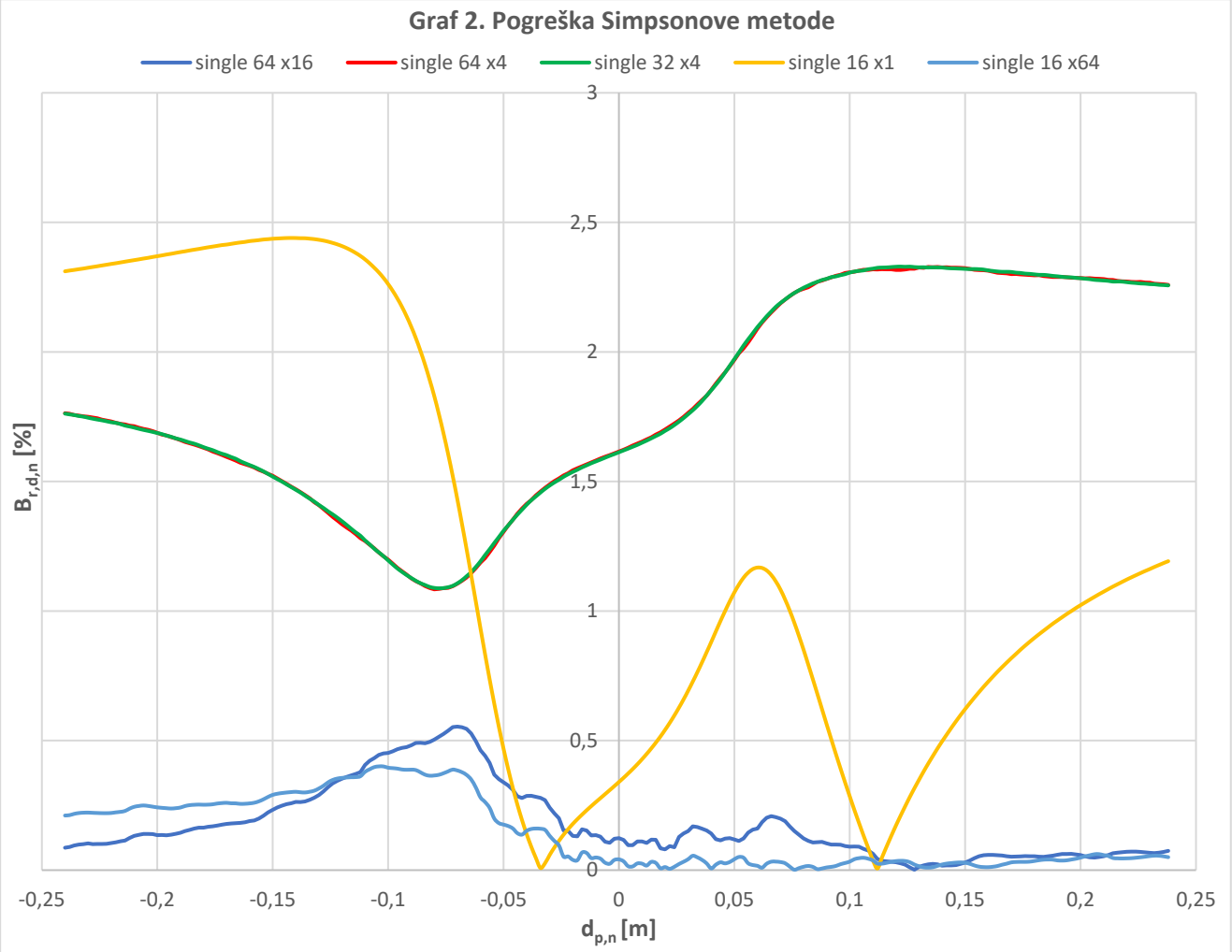
$$(14a)\ B_{r,d,n} = \frac{\left|B_{t,13,d,n} - B_{t,9,d,n}\right|}{B_{t,13,d,n}}$$

Vrijednosti $B_{t,24,d,n}$ za neku udaljenost od središta zavojnice d_p, dobivene su računanjem forumule (13) s preciznošću double x256 (drugi dio je izostavljen zato integriramo samo po površini) dok su vrijednosti $B_{t,24,d,n}$ izračunate uporabom obje metode u preciznostima single 64 x16, single 64 x4, single 32 x4, single 16 x4. U prvom grafu prikazujemo vrijednosti za prvu metodu dok u drogome za drugu. Zatim računamo prosječnu pogrešku kao

$$(14b)\ \overline{B_{r,d,n}} = \frac{\sum_0^n B_{r,d,n}}{n}$$

Dimenzije zavojnice koja je korištena su: debljina 0.03m, unutarnji promjer 0.03m, duljina 0.12m.





Na Grafu 2. crvena i ze crta u potpunosti preklapaju. To nam govori da zapravo za Simpsonovo pravilo nije previše bitan inkrement po kutu IncFi. Za tu svrhu dodana je preciznost single 16 x64 za koju vidimo da pogreška teži 0. Očito je dakle da je simpsonova metoda bolja za izračun polja. Prosječne pogreške za redom x64, x16, x4, x1 te fiksan IncFi=20 su

$$(A)\ \overline{B_{r,x64}} = 0.1424\ \%$$

$$(B)\ \overline{B_{r,x16}} = 0.1794\%$$

$$(C)\ \overline{B_{r,x4}} = 1.8112\%$$

$$(D)\ \overline{B_{r,x1}} = 1.2495\%$$

Uočavamo da je za osnovnu primjenu dostatna preciznost single 16 x1. Ukoliko nam stvarno treba velika preciznost u tom slučaju je preporučljiva uporaba single 32 x64 koji proizvodi gotovo zanemarivu pogrešku.

4. Informatička strana rješenja (Nikola Sočec)

4.1. Općenito o funkciji za računanje

Funkciju za računanje magnetskog polja u određenoj točki pored zavojnice je originalno napisao Davor jer ju je on razvio matematički. Problem je bio u tome što funkcija prima argumente kao što su udaljenost tražene točke od središta zavojnice i kut koji tražena točka zatvara s osi zavojnice. Pri testiranju Davorove funkcije, te informacije smo koristili kao konstante, ali je u slučaju pravog programa te vrijednosti trebalo izračunati. Udaljenost tražene točke od središta zavojnice je jednostavno izračunati, ali metoda računanja kuta između osi zavojnice i pravca koji prolazi središtem zavojnice i traženom točkom mi nije odmah pala na pamet. Rješenje je u tome da se uz pomoć kosinusovog poučka i udaljenosti nekih rubnih točaka zavojnice s traženom točkom taj kut može jednostavno dobiti. U prvim inačicama Davorove funkcije jedan od parametara je bio kut koji tražena točka zatvara u ravnini zavojnice s ishodištem polarnog sustava, što je imalo smisla u polarnom sustavu u kojem je funkcija razvijena, ali u trodimenzionalnom sustavu je stvarao velike probleme. Rješenje je ležalo u tome da je ukupno polje identično na svakoj kružnici oko središnje površine zavojnice, a njegov smjer je uvijek zbroj komponenti od kojih je jedna usporedna sa zavojnicom (z komponenta), a druga suprotna vektoru koji ide od tražene točke prema središtu ravnine (h komponenta / horizontalna komponenta). Uz pomoć te opservacije, shvatio sam da je jedino potrebno izračunati vektor osi zavojnice i vektor koji je okomit na tu os i spaja traženu točku i os. Davorova funkcija je davala iznose tih vektora, a tom metodom se mogao dobiti iznos svake njihove komponente. Prilikom računanja s više zavojnica, za svaku zavojnicu se zasebno računa polje i onda se ta polja zbrajaju, što omogućuje prikaz interakcije zavojnica u prostoru. Tek nakon što su izračunata polja svih zavojnica i njihova interakcija, računa se ukupno polje koje je modul svih komponenata polja u svakoj točki.

4.2. Generiranje slika

Nakon računanja polja oko zavojnica, potrebno je izgenerirati slike tih polja. Problem je u tome što se polje drastično razlikuje već unutar jednog okvira koji prikazuje jednu komponentu, a nepreciznosti u funkciji mogu uzrokovati nekoliko točaka po okviru s izrazito prevelikim poljima. U nekim perspektivama se polja poništavaju, pa im je vrijednost bliska 0. Potrebno je uskladiti prikaze svih polja na svim komponentama. Kao rješenje smo izabrali koristi logaritamsko prikazivanje boja tako da se drastične razlike među poljima ublaže. Uz to, bilo je potrebno zanemariti određen postotak najvećih vrijednosti zbog nepreciznosti u funkciji i staviti ograničenje na minimalnu vrijednost koja se uzima u obzir (zbog poništavanja polja).

U obzir se uzimaju samo točke ukupnog polja. Same boje se računaju prema formuli: $r(x, y) = \log_{10}(B(x, y) \cdot min\_offset) / \log_{10}(B_{max}) \cdot 255$, $b(x, y) = 255 – r(x, y)$ gdje je $r(x, y)$ crvena komponenta boje u točki, $B(x, y)$ polje u točki, min_offset vrijednost kojom se svaka točka množi tako da sve budu veće od 1 (kako bi logaritam uvijek bio pozitivan), B_{max} globalno najveće trenutno izračunato prihvatljivo polje (bez ekstremnih polja uzrokovanih nepreciznostima), a $b(x, y)$ plava komponenta boje u točki. Ekstremne točke se smanjuju na vrijednost trenutnog prihvatljivog maksimuma ili povećavaju na vrijednost trenutnog prihvatljivog minimuma. Točke koje se nalaze u zavojnici su crne. Te točke se pronalazi funkcijom koja pomoću kosinusovog poučka i kutova detektira je li točka u zavojnici ili ne.

4.3. Ubrzavanje računanja uz pomoć CUDA alata

Glavni problem koji još nisam spomenuo vezan uz funkciju je njezina zahtjevnost. Na normalnim preciznostima računanja, procesoru treba velika količina vremena kako bi izračunao samo nekoliko točaka, a normalna slika se dobiva tek na vrijednostima iznad nekoliko desetaka tisuća točaka. Funkciju je bez sumnje trebalo računati pomoću grafičke kartice. Zbog mojeg prošlog iskustva s alatom CUDA, koji je razvila NVIDIA, odlučio sam da je to najbolja opcija za ubrzavanje računanja funkcije. Kako bi se funkcija ubrzala pomoću grafičke kartice, prvo ju je potrebno paralelizirati. Davorova osnovna funkcija je bila u potpunosti serijska. Prema onome što sam naučio u CUDA dokumentaciji, za grafičke kartice koje koriste CUDA računanje je pogodno napraviti što je moguće više threadova koji rade što je moguće manje posla (u našem slučaju može doći i do nekoliko milijuna threadova) jer ih kontroler u kartici raspoređuje među njezinim procesorima u blokovima. Kompleksnost računanja najbolje pokazuje činjenica da je minijaturna promjena u kodu, kao što je spremanje zbrajanja dvije vrijednosti u varijablu umjesto zbrajanja tih dviju vrijednosti dvaput, osjetna u redu veličine sekunde na ukupnom vremenu računanja od 30-ak sekundi. Jedan od problema s korištenjem velikog broja threadova je spremanje rezultata računanja. Naime, od funkcije se očekuje da će rezultat spremiti u jednu varijablu, a milijuni threadova pokušavaju pisati u tu istu varijablu. Prvo rješenje je bilo korištenje atomic operacija (svaki thread čeka svoj red da upiše rezultat u varijablu), ali one su izuzetno spore. To sam riješio korištenjem „shared“ memorije koja se dijeli između svih threadova u jednom bloku i izuzetno je brza. Svaki thread zapiše svoj rezultat u svoj komad memorije i na kraju jedan od threadova u bloku (prvi thread u našoj implementaciji) sumira rezultate ostalih threadova i upiše ga u konačan rezultat. To je izuzetno ubrzalo izvođenje funkcije. Sve funkcije su napravljene tako da imaju isto sučelje kako bi se u glavnom programu mogle koristiti bez ikakvih preinaka glavnog programa.

4.4. Ubrzavanje računanja uz pomoć jezgara procesora

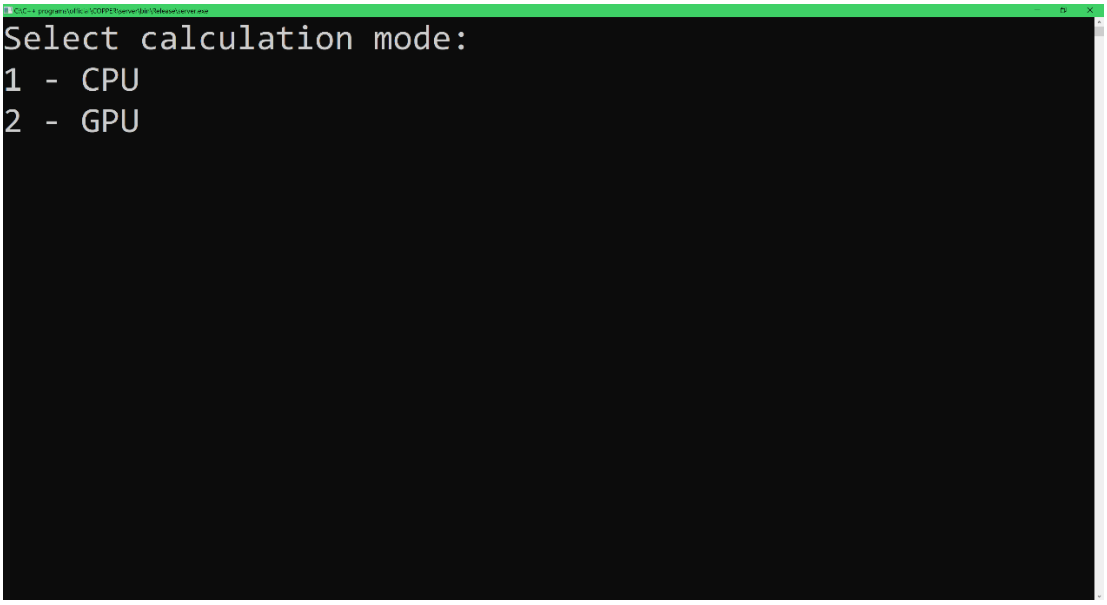
S obzirom na to da dosta računala nema instaliran CUDA alat ili uopće nema NVIDIA grafičku karticu mlađu od 5 godina, bilo je potrebno napraviti verziju funkcije koja funkcionira samo uz pomoć procesora. Brzina jednog threada procesora je premala za bilo kakvo realno vrijeme računanja, ali već dugo procesori dolaze s 2 ili više threadova, a neki procesori (Xeoni i Threadripperi) dolaze s izuzetno velikim brojem threadova (neki prelaze 32 threada), što omogućuju računanje slika minimalne kvalitete, ali u relativno prihvatljivom vremenu. Ponovno je bilo potrebno paralelizirati osnovnu funkciju jer procesori funkcioniraju suprotno od grafičkih kartica. Kreiranje threada u Windowsu je vremenski skupo i stvaranje više threadova od broja fizičkih threadova procesora neće donijeti nikakvu korist za računanje (uglavnom će još dodatno usporiti računanje). Srećom, ovu vrstu paralelizacije (malo threadova, puno posla po threadu) je lakše napraviti. Potrebno je podijeliti prvi stupanj računanja (broj točaka koje treba izračunati) na onoliko dijelova koliko ima fizičkih procesorskih threadova (taj broj se može dohvatiti pomoću WinAPI funkcije), pokrenuti te threadove i pričekati da završe. Ovom metodom se mogu u prihvatljivom vremenu izračunati slike veličina oko 50 x 50 (optimalna veličina je 500 x 500, 100 puta veća).

4.5. Računanje preko interneta

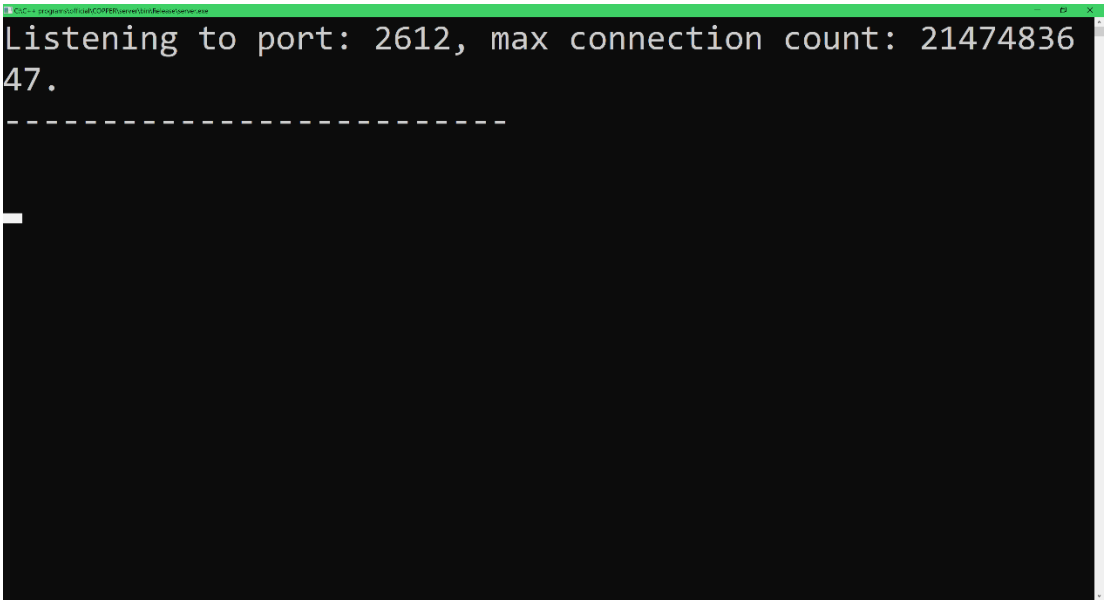
Računanje preko interneta je bio ambiciozni sekundarni pokušaj omogućavanja slabim uređajima da koriste COPPER. Ova metoda ima puno veći potencijal jer čak i jako slaba računala imaju internetske brzine od nekoliko desetaka Mb, a sama ne moraju računati polja. Računanje se odvija na serveru koji ima vrlo moćan hardware i može omogućiti složenosti kalkulacija koje nadilaze slike veličine 500 x 500 u vremenu ispod minute. Nazvao sam ovo „ambicioznim pokušajem“ zato što sam ga implementirao, ali je implementacija vrlo nestabilna. Kako bih usavršio ovu funkciju, moram se posvetiti istraživanju i eksperimentiranju s internetom i slanjem velikih količina podataka preko interneta. Trenutno funkcija radi na sljedeći način:

1. klijent serveru pošalje konstante za računanje
2. klijent serveru u paketima šalje parametre za računanje
3. za svaki paket server vraća količinu primljenih podataka
4. klijent provjerava je li server primio sve podatke i traži ponavljanje postupka ako nije
5. nakon uspješnog primanja svih paketa, server počinje računati
6. klijent se spaja na server i dobiva informaciju o statusu računanja
7. kada je računanje gotovo, server klijentu na upit odgovara da je računanje 100% gotovo
8. server počinje slati rezultate klijentu na isti način kao što je klijent slao serveru parametre
9. nakon uspješnog primanja svih podataka, server se vraća u početno stanje

Problemi ovoga sustava su u tome što paketi vrlo često ne stižu u cijelosti i značajan broj puta se komunikacija između servera i klijenta zbog toga zablokira. Uz to, samo jedan klijent može biti spojen na server odjednom.



Server je implementiran kao konzolna aplikacija. Pri njegovom pokretanju, korisniku je na izbor dano koju metodu računanja će server koristiti (u 99% slučajeva bi to trebala biti grafička kartica).



Nakon izbora metode računanja, pojavit će se ovi podatci. Server je implementiran tako da se uvijek spaja na port 2612. Kada se klijent spoji na server, ovdje će se pojaviti informacije o interakcijama između njega i servera.

4.6. WHS

WHS je nastao zbog mojeg nezadovoljstva sučeljem jedne druge biblioteke, SDL, i zbog puke znatiželje o funkcioniranju Windows aplikacija. WHS nipošto nije bolji od SDL-a, ali je napravljen isključivo za moje specifične potrebe i donio mi mnogo znanja. Njegove funkcije osiguravaju stvaranje prozora, obrađivanje događaja koji se događaju prozorima, spremanje slika u memoriju i na disk, razvlačenje slika, color keying, konstantnu petlju koja određeni broj puta u sekundi zove funkciju za osvježivanje grafičkog sučelja, crtanje teksta na prozor i crtanje slika na prozor. Sve je to umotano u klase, što čini biblioteku izuzetno jednostavnom za korištenje. S druge strane, oko WHS-a ima izuzetno mnogo prostora za poboljšanje. Slike se trenutno spremaju u mojim spremnicima (tzv. „pixel array“), što zahtijeva konverzije u bitmapove koje WinAPI zahtijeva kako bi nacrtao sliku u prozor. Rješenje tog problema je spremanje slika u WinAPI bitmapove. Također, procese kao rastezanje slika bi trebalo ubrzati grafičkim karticama. Uz to, konstantno osvježivanje programa nije isplativo za aplikacije koje nisu videoigre jer zauzima jedan cijeli thread procesora. Ponovno crtanje elemenata grafičkog sučelja se može minimizirati tako da se događa isključivo kad korisnik napravi neku promjenu (npr. pritisne neki gumb). Na WHS-u planiram raditi tako da njegovo korištenje u ozbiljnim aplikacijama postane opravdano.

5. Tehnički podatci sustava

5.1. Korištene specifikacije

- PC
  - Procesor: Intel Core i7 8700k (6C/12T, 5.0Ghz)
  - Memorija: 32GB (2x16GB 3000MHz)
  - Grafička kartica: Nvidia GTX 1080 8GB
- Laptop
  - Procesor: Intel Core i7 7700HQ (4C/8T, 3.4GHz)
  - Memorija: 16GB (2x8GB, 2400MHz)
  - Grafička kartica: Nvidia GTX 1050 4GB

5.2. Minimalne speficikacije

- Procesor: Intel Core 2 Duo
- Memorija: 4GB
- Grafička kartica: /

(Iako je ovo tehnički minimalna specifikacija nikako ne preporučamo radi potrebnog vremenam, eventualno ako ste spojeni na server)

- Procesor: Intel Core i5 6400U ili slično
- Memorija: 8GB
- Grafička kartica: Nvidia GTX 950 ili slično

5.3. Preporučene specifikacije

- Procesor: Intel Core i7 6700 ili bolje
- Memorija: 16GB
- Grafička kartica: Nvidia GTX 1070

(Ako vam je ovo za svrhe istraživanja možete i razmisliti o sljedećemu)

- Procesor: Intel Core i9 9700k ili bolje
- Ram: 32GB ili 64GB (za rendere visoke rezolucije)
- Grafička: Nvidia RTX 2080 Ti

***Jedini operacijski sustav na kojem program garantirano radi je Windows 10**

6. Reference

[1] Biot-Savart-Laplaceov zakon za magnetno polje

https://en.wikipedia.org/wiki/Biot%E2%80%93Savart_law

[2] NASA-in članak pomoću kojega smo provjerili valjanost naše formule

https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20140002332.pdf

[3] Metode aproksimacije integrala

http://tutorial.math.lamar.edu/Classes/CalcII/ApproximatingDefIntegrals.aspx