Rješavanje običnih diferencijalnih jednadžbi metodom konačnih razlika

Stipanović, Petar

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Prijediplomski sveučilišni studij elektrotehnike

Završni rad

NUMERIČKO RJEŠAVANJE OBIČNIH DIFERENCIJALNIH JEDNADŽBI METODOM KONAČNIH RAZLIKA

Rijeka, rujan 2023.

Petar Stipanović
0069086694

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Prijediplomski sveučilišni studij elektrotehnike

Završni rad

NUMERIČKO RJEŠAVANJE OBIČNIH DIFERENCIJALNIH JEDNADŽBI METODOM KONAČNIH RAZLIKA

Mentor: izv. prof. dr. sc. Ivan Dražić

Komentor: doc. dr. sc. Angela Bašić-Šiško

Rijeka, rujan 2023.

Petar Stipanović
0069086694

Želim izraziti duboku zahvalnost svojim mentorima, izv. prof. dr. sc. Ivanu Dražiću i doc. dr. sc. Angeli Bašić-Šiško, na njihovoj neizmjerljivoj podršci, stručnim savjetima i nesebičnom angažmanu tijekom procesa izrade ovog završnog rada. Vaše zajedničko mentorstvo i posvećenost su bili ključni faktori za postizanje ovog rezultata, i zato sam vam duboko zahvalan.

Također, želim izraziti duboku zahvalnost mojim roditeljima i cijeloj obitelji. Vaša vjera u moje sposobnosti i neprestana podrška su mi bili neophodni tijekom ovog akademskog putovanja. Hvala vam što ste bili moj najveći oslonac i motivacija.

Posebna zahvala ide i mojim prijateljima i kolegama koji su mi pomagali i podržavali tijekom cijelog ovog školovanja.

Sadržaj

1. Uvod

Diferencijalne jednadžbe se intenzivno koriste za modeliranje širokog spektra problema u elektrotehnici, mehanici čvrstog i fluidnog materijala, biologiji, znanosti o materijalima, ekonomiji, ekologiji, informatici itd. Neke od najpoznatijih diferencijalnih jednadžbi su Navier Stokesove jednadžbe u dinamici fluida, Maxwellove jednadžbe za elektromagnetizam itd.

Nažalost, iako diferencijalne jednadžbe mogu opisati velik broj problema, samo mali dio njih može se egzaktno riješiti pomoću elementarnih funkcija (polinomi, sinus, kosinus, logaritmi itd.) i njihovih kombinacija (kompozicijske funkcije). Neke klase diferencijalnih jednadžbi mogu se riješiti analitički, no često to zahtijeva velike napore i primjenu naprednih matematičkih teorijskih znanja. Ako analitičko rješenje nije dostupno, poželjno je pronaći približno rješenje jednadžbe. To se postiže numeričkim metodama. U ovom radu, pokazat ću kako numerički riješiti diferencijalne jednadžbe, koristeći jednu od metoda rješavanja, metodu konačnih razlika (eng.finite difference method - FDM).

Metoda konačnih razlika je aproksimacijska metoda za rješavanje raznih rubnih i početno-rubnih diferencijalnih problema za diferencijalne jednadžbe. To uključuje linearne i nelinearne, vremenski neovisne i ovisne probleme. Ova se metoda može primijeniti na probleme s različitim oblicima ruba, različitim vrstama rubnih uvjeta i za područje koje sadrži više različitih materijala.

U metodi konačnih razlika, derivacije u diferencijalnoj jednadžbi aproksimiraju se pomoću formula konačnih razlika. Na taj način možemo transformirati diferencijalnu jednadžbu u sustav algebarskih jednadžbi za lakše rješavanje. Tridijagonalni sustavi su posebna vrsta linearnih sustava s ne-nul elementima samo na glavnoj i dvjema susjednim sporednim dijagonalama. Oni se često koriste u različitim područjima kao što su numerička analiza, inženjerstvo i financije. Prednost tridijagonalnih sustava je da se mogu rješavati efikasnijim algoritmima poput Thomasovog algoritma, koji ima linearnu složenost u odnosu na broj nepoznanica. Njihova posebna struktura pruža mogućnost razvijanja specifičnih metoda za rješavanje, čime se postiže dublje razumijevanje i primjena matematičkih koncepta u rješavanju linearnih sustava.

Aproksimacija derivacija konačnim razlikama je jedna od najjednostavnijih i najstarijih metoda rješavanja diferencijalnih jednadžbi. Koristio ih je L. Euler već u 18. stoljeću, za probleme jednodimenzijskog prostora, dok ih je početkom 20. st., najvjerojatnije, C. Runge proširio na primjenu u dvodimenzionalnim prostorima. Nagli razvoj metode konačnih razlika u numeričkim primjenama započele su ranih 1950-ih potaknuti pojavom i razvojem računala koji su omogućili tehnološku podlogu za rješavanje složenih problema znanosti i tehnologije.

Python je popularan programski jezik koji se često koristi u numeričkoj analizi. Zahvaljujući jednostavnosti i bogatoj biblioteci modula, Python pruža programerima moćan alat za implementaciju i analizu numeričkih metoda konačnih razlika. Koristeći Python, možemo lako izračunati

numeričke aproksimacije diferencijalnih jednadžbi, simulirati fizikalne procese i analizirati rezultate. Sintaksa Pythona olakšava pisanje čitljivog koda, dok brojne biblioteke poput NumPy i SciPy pružaju funkcionalnosti za brzo izračunavanje i korištenje numeričkim podacima.

U ovom radu pružit ćemo sveobuhvatan pregled strukture i primjene metode konačnih razlika na rubne probleme. Započet ćemo s uvodom u osnovne pojmove koji su ključni za razumijevanje rubnih problema i njihovo rješavanje. Nakon toga, detaljno ćemo objasniti metodu konačnih razlika, koja je jedna od najčešće korištenih numeričkih tehnika za rješavanje rubnih problema. Opisat ćemo korake i postupke primjene metode, ističući njezine prednosti i ograničenja. Ova analiza će omogućiti razumijevanje kako metoda konačnih razlika funkcionira i na koji način se primjenjuje u praksi. Jedan od ključnih dijelova rada bit će primjena metode konačnih razlika na konkretnim primjerima iz područja inženjerstva. Kroz ove primjere, čitatelji će dobiti uvid u stvarne probleme koje metoda konačnih razlika može rješavati, kao i njenu primjenu u različitim inženjerskim disciplinama. Analizirat ćemo rezultate i donositi zaključke o primjenjivosti metode na različite scenarije. Da bismo omogućili praktičnu primjenu metode konačnih razlika, također ćemo implementirati ovu metodu koristeći Python programski jezik.

2. Osnovni pojmovi

U ovom poglavlju dajemo kratak uvod u osnovne matematičke pojmove i tvrdnje koje su bitne za metodu konačnih razlika.

2.1. Derivacija

Derivacija je matematička operacija koja označava stopu promjene neke funkcije u odnosu na promjenu njezine varijable. Derivacija funkcije $f : D \to R$ u točki $x \in D$ se definira kao:

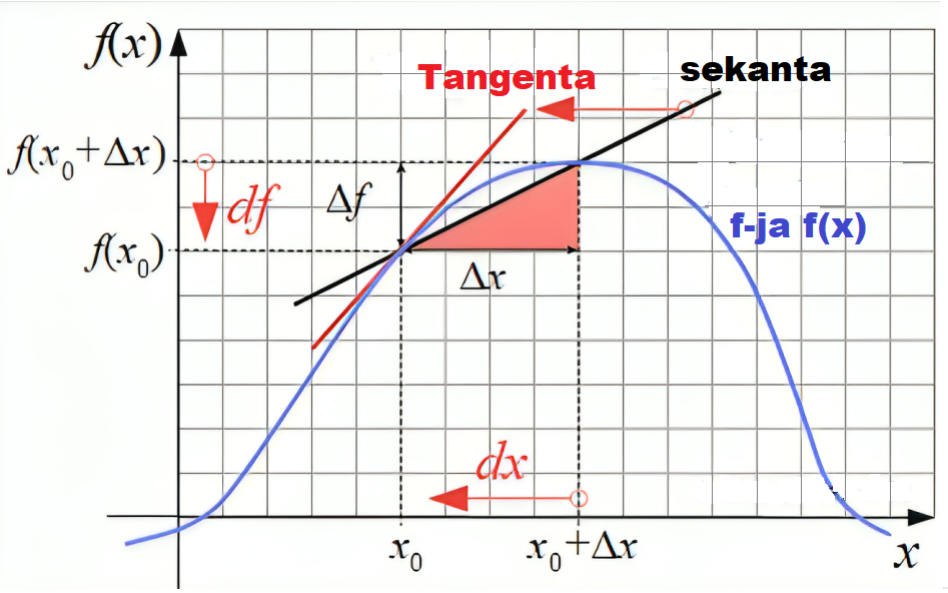
$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}, \tag{2.1}$$

ako postoji.$f'(x)$ još zovemo i prva derivacija funkcije $f(x)$, dok derivacije višeg reda, $f^{(n)}$ definiramo rekurzivno $f^{(n)} = (f^{(n-1)})$.

$f(x)$	$f'(x)$	$f(x)$	$f'(x)$	$f(x)$	$f'(x)$
c	0	$\arcsin x$	$\frac{1}{\sqrt{1-x^2}}$	$\operatorname{sh} x$	$\operatorname{ch} x$
x	1	$\arccos x$	$-\frac{1}{\sqrt{1-x^2}}$	$\operatorname{ch} x$	$\operatorname{sh} x$
x^n	nx^{n-1}	$\arctan x$	$\frac{1}{1+x^2}$	$\operatorname{th} x$	$\frac{1}{\operatorname{ch}^2 x}$
$\sqrt{x}$	$\frac{1}{2\sqrt{x}}$	$\operatorname{arccot} x$	$-\frac{1}{1+x^2}$	$\operatorname{cth} x$	$-\frac{1}{\operatorname{sh}^2 x}$
$\sin x$	$\cos x$	e^x	e^x	$\operatorname{arsh} x$	$\frac{1}{\sqrt{1+x^2}}$
$\cos x$	$-\sin x$	a^x	$a^x \ln a$	$\operatorname{arch} x$	$\frac{1}{\sqrt{x^2-1}}$
$\tan x$	$\frac{1}{\cos^2 x}$	$\ln x$	$\frac{1}{x}$	$\operatorname{arth} x$	$\frac{1}{1-x^2}$
$\cot x$	$-\frac{1}{\sin^2 x}$	$\log_a x$	$\frac{1}{x \ln a}$	$\operatorname{arcth} x$	$-\frac{1}{x^2-1}$

Slika 2.1. Derivacije elementarnih funkcija, Izvor: [14]

Geometrijska interpretacija derivacije je vidljiva na Slici 2.2., dok smo na Slici 2.1. prikazali derivacije elementarnih funkcija.



Slika 2.2. Geometrijska interpretacija derivacije, Izvor: autor

2.2. Diferencijalna jednadžba

Diferencijalne jednadžbe su matematičke jednadžbe koje povezuju nepoznatu funkciju s njezinim derivacijama. Koriste se za modeliranje širokog raspona fizičkih fenomena, od kretanja objekata do ponašanja električnih krugova i širenja bolesti. Diferencijalna jednadžba obično se piše u obliku:

$$f\left(x, y, y', y'', ..., y^{(n)}\right) = 0, \tag{2.2}$$

gdje je y nepoznata funkcija, x je nezavisna varijabla, a f je funkcija koja opisuje odnos između njih. Diferencijalne jednadžbe se mogu klasificirati prema njihovom redu i linearnosti. Red diferencijalne jednadžbe je najviši red derivacije koji se pojavljuje u jednadžbi. Na primjer, diferencijalna jednadžba prvog reda uključuje samo prvu derivaciju od y, dok diferencijalna jednadžba drugog reda uključuje drugu derivaciju od y [1].

Diferencijalna jednadžba je linearna ako se može zapisati u obliku:

$$a_n(x)y^{(n)} + a_{n-1}(x)y^{(n-1)} + \cdots + a_1(x)y' + a_0(x)y = f(x), \tag{2.3}$$

gdje su $a_n(x), a_{n-1}(x), ..., a_1(x), a_0(x)$ i $f(x)$ funkcije od x. Bitno je istaknuti u linearnoj diferencijalnoj jednadžbi, koeficijenti $a_n(x), a_{n-1}(x), ..., a_1(x), a_0(x)$ su konstante ili funkcije samo od x, a ne ovise o y ili njezinim derivacijama [1].

Rješavanje diferencijalne jednadžbe uključuje pronalaženje funkcije y koja zadovoljava jednadžbu. To se može učiniti analitički ili numerički. Analitičke metode uključuju pronalaženje eksplicitne formule za y u odnosu na x, koristeći tehnike poput separacije varijabli, integracije i supstitucije. Numeričke metode uključuju aproksimaciju rješenja diferencijalne jednadžbe upotre-

bom metoda poput Eulerove metode, Runge-Kutta metoda i metoda konačnih elemenata i konačnih razlika.

2.3. Rubni problem

Rubni problemi su klasa problema u matematici koja uključuju pronalaženje rješenja diferencijalne jednadžbe uz zadane rubne uvjete. Pojavljuju se u mnogim područjima znanosti i tehnike, uključujući fiziku, kemiju i financijski sektor.

U rubnom problemu, diferencijalna jednadžba je definirana na nekoj domeni, a rješenjem je potrebno zadovoljiti određene uvjete na granici domene. Ti uvjeti mogu biti definirani u obliku fiksnih vrijednosti rješenja, njegovih derivacija ili njihove kombinacije. Cilj problema je pronaći rješenje koje zadovoljava i diferencijalnu jednadžbu i rubne uvjete [1].

Klasificiraju se u nekoliko vrsta: Dirichletovi rubni problemi, Neumannovi rubni problemi ili mješoviti. U Dirichletovom rubnom problemu, rubni uvjeti određuju vrijednosti rješenja na rubu domene. Dirichletov rubni problem za ODJ prvog reda općenito glasi:

$$\begin{cases} f(x, y, y') = 0 \\ f(a) = y_a, f(b) = y_b \end{cases}. \tag{2.4}$$

Na primjer, raspodjela temperature u metalnoj ploči može se opisati pomoću toplinske jednadžbe, a Dirichletov rubni problem može uključivati zadavanje temperature na rubovima ploče. U Neumannovom rubnom problemu, rubni uvjeti određuju vrijednosti derivacije rješenja na granici domene. Neumannov rubni problem za ODJ 1. reda:

$$\begin{cases} f(x, y, y') = 0 \\ f'(a) = y_a, f'(b) = y_b \end{cases}. \tag{2.5}$$

Na primjer, protok tekućine u cijevi može se opisati pomoću Navier-Stokesove jednadžbe, a Neumannov rubni problem može uključivati zadavanje gradijenta tlaka na krajevima cijevi [4].

Rubni problemi se mogu rješavati analitički ili numerički. Analitička rješenja su moguća za neke jednostavne rubne probleme, ali za većinu problema su potrebne numeričke metode. Numeričke metode uključuju diskretizaciju domene u skup točaka i aproksimaciju rješenja na svakoj točki pomoću metode konačnih razlika ili metode konačnih elemenata. Dobiveni sustav jednadžbi se može riješiti pomoću tehnika linearne algebre, poput Gaussove eliminacije ili faktorizacije matrice [1].

2.4. Aproksimacija funkcije i domena

2.4.1. Taylorova metoda

Taylorova metoda je matematička metoda koja se koristi za aproksimaciju funkcija u nekoj točki. Ona se bazira na Taylorovom razvoju funkcije oko neke točke, koji se sastoji od polinoma koji aproksimira funkciju u nekoj točki i pripadne greške aproksimacije [1].

Taylorov razvoj funkcije $f(x)$ oko točke a dan je sa:

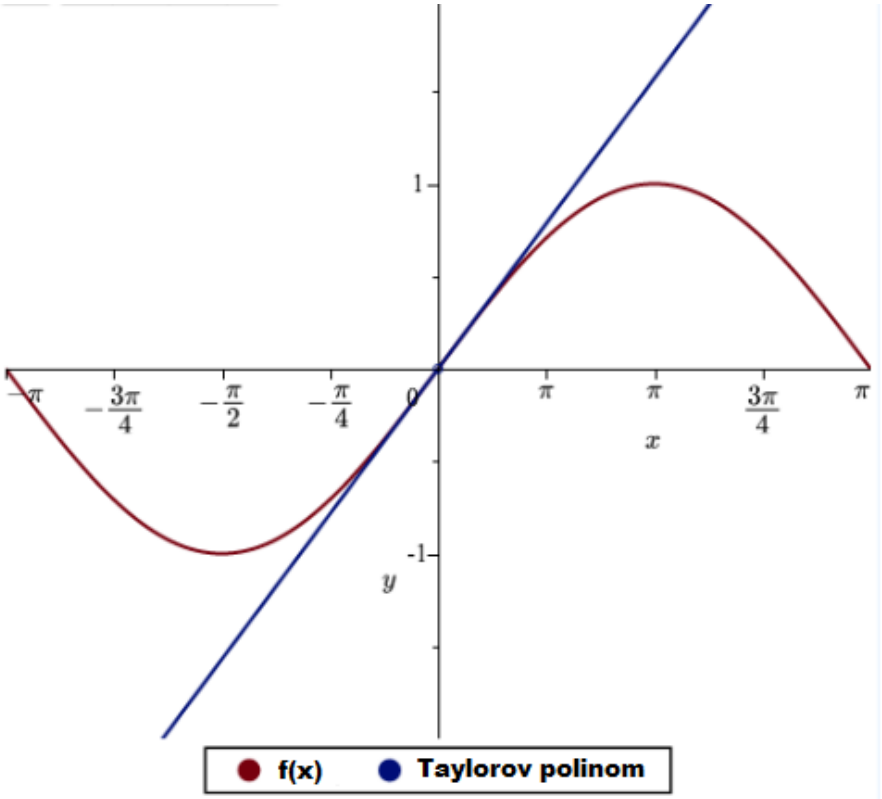
$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^n(a)}{n!}(x-a)^n + R_n(x), \tag{2.6}$$

gdje $R_n(x)$ označava n-ti ostatak Taylorovog razvoja:

$$R_n(x) = \frac{f^{(n+1)}(z_x)}{(n+1)!}(x-a)^{n+1}, \tag{2.7}$$

pri čemu je z_x neki broj između x i a [5].

Taylorova metoda je vrlo precizna za funkcije koje su blizu linearne u točki a, međutim, kako se udaljavamo od točke a preciznost metode se smanjuje. Taylorova metoda također zahtijeva poznavanje viših redova derivacija funkcije koji se koriste, a što može biti teško u nekim slučajevima [5]. Prikazali smo aproksimaciju Taylorovim polinomom na Slici 2.3.



Slika 2.3. Prikaz usporedbe f(x) i aproksimacije Taylorovim polinomom, Izvor: [12]

Metoda konačnih razlika je metoda aproksimacije derivacije funkcije korištenjem konačnih razlika. Derivacija se aproksimira omjerom prirasta funkcije (razlikom funkcijskih vrijednosti) i

pripadnog prirasta nezavisne varijable. Postoji nekoliko metoda konačnih razlika koje ćemo uvesti u nastavku.

2.4.2. Shema prema naprijed

Ova shema aproksimira prvu derivaciju funkcije koristeći razliku između vrijednosti funkcije u dvije susjedne točke. Zove se "shema prema naprijed" jer koristi vrijednost funkcije u trenutnoj točki i sljedećoj točki [2]. Formula za shemu prema naprijed je dana kao:

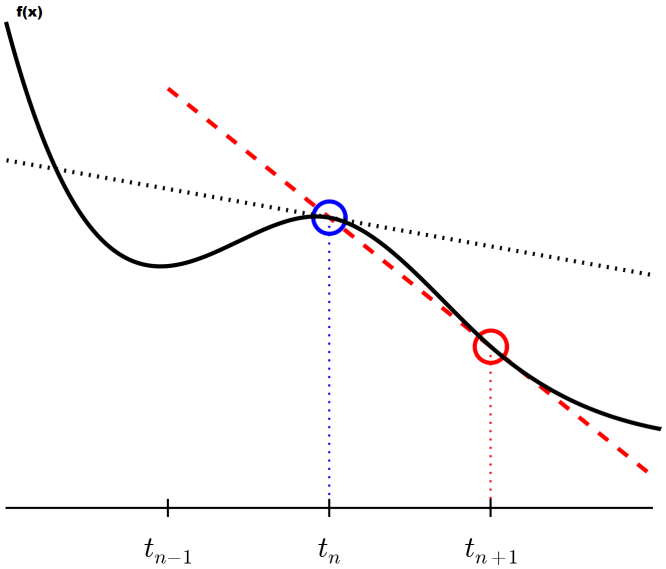
$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \tag{2.8}$$

gdje je $h > 0$ udaljenost izmedu dviju susjednih točaka. Pogreška aproksimacije je proporcionalna h.

Naime, iz Taylorovog razvoja 1. reda imamo vezu s aproksimacijama konačnih razlika:

$$\begin{aligned} f(x) &= f(a) + \frac{f'(a)}{1!}(x-a) + R_1(x), \\ f(a+h) &= f(a) + f'(a)(a+h-a) + R_1, \\ f'(a) &= \frac{f(a+h) - f(a)}{h} - \frac{R_1}{h}, \end{aligned} \tag{2.9}$$

pri čemu je R_1 po (2.7) proporcionalan h^2. Ovo je formula za konačnu razliku prvog reda koja se često koristi za deriviranje funkcija. Vidimo da je ova formula u biti aproksimacija derivacije funkcije pomoću Taylorovog razvoja 1. reda [1]. Na Slici 2.4. se nalazi geometrijska interpretacija sheme unaprijed.



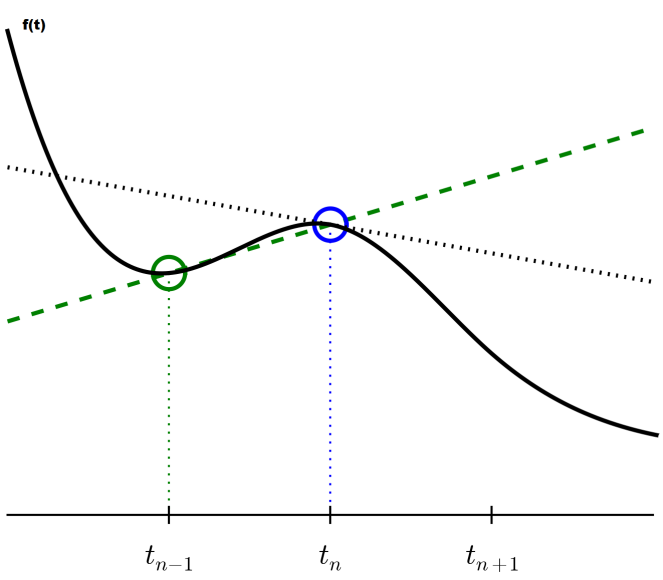
Slika 2.4. Geometrijska interpretacija sheme unaprijed, Izvor: [13]

2.4.3. Shema prema nazad

Ova shema aproksimira prvu derivaciju funkcije koristeći razliku između vrijednosti funkcije u dvije susjedne točke. Zove se "shema prema nazad" jer koristi vrijednost funkcije u trenutnoj točki i prethodnoj točki [2]. Formula za shemu prema nazad je dana kao:

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}, \tag{2.10}$$

gdje je h udaljenost između dviju susjednih točaka. Pogreška u ovoj shemi je također proporcionalna h [1]. Na Slici 2.4. se nalazi geometrijska interpretacija sheme pri kojoj se uzima prethodna točka.



Slika 2.5. Geometrijska interpretacija sheme unazad, Izvor: [13]

2.4.4. Centralna shema

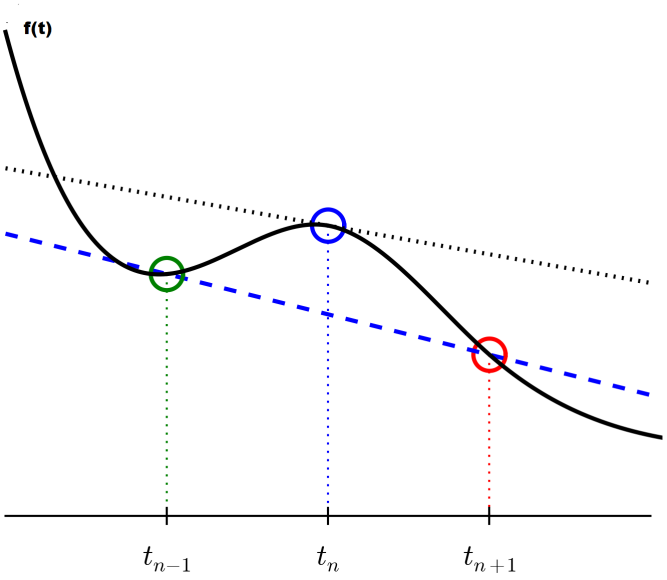
Ova shema aproksimira prvu derivaciju funkcije koristeći razliku između vrijednosti funkcije u dvije točke, jedne smještene "ispred" trenutne točke i jedne smještene "iza" trenutne točke [2]. Zove se "centralna" ili "midpoint" shema jer koristi sredinu između dvije točke, kao što je vidljivo na Slici 2.6., dok smo Slikom 2.7. pokazali sve sheme s pripadnim jednadžbama. Formula za centralnu razliku je dana kao:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \tag{2.11}$$

gdje je h udaljenost izmedu dviju susjednih točaka. Pogreška je proporcionalna h^2 [1].

2.4.5. Aproksimacija druge derivacije

Drugu derivaciju aproksimiramo na jedan od sljedeća tri načina:



Slika 2.6. Geometrijska interpretacija centralne sheme, Izvor: [13]

- konačnim razlikama unaprijed

$$f''(x) \approx \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2}, \tag{2.12}$$

- konačnim razlikama unazad

$$f''(x) \approx \frac{f(x) - 2f(x-h) + f(x-2h)}{h^2}, \tag{2.13}$$

- centralnim konačnim razlikama

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}, \tag{2.14}$$

gdje je h udaljenost između dviju susjednih točaka.

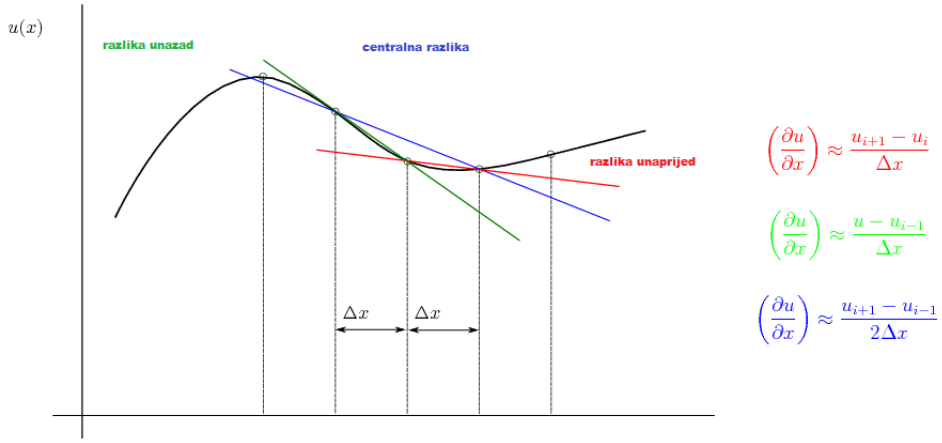
Sada ćemo izvesti formulu (2.14) koristeći izraz za centralne konačne razlike prvog reda (2.11) u točkama $x + \frac{h}{2}$ i $x - \frac{h}{2}$ s korakom $\frac{h}{2}$. Dobivamo

$$f'(x + \frac{h}{2}) \approx \frac{f(x+h) - f(x)}{h}$$
$$f'(x - \frac{h}{2}) \approx \frac{f(x) - f(x-h)}{h}.$$

Sada možemo aproksimirati drugu derivaciju koristeći ove aproksimacije za prvu derivaciju i aproksimaciju centralnom konačnom razlikom prvog reda za drugu derivaciju u točki x s korakom h (2.11):

$$\begin{aligned} f''(x) &\approx \frac{f'(x + \frac{h}{2}) - f'(x - \frac{h}{2})}{h} \\ &\approx \frac{\frac{f(x+h)-f(x)}{h} - \frac{f(x)-f(x-h)}{h}}{h} \\ &= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \end{aligned} \tag{2.15}$$

Red pogreške centralne konačne razlike drugog reda je h^2.

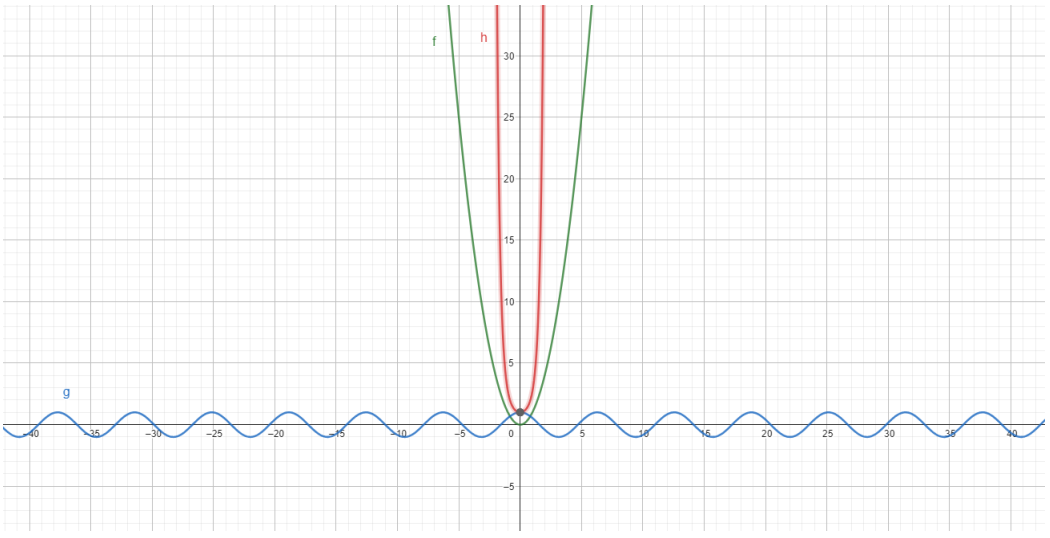


Slika 2.7. Usporedba prethodno navedenih shema konačnih razlika, Izvor: [3]

2.5. Usporedba različitih aproksimacija konačnim razlikama

U ovom poglavlju ćemo pokazati kolika je zapravo točnost aproksimacije, pogreška i eventualna odstupanja između različitih shema i pri različitim aproksimacijskim koracima. Razmotrit ćemo tri različite funkcije:

$$f(x) = x^2,$$
$$g(x) = \cos(x),$$
$$h(x) = e^{x^2}.$$



Slika 2.8. f(x), g(x), h(x), Izvor: autor

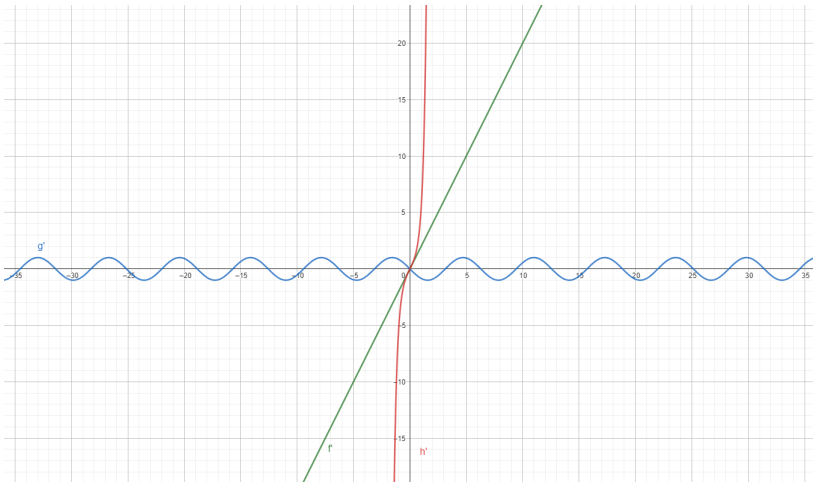
Za svaku od ovih funkcija aproksimirat ćemo vrijednost prve derivacije u točki $x_0 = 2$. S obzirom da su funkcije jednostavne, prvo ćemo ih derivirati analitički kako bismo dobili referentne

vrijednosti za usporedbu s aproksimacijama. Derivacije gore navedenih funkcija su redom:

$$f'(x) = 2x,$$
$$g'(x) = -\sin(x),$$
$$h'(x) = 2xe^{x^2},$$

te u točki $x_0 = 2$, ove derivacije imaju vrijednosti:

$$f'(2) = 4,$$
$$g'(2) = -\sin(2) \approx -0.9093,$$
$$h'(2) = 4e^4 \approx 218.3926.$$



Slika 2.9. f'(x), g'(x), h'(x), Izvor: autor

Na Slici 2.8. su grafički prikazane funkcije $f(x), g(x), h(x)$, dok na Slici 2.9. možemo vidjeti kako se te iste funkcije promijene pri njenim derivacijama.

Sada ćemo primijeniti metodu konačnih razlika kako bismo aproksimirali derivacije ovih funkcija. Koristit ćemo aproksimacije prve derivacije za sve tri sheme (unaprijed, unazad i centralnu) pri različitim vrijednostima koraka $h = 0.1$, $h = 0.01$, $h = 0.001$ i usporediti ih.

$$f'_{unaprijed}(x_0, h) = \frac{f(x_0 + h) - f(x_0)}{h} = \frac{f(2 + h) - f(2)}{h},$$
$$f'_{unazad}(x_0, h) = = \frac{f(x_0) - f(x_0 - h)}{h} = \frac{f(2) - f(2 - h)}{h},$$
$$f'_{centralna}(x_0, h) = \frac{f(x_0 + h) - f(2 - h)}{2h} = \frac{f(2 + h) - f(2 - h)}{2h}.$$

- $f(x) = x^2$, $x_0 = 2$, $f'(x_0) = f'(2) = 4$

  U tablicama 2.1. i 2.2. možemo vidjeti razliku pri različitim koracima i za različite aproksimacije. Iz njih je vidljivo da pri centralnoj aproksimaciji jedino nema pogreške, ni pri

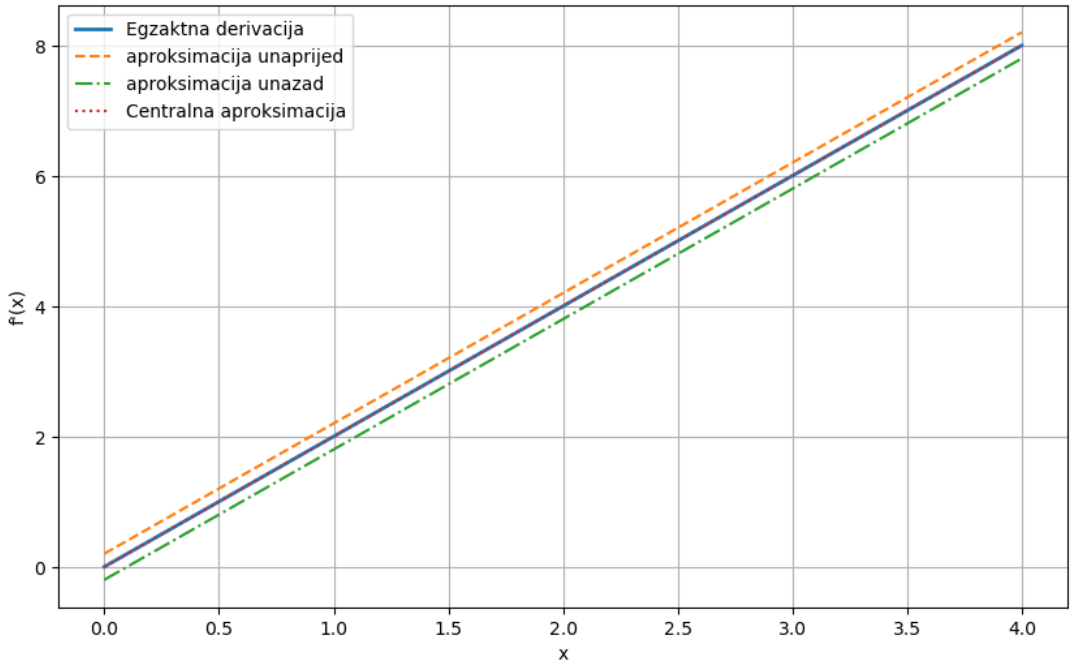
Tablica 2.1. Aproksimacije konačnim razlikama pri različitim koracima za f(x)

Aproksimacija	$h = 0.1$	$h = 0.01$	$h = 0.001$
$f'_{unaprijed}$	4.1	4.01	4.001
f'_{unazad}	3.9	3.99	3.999
$f'_{centralna}$	4	4	4

Tablica 2.2. Apsolutne pogreške pri različitim koracima za f(x)

Apsolutna pogreška	$h = 0.1$	$h = 0.01$	$h = 0.001$
$\lvert f'(2) - f'_{unaprijed}(2,h)\rvert$	0.1	0.01	0.001
$\lvert f'(2) - f'_{unazad}(2,h)\rvert$	0.1	0.01	0.001
$\lvert f'(2) - f'_{centralna}(2,h)\rvert$	0	0	0

različitim koracima ni od egzaktnog rješenja, dok su među ostalim dvjema aproksimacijama pogreške iste, a što im je korak manji, time im je i pogreška manjeg iznosa. Aproksimacije za funkciju $f(x)$ i egzaktne derivacije možemo vidjeti na Slici 2.10. iz čega možemo primijetiti da se centralna aproksimacija poklapa s egzaktnom derivacijom, dok su ostale aproksimacije s istom apsolutnom pogreškom, koja je sve manja, s što većim korakom.



Slika 2.10. Aproksimacija derivacije za $f(x) = x^2$, Izvor: autor

- $g(x) = \cos(x)$, $x_0 = 2$, $g'(x_0) = g'(2) \approx -0.9093$

  U tablicama 2.3. i 2.4. uspoređujemo razliku među aproksimacijama i koracima za funkciju $g(x)$, iz kojih je vidljivo da kod ove funkcije ima pogreške i za centralnu aproksimaciju, ali je svejedno pogodnija od ostale dvije aproksimacije, dok je među njima pogodnija ona za unazad. I u ovom slučaju vrijedi, da je najmanja pogreška za sve tri aproksimacije pri najmanjem koraku.

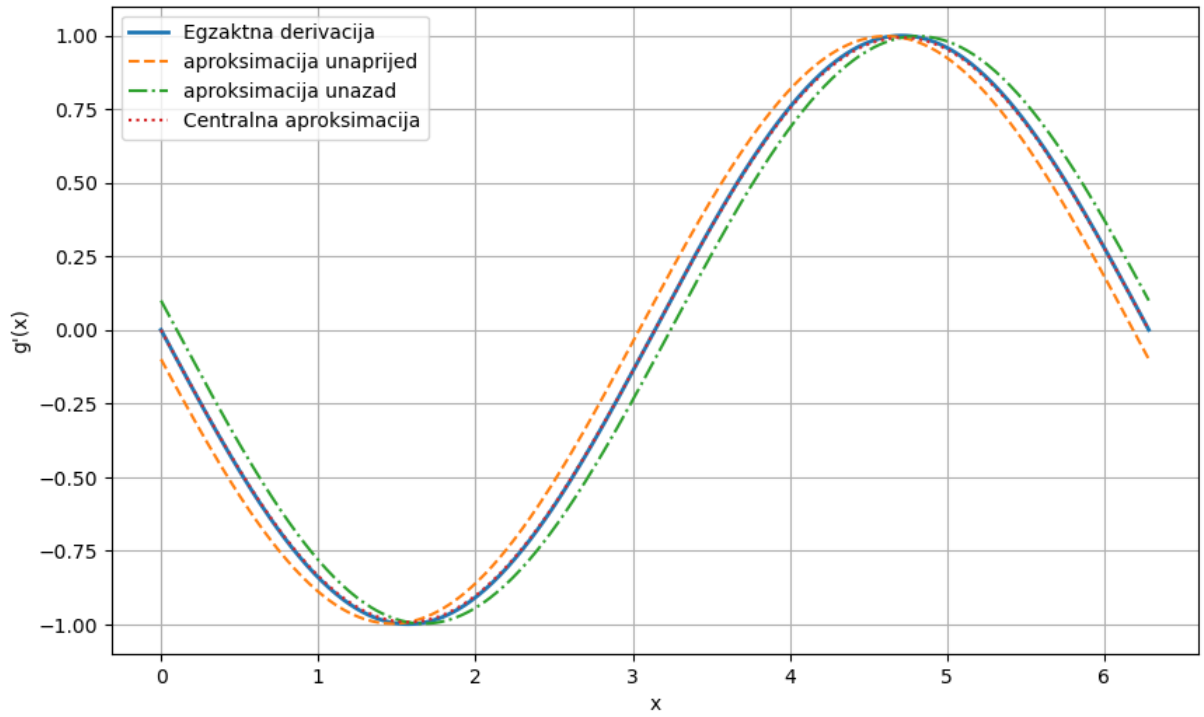
Tablica 2.3. Aproksimacije konačnim razlikama pri različitim koracima za g(x)

Aproksimacija	$h = 0.1$	$h = 0.01$	$h = 0.001$
$g'_{unaprijed}$	-0.88699	-0.90720	-0.90908
g'_{unazad}	-0.92857	-0.91136	-0.90950
$g'_{centralna}$	-0.90778	-0.90928	-0.90929

Tablica 2.4. Apsolutne pogreške pri različitim koracima za g(x)

Apsolutna pogreška	$h = 0.1$	$h = 0.01$	$h = 0.001$
$\lvert g'(2) - g'_{unaprijed}(2,h)\rvert$	0.02231	0.0021	0.00022
$\lvert g'(2) - g'_{unazad}(2,h)\rvert$	0.01927	0.00206	0.0002
$\lvert g'(2) - g'_{centralna}(2,h)\rvert$	0.00152	0.00002	0.00001

Na slici 2.11. smo pokazali koliko se aproksimacije razlikuju od egzaktne derivacije. Također, možemo primijetiti kako pri ekstremima derivirane funkcije $g(x)$ su pogreške aproksimacija minimalne za sve tri aproksimacije, pa se udaljavanjem od ekstrema, pogreške povečavaju.



Slika 2.11. Aproksimacija derivacije za $g(x) = \cos(x)$, Izvor: autor

- $h(x) = e^{x^2}$, $x_0 = 2$, $h'(x_0) = h'(2) \approx 218.3926$

Tablice 2.5. i 2.6. imaju veće oscilacije u pogreškama zbog aproksimacije kompleksnije funkcije $h(x)$. Možemo zaključiti da pri nedovoljno malim korakom, pogreške su puno veće nego za funkcije $f(x)$ i $g(x)$. I u ovom slučaju kao u svakom dosadašnjem, najpogodnija aproksimacija je centralna. Za funkcije ovog tipa je izuzetno bitno uzeti što manji korak da bi dovoljno smanjili

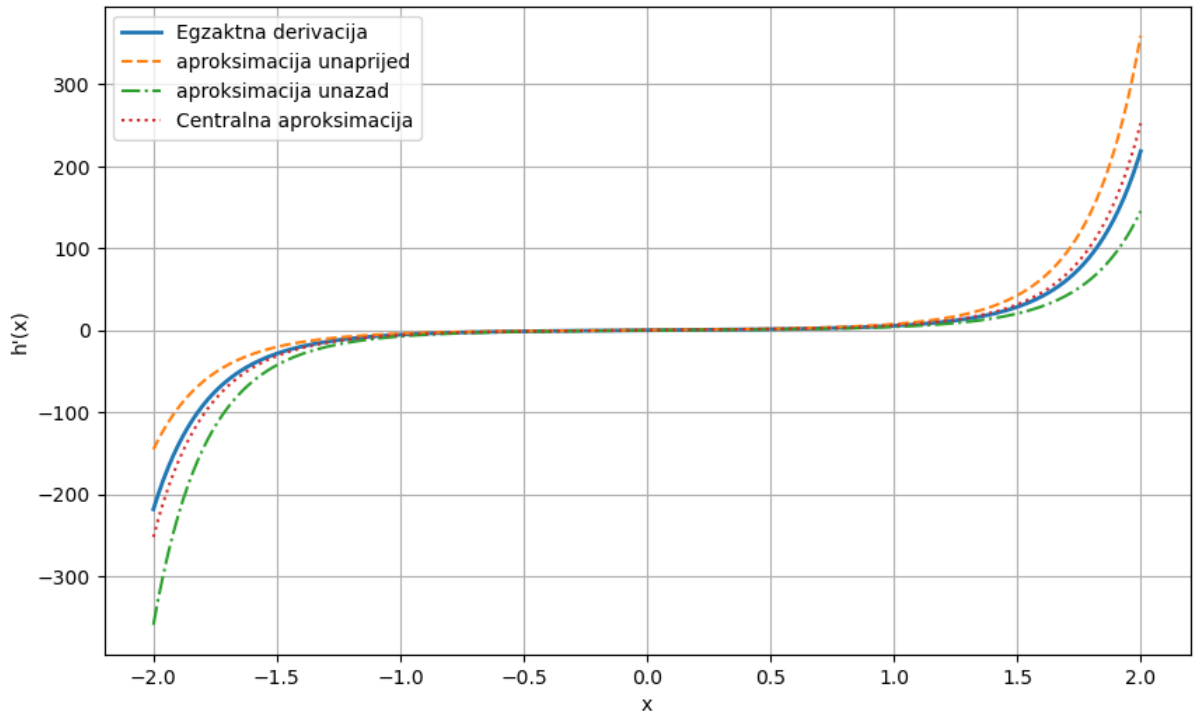
Tablica 2.5. Aproksimacije konačnim razlikama pri različitim koracima za h(x)

Aproksimacija	$h = 0.1$	$h = 0.01$	$h = 0.001$
$h'_{unaprijed}$	276.71313	223.38757	218.88478
h'_{unazad}	176.32097	213.55781	217.90202
$h'_{centralna}$	226.51705	218.47269	218.39340

Tablica 2.6. Apsolutne pogreške pri različitim koracima za h(x)

Apsolutna pogreška	$h = 0.1$	$h = 0.01$	$h = 0.001$
$\lvert h'(2) - h'_{unaprijed}(2,h)\rvert$	58.32053	4.99497	0.49218
$\lvert h'(2) - h'_{unazad}(2,h)\rvert$	42.07163	4.83479	0.49058
$\lvert h'(2) - h'_{centralna}(2,h)\rvert$	8.12445	0.08009	0.0008

pogrešku i time povećali točnost aproksimacije. Usporedba egzaktne derivacije i pripadnih aproksimacija za funkciju $h(x)$ možemo vidjeti na Slici 2.12. gdje možemo primijetiti da pri nekim dijelovima funkcije je relativno mala pogreška u aproksimacijama, dok u nekim se naglo povećava. To se događa zato što je u tim djelovima prirast derivacije, odnosno druga derivacija od h, relativno velik broj ($h''(2) = 18e^4 \approx 982.8$), a vidjeli smo da pogreška aproksimacije ovisi o drugoj derivaciji (vidjeti (2.6)–(2.7), 2.9)).



Slika 2.12. Aproksimacija derivacije za $h(x) = e^{x^2}$, Izvor: autor

Općenito, možemo zaključiti da kvaliteta aproksimacije derivacija metodom konačnih razlika ovisi o funkciji koja se analizira, a također i o veličini koraka h. Za neke funkcije kao što su $f(x) = x^2$ i $g(x) = \cos(x)$, aproksimacije su točnije i apsolutne greške su manje. Međutim, za druge funkcije, kao što je $h(x) = e^{x^2}$, apsolutna pogreška je puno veća naspram one u $f(x)$ i $g(x)$. Razlog je što greška aproksimacije derivacije ovisi o veličini druge derivacije. Potrebno

je koristiti izuzetno male korake kako bi se postigla prihvatljiva točnost, što možemo primijetiti u tablici 2.6. Centralna aproksimacija obično daje najmanje apsolutne pogreške u usporedbi s ostalim aproksimacijama, te je to u skladu s očekivanjima jer smo vidjeli u ranijem poglavlju da je red pogreške centralne konačne razlike proporcionalan h^2, dok je za druge dvije vrste konačnih razlika red pogreške proporcionalan h.

3. Metoda konačnih razlika

Metoda konačnih razlika je jedan od načina numeričkog rješavanja diferencijalnih jednadžbi. Naime, numeričkim metodama aproksimiraju se rješenja jednadžbi, te iako nisu egzaktna ona uvelike olakšavaju rješavanje neke diferencijalne jednadžbe. U suštini, derivacije u diferencijalnoj jednadžbi zamijenimo s aproksimiranim izrazima čime dobivamo običnu algebarsku jednadžbu koju je znatno lakše riješiti. Rezultat rješavanja te algebarske jednadžbe su aproksimirana rješenja [1].

Uzmimo za primjer jednostavnu običnu diferencijalnu jednadžbu:

$$y'(x) = x, \tag{3.1}$$

čije je opće rješenje:

$$y = \int x dx = \frac{x^2}{2} + C, \tag{3.2}$$

gdje je C neodređena konstanta.

Međutim, nas ne zanima rješavanje ove jednadžbe integriranjem, već ju želimo riješiti numerički. Jednadžba poput ove je kontinuirani problem, a da bi ju riješili numerički, potrebno ju je pretvoriti u diskretan problem tako da uzmemo konačan broj vrijednosti y i derivaciju zamijenimo aproksimiranim izrazima, specifično, konačnom razlikom unaprijed:

$$\frac{dy}{dx}(x_i) \approx \frac{y_{i+1} - y_i}{h}, \tag{3.3}$$

gdje je $y_i \approx y(x_i)$, $x_0 < x_1 < ... < x_n$ ekvidistantne točke, te $h = x_i - x_{i-1}$. Na taj način dobijemo sljedeću aproksimativnu jednadžbu [3]:

$$\frac{y_{i+1} - y_i}{h} = x_i. \tag{3.4}$$

Preuređivanjem jednadžbe (3.4) dobivamo sljedeći izraz za y_{i+1}

$$y_{i+1} = xh + y_i, \tag{3.5}$$

i sada možemo izračunati svaki y_{i+1} s pretpostavkom da nam je poznato y_i.

Za što točniju aproksimaciju želimo uzeti što više podintervala, i što manji korak. Na taj način, smanjujemo pogrešku te dobivamo rješenje koje je približno jednako egzaktnom rješenju. Međutim, povećanjem broja podintervala ne smanjujemo samo moguću pogrešku, nego i povećavamo broj jednadžbi koje će biti potrebno riješiti.

3.1. Primjeri implementacije metode konačnih razlika na rubne probleme

3.1.1. Primjer 1

Zadan je rubni problem:

$$y'' + y = x, \quad y(0) = 1, \quad y\left(\frac{\pi}{2}\right) = \frac{\pi}{2} - 1. \tag{3.6}$$

Riješit ćemo ovaj problem metodom diskretizacije, odnosno konačne razlike uzevši da je $m = 5$ te ćemo usporediti dobivena rješenja s egzaktnim rješenjem.

Da bismo riješili ovaj problem koristeći metodu konačnih razlika, prvo moramo diskretizirati domenu $[0, \pi/2]$. Ako postavimo da veličina pomaka bude $h = \frac{\pi/2}{m}$, gdje je m broj podintervala. Definirat ćemo čvorove kao $x_i = ih$ za $i = 0, 1, \ldots, m$. Koristeći aproksimaciju za drugu derivaciju y'' i prve derivacije y', dobit ćemo sljedeću jednadžbu:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + y_i = x_i \quad \text{za } i = 1, 2, \ldots, m, \tag{3.7}$$

gdje je y_i aproksimacija $y(x_i)$. Preoblikovat ćemo jednadžbu da dobijemo y_{i+1}:

$$y_{i+1} = 2y_i - y_{i-1} + h^2(x_i - y_i) \quad \text{za } i = 0, 1, \ldots, m - 1. \tag{3.8}$$

Možemo koristiti ovu jednadžbu za izračun vrijednosti y_i za $i = 0, 1, \ldots, m$ primjenom rubnih uvjeta $y_0 = 1$ i $y_m = \frac{\pi}{2} - 1$.

$$y_{i+1} = 2y_i - y_{i-1} + h^2(x_i - y_i), \quad i = 1, \ldots, 4. \tag{3.9}$$

Za $i = 2$, imamo:

$$y_3 = 2y_2 - y_1 + h^2(x_2 - y_2) = 2y_2 - y_1 + (0.31416)^2(0.62832 - y_2), \tag{3.10}$$

Za $i = 3$:

$$y_4 = 2y_3 - y_2 + h^2(x_3 - y_3) = 2y_3 - y_2 + (0.31416)^2(0.94248 - y_3), \tag{3.11}$$

Konačno, za $i = 4$:

$$y_5 = 2y_4 - y_3 + h^2(x_4 - y_4) = 2y_4 - y_3 + (0.31416)^2(1.25664 - y_4). \tag{3.12}$$

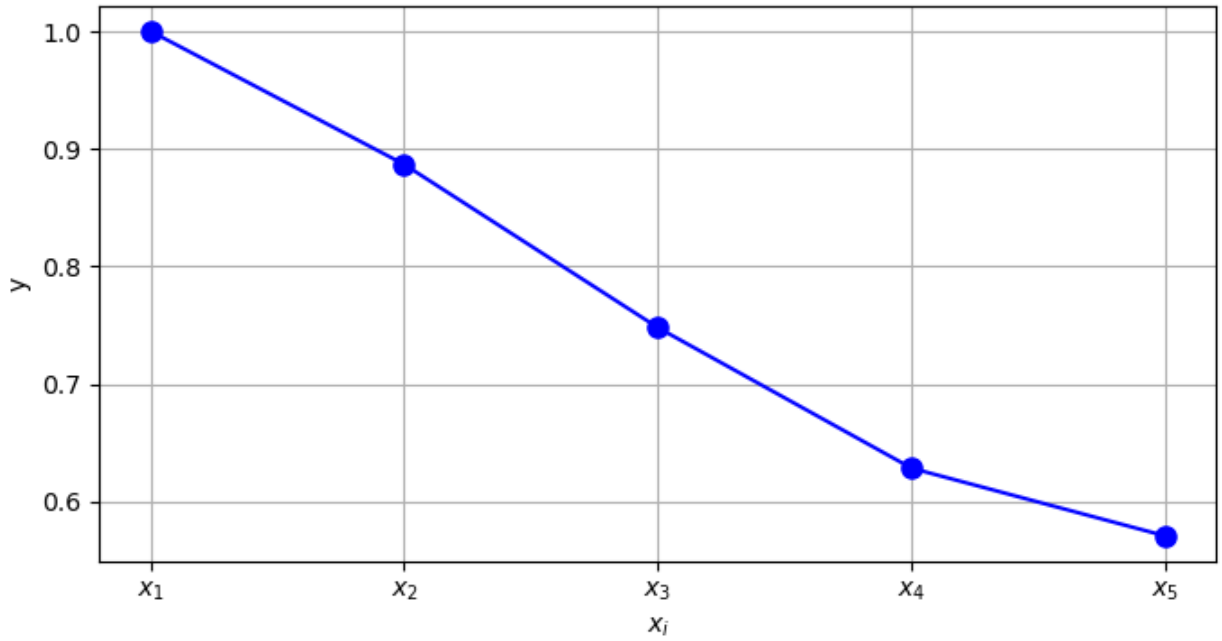
Dobivamo sustav od pet jednadžbi i pet nepoznanica koje možemo riješiti. Naime, da si olakšamo rješavanje, koristit ćemo matrice:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -1.9011 & 1 & 0 & 0 \\ 0 & 1 & -1.9011 & 1 & 0 \\ 0 & 0 & 1 & -1.9011 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.0621 \\ 0.0933 \\ 0.1241 \\ \frac{\pi}{2} - 1 \end{pmatrix}.$$

Rješavanjem dobivenog linearnog sustava slijedi:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.8867 \\ 0.7479 \\ 0.6284 \\ \frac{\pi}{2} - 1 \end{pmatrix}.$$

Na sljedećoj slici prikazat ćemo dobiveno rješenje grafički.



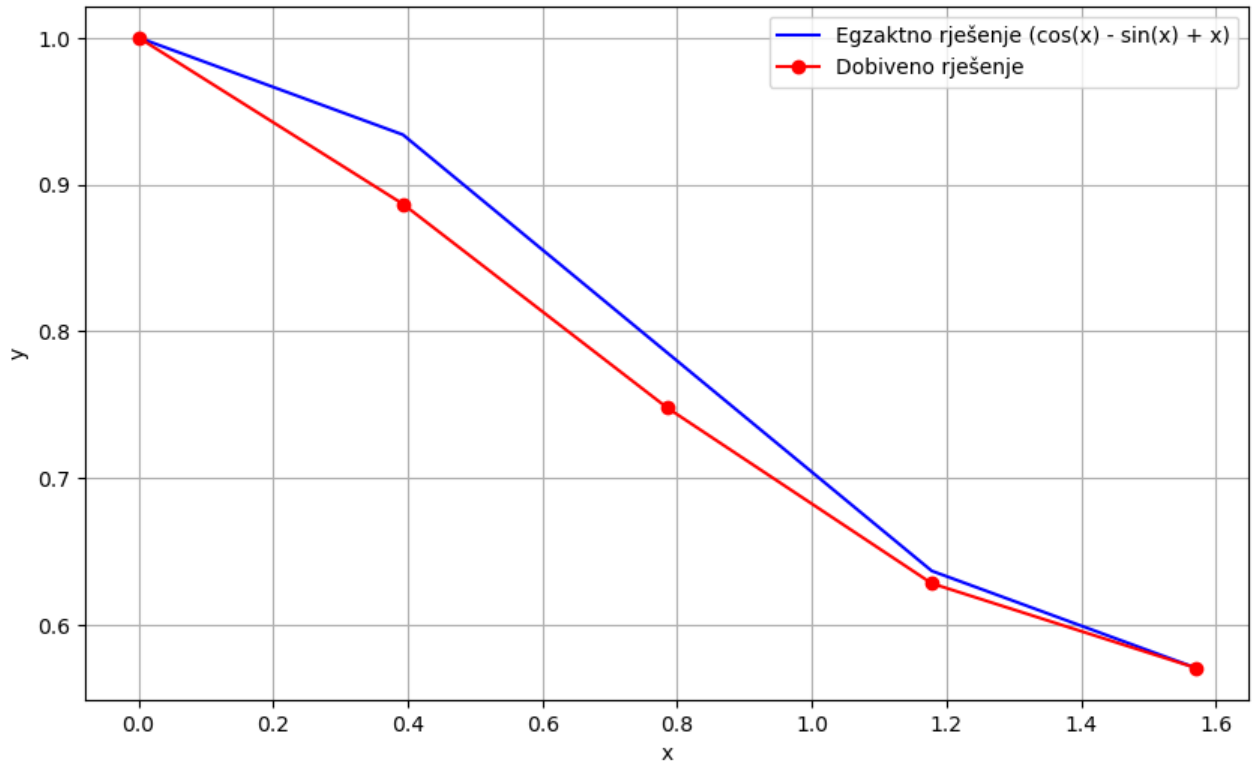
Slika 3.1. Graf dobivenog rješenja, Izvor: autor

Egzaktno rješenje diferencijalne jednadžbe je $y(x) = \cos(x) - \sin(x) + x$. Kako su dane vrijednosti za y_i za $i = 1, 2, 3, 4, 5$, apsolutna pogreška za svaki od i-tih koraka može se izračunati kao $|y_i - y(x_i)|$. Uzimajući vrijednosti za y_i iz prethodnog rješenja, dobivamo:

$$\begin{aligned}
|y_1 - y(x_1)| &= |1 - (\cos(0) - \sin(0) + 0)| = 0, \\
|y_2 - y(x_2)| &= |0.8867 - (\cos(0.62832) - \sin(0.62832) + 0.62832)| \approx 0.03715, \\
|y_3 - y(x_3)| &= |0.7479 - (\cos(0.94248) - \sin(0.94248) + 0.94248)| \approx 0.02665, \\
|y_4 - y(x_4)| &= |0.6284 - (\cos(1.25664) - \sin(1.25664) + 1.25664)| \approx 0.01380, \\
|y_5 - y(x_5)| &= \left|\frac{\pi}{2} - 1 - \left(\cos(\frac{\pi}{2}) - \sin(\frac{\pi}{2}) + \frac{\pi}{2}\right)\right| = 0.
\end{aligned}$$

Vidimo da su apsolutne pogreške male što ukazuje na to da je numeričko rješavanje metodom konačnih razlika dobro aproksimiralo egzaktno rješenje.

Slika 3.2. nam daje predodžbu koliko se aproksimirana rješenja razlikuju od egzaktnih. Kao što smo prije naveli, povećanjem broja podintervala i smanjenjem koraka h pogreška postaje sve manja.



Slika 3.2. Usporedba egzaktnog rješenja i dobivenog rješenja, Izvor: autor

3.1.2. Primjer 2

Pokazat ćemo metodu konačnih razlika na još jednom primjeru ODJ:

$$y'' + xy' - xy = 2x, \tag{3.13}$$

s rubnim uvjetima $y(0) = 1$ i $y(2) = 8$.

Prvi korak u rješavanju ovog problema jest diskretizacija kontinuirane domene, tj. dijeljenje na diskretne segmente. Nakon toga se odabire veličina koraka, h, koji definira diskretne točke u domeni. Na primjer, odabirom $h = 0.5$, dobivamo četiri segmenta i pet diskretnih točaka: $x_1 = 0$, $x_2 = 0.5$, $x_3 = 1$, $x_4 = 1.5$, $x_5 = 2$.

Cilj je pronaći približne vrijednosti funkcije y u svakoj od ovih točaka, što znači da imamo pet nepoznanica i potrebno je pet jednadžbi kako bismo ih riješili. Da bi dobili te jednadžbe, koristit ćemo zadanu ODJ i zamijeniti derivacije u njoj s aproksimacijama konačnih razlika.

S obzirom na to da imamo točke ili čvorove u lokacijama koje su udaljene h, možemo razmotriti točku x_i, gdje je $y(x_i) = y_i$, $y(x + h) = y(x_{i+1}) = y_{i+1}$, i $y(x_i - h) = y(x_{i-1}) = y_{i-1}$.

Kako bismo to postigli, slijedimo nekoliko koraka:

1. Zamijenimo egzaktne derivacije u zadanoj ODJ aproksimacijama konačnih razlika i primijenimo jednadžbu na određenoj lokaciji (x_i, y_i).

U našem primjeru, to možemo zapisati kao:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + x_i \frac{y_{i+1} - y_{i-1}}{2h} - x_i y_i = 2x_i. \tag{3.14}$$

2. Sljedeće što možemo napraviti je preoblikovati kako nam jednadžba 3.14 izgleda:

$$y_{y-1}(1 - x_i \frac{h}{2}) + y_i(-2 - h^2 x_i) - y_{i+1}(1 + x_i \frac{h}{2}) = 2x_i h^2. \tag{3.15}$$

Koristeći jednadžbu (3.15), možemo dobiti jednadžbu za svaku unutarnju točku u domeni. Što se tiče prve i posljednje točke, odnosno rubne točke, već imamo za njih jednadžbe zahvaljujući danim rubnim uvjetima.

3. U idućem koraku cilj nam je sastaviti sustav linearnih jednadžbi.

Primjenjujući rekurzivnu formulu na unutarnje točke i koristeći rubne uvjete za već definirane točke, možemo uspostaviti sustav linearnih jednadžbi:

$$\begin{aligned} y_1 &= 1, \\ 0.875y_1 - 2.125y_2 - 1.125y_3 &= 0.25, \\ 0.75y_2 - 2.25y_3 + 1.25y_4 &= 0.5, \\ 0.625y_3 - 2.375y_4 + 1.375y_5 &= 0.75, \\ y_5 &= 8. \end{aligned} \tag{3.16}$$

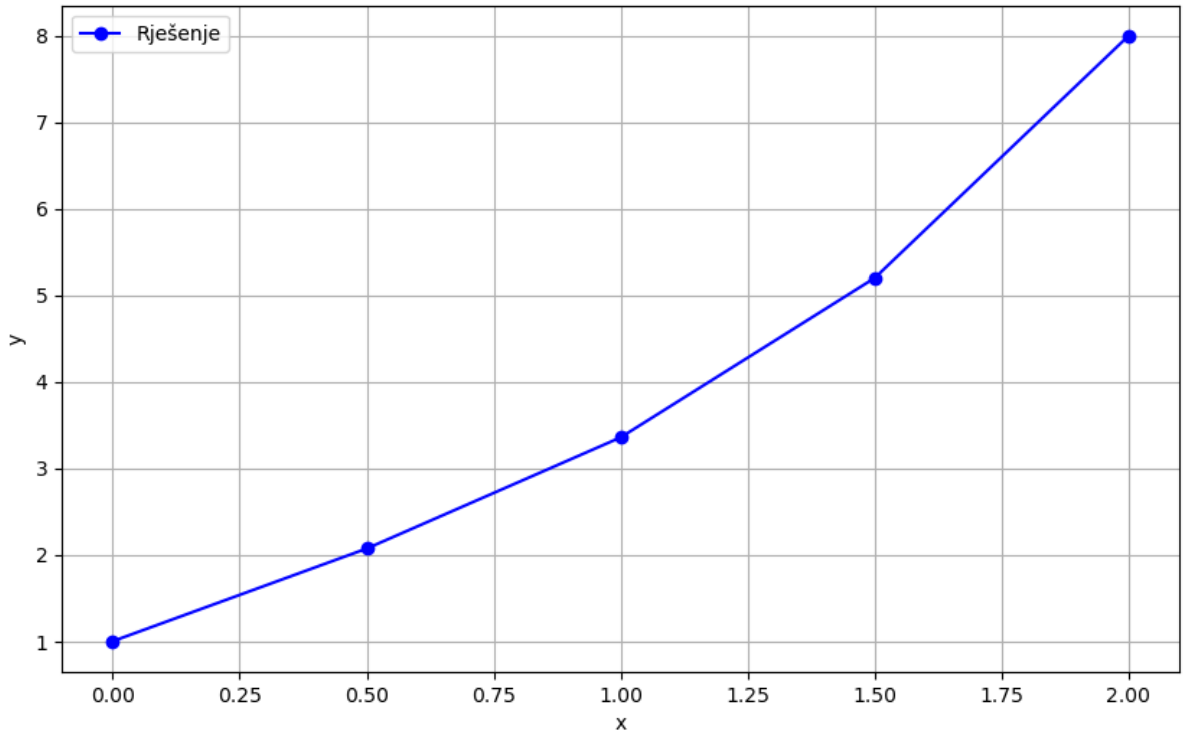
Dobiveni sustav se sastoji od pet jednadžbi s pet nepoznanica, koji se može riješiti. Sustav zapisan matrično glasi:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.875 & -2.125 & 1.125 & 0 & 0 \\ 0 & 0.75 & -2.25 & 1.25 & 0 \\ 0 & 0 & 0.625 & -2.375 & 1.375 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.25 \\ 0.5 \\ 0.75 \\ 8 \end{pmatrix}.$$

4. Sada nam samo preostaje riješiti linearni sustav jednadžbi pomoću matrica, dobiveno rješenje je:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 2.0711 \\ 3.3565 \\ 5.1991 \\ 8 \end{pmatrix}.$$

Dobivena matrična rješenja možemo plotirati na grafu da dobijemo ideju o njenim kretanjima, što smo i uradili kao što je vidljivo na Slici 3.3.



Slika 3.3. Graf dobivenog rješenja, Izvor: autor

3.1.3. Tridijagonalni sustavi linearnih jednadžbi

Pri rješavanju linearnih sustava u kontekstu metode konačnih razlika, dobiveni sustav je tridijagonalan, odnosno ima ne-nula elemente eventualno samo na glavnoj i dvjema susjednim sporednim dijagonalama. Iz tog razloga, kod primjene metode konačnih razlika, često se koriste algoritmi specijalno dizajnirani za rješavanje tridijagonalnih sustava.

Sustav algebarskih jednadžbi koji dobijemo prilikom primjene metode konačnih razlika se može zapisati u obliku matrice A i vektora b:

$$Ax = b. \tag{3.17}$$

Ovdje je A kvadratna matrica koja sadrži koeficijente konačnih razlika, x je vektor nepoznatih vrijednosti koje želimo odrediti, a b je poznati vektor s desne strane. Cilj je pronaći vektor x koji zadovoljava jednadžbu i daje rješenje problema koji se modelira [9].

Rješavanje tridijagonalnih sustava može biti posebno efikasno jer omogućava primjenu posebnih algoritama koji iskorištavaju strukturu tridijagonalnih matrica. Jedan od takvih algoritama je Thomasov algoritam, koji se često koristi za rješavanje tridijagonalnih sustava s linearnom vremenskom složenošću [10].

Matematički, takav tridijagonalni sustav može se zapisati kao:

$$\begin{gathered} b_1 x_1 + c_1 x_2 = d_1, \\ \dots \\ a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \\ \dots \\ a_N x_{N-1} + b_N x_N = d_N, \\ i = 1, 2, \dots, N, \quad a_1 = c_N = 0. \end{gathered} \tag{3.18}$$

Alternativno, tridijagonalni sustav se može prikladnije prikazati u obliku matrice:

$$\begin{bmatrix} b_1 & a_2 & 0 & \cdots & 0 \\ c_1 & b_2 & c_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{N-1} & b_{N-1} & a_N \\ 0 & \cdots & 0 & c_{N-1} & b_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N-1} \\ d_N \end{bmatrix}. \tag{3.19}$$

Linearni algebarski sustav jednadžbi može se riješiti pomoću Gaussove eliminacije, koja ima pojednostavljeni oblik za tridijagonalne matrice i često se naziva Thomasov algoritam. Eliminacijski korak Thomasovog algoritma uključuje eliminaciju elemenata ispod glavne dijagonale dodavanjem dviju uzastopnih redaka pomnoženih odgovarajućim skaliranim faktorima, počevši od prvog retka odozgo. Nakon završetka eliminacijskog koraka, vrijednost zadnje nepoznanice, x_N, može se izračunati kao $x_N := \frac{d_N}{b_N}$.

U jednadžbi za x_{N-1} neposredno iznad zadnje jednadžbe, postoje samo dvije nepoznanice, x_N i x_{N-1}. Budući da je x_N već izračunat u eliminacijskom koraku, njegova vrijednost se može supstituirati u ovu jednadžbu kako bismo odredili x_{N-1}. Taj postupak se nastavlja supstitucijom izračunatih vrijednosti unatrag u prethodne jednadžbe. Prijelazom na jednadžbu za x_{N-2}, postoje samo dvije nepoznanice, x_{N-2} i x_{N-1}, a budući da je x_{N-1} već izračunat, može se supstituirati u tu jednadžbu kako bismo odredili x_{N-2}. Ovaj postupak supstitucije unatrag ponavlja se sve dok se ne odrede sve nepoznate vrijednosti tridijagonalnog sustava jednadžbi.

3.2. Metoda konačnih razlika u Pythonu

Python je visoko razvijani programski jezik općenito poznat po svojoj jednostavnosti, čitljivosti i fleksibilnosti. Python je interpretirani jezik visoke razine koji omogućava brzo pisanje koda i olakšava rad s različitim tipovima podataka. Njegova sintaksa je čista i intuitivna, što ga čini pogodnim za početnike, ali također i za iskusne programere. Python je popularan u raznim područjima, uključujući web razvoj, znanstveno istraživanje, analizu podataka i umjetnu inteligenciju, te ima veliku i aktivnu zajednicu koja podržava razvoj i širenje jezika.

Rješavanje tridijagonalnih sustava je važan problem u matematici i numeričkoj analizi. Tridijagonalni sustav sastoji se od jednadžbi u kojima postoje tri dijagonalne linije koje sadrže koeficijente. Python nudi razne metode za rješavanje tridijagonalnih sustava. Jedan od pristupa je korištenje Thomasovog algoritma. U Pythonu postoji implementacija Thomasovog algoritma koja omogućava brzo rješavanje tridijagonalnih sustava. Osim toga, Python također pruža mogućnosti za numeričku analizu i manipulaciju matrica, što olakšava rješavanje složenih sustava jednadžbi.

```python
import numpy as np

def rjesavanje_tridijagonalnih_sistema(a, b, c, d):
    n = len(d)
    c_dash = np.zeros(n)
    d_dash = np.zeros(n)
    x = np.zeros(n)

    c_dash[0] = c[0] / b[0]
    d_dash[0] = d[0] / b[0]

    for i in range(1, n):
        c_dash[i] = c[i] / (b[i] - a[i] * c_dash[i - 1])
        d_dash[i] = (d[i] - a[i] * d_dash[i - 1]) / (b[i] - a[i] * c_dash[i - 1])

    x[n - 1] = d_dash[n - 1]

    for i in range(n - 2, -1, -1):
        x[i] = d_dash[i] - c_dash[i] * x[i + 1]

    return x

# Primjer:
a = [1, 2, 3]  # Dornja dijagonala
b = [4, 5, 6]  # glavna dijagonala
c = [7, 8, 9]  # gornja dijagonala
d = [10, 11, 12]  # desna strana

solution = rjesavanje_tridijagonalnih_sistema(a, b, c, d)
print("Solution:", solution)
```

```
Solution: [-4.5  4.  -0. ]
```

Slika 3.4. Kod u Pythonu za tridijagonalne sustave, Izvor: autor

Na Slici 3.4. Prikazana je implementacija metode za rješavanje tridijagonalnih sustava. U nastavku se nalazi objašnjenje oznaka u kodu.

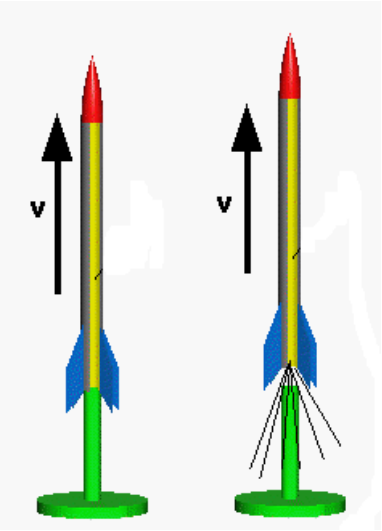
- a, b, c, d: Ovo su liste koje predstavljaju koeficijente tridijagonalnog sustava $Ax = b$. Svaka lista odgovara jednoj dijagonali matrice A, pri čemu a predstavlja donju dijagonalu, b glavnu dijagonalu, a c gornju dijagonalu. Lista d predstavlja desnu stranu sustava.
- n: Ova varijabla sprema veličinu tridijagonalnog sustava, odnosno duljinu liste d.
- c_dash, d_dash, x: To su privremene liste koje se koriste u algoritmu za pohranu privremenih vrijednosti.

- $c\_dash[0]$ i $d\_dash[0]$: Ovo su početne vrijednosti za prve elemente lista c_dash i d_dash. Izračunavaju se na temelju prvih elemenata lista c i d, podijeljenih s odgovarajućim elementom liste b.
- Prva petlja od 1 do n izračunava vrijednosti c_dash i d_dash za preostale elemente. Te se vrijednosti određuju na temelju trenutnog elementa liste c, prethodnog elementa c_dash, liste a, b i prethodnog elementa d_dash. Ova petlja efektivno izvodi "forward sweep" (progresivno kretanje) Thomasova algoritma.
- $x[n-1]$ dodjeljuje vrijednost $d\_dash[n-1]$, što je posljednji element vektora rješenja x.
- Druga petlja se kreće unatrag od $n-2$ do 0 i izračunava preostale elemente vektora rješenja x koristeći vrijednosti d_dash i c_dash. Ova petlja izvodi "back substitution" (supstituciju unatrag) korak Thomasova algoritma.
- Na kraju, funkcija vraća vektor rješenja x.

Thomasov algoritam je učinkovita metoda za rješavanje tridijagonalnih sustava, a ovaj kod ga implementira u Pythonu. Izbjegava potrebu za inverzijom matrice i smanjuje računalnu složenost u usporedbi s drugim metodama za računanje linearnih sustava.

3.2.1. Primjena konačnih razlika kod modeliranja vertikalnog hica

Imamo raketu koju lansiramo, a želimo da se nalazi 50 metara iznad tla nakon 5 sekundi od lansiranja. Pretpostavljamo da nema otpora zraka. Zanima nas koja bi trebala biti početna brzina pri lansiranju rakete [11].



Slika 3.5. Raketa koja se vertikalno lansira nepoznatom početnom brzinom, Izvor: [11]

Da bismo riješili ovaj zadatak, možemo primijeniti metodu konačnih razlika na drugu derivaciju promjene visine rakete u odnosu na vrijeme. Ovu drugu derivaciju možemo aproksimirati

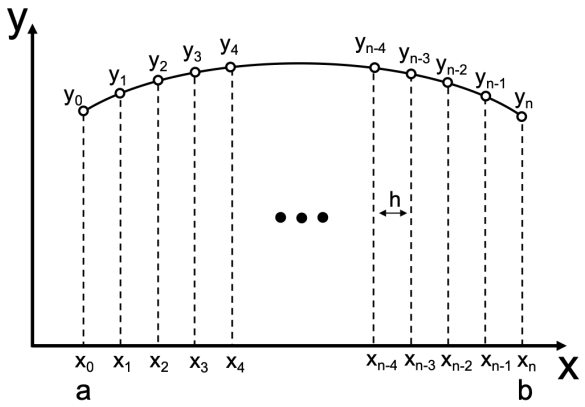
koristeći aproksimacije konačnim razlikama. Da bismo odgovorili na ovo pitanje, možemo oblikovati problem kao rubni problem za ODJ drugog reda. ODJ je sljedeća:

$$y'' = -g. \tag{3.20}$$

Pripadni rubni uvjeti glase $y(0) = 0$ i $y(5) = 50$. Želimo odgovoriti na pitanje, koja je vrijednost $y'(0)$ pri lansiranju.
Ovo je prilično jednostavno pitanje koje možemo riješiti analitički, i to s točnim odgovorom $y'(0) = 34.5$, no mi ćemo to riješiti metodom konačnih razlika, i to pomoću programskog alata Pythona.

Na Slici 3.5. se nalazi karikatura rakete koju želimo lansirati vertikalno u zrak za koju tražimo početnu brzinu kojom bi uspješno lansirali poštivajući uvjet da postigne visinu od 50m nakon 5s. Slika 3.6. predstavlja diskretiziranu mrežu na n jednakih podintervala, udaljene za korak h među susjednim točkama.

Zamjenom druge derivacije sa centralnom konačnom razlikom dobivamo našu željenu algebarsku jednadžbu koju ćemo jednostavnije riješiti.



Slika 3.6. Interval $[a, b]$ na n jednakih podintervala duljine h, Izvor: [11]

S obzirom da je vremenski interval $[0, 5]$, uzmemo da je $n = 10$, tada je $h = 0.5$. Koristeći aproksimirane derivacije konačnih razlika, imamo sljedeće:

$$\begin{gathered} y_0 = 0, \\ y_{i-1} - 2y_i + y_{i+1} = -gh^2, i = 1, 2, ..., n-1, \\ y_{10} = 50. \end{gathered} \tag{3.21}$$

U matričnom zapisu dobijemo:

$$\begin{pmatrix} 1 & 0 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2 & 1 & \\ & & & & 1 & \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \dots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} 0 \\ -gh^2 \\ \dots \\ -gh^2 \\ 50 \end{pmatrix}.$$

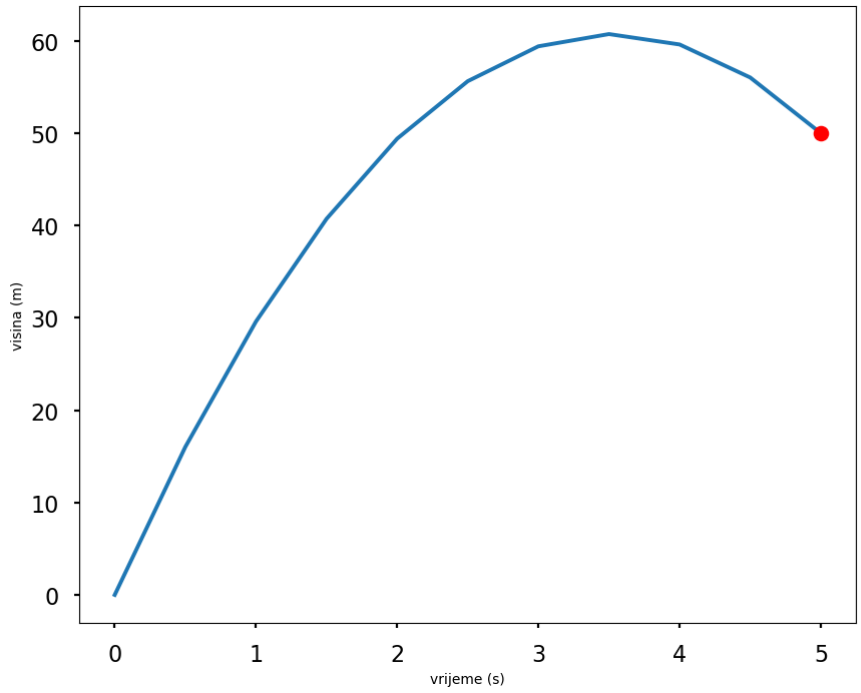
Stoga imamo ukupno 11 jednadžbi u sustavu koje možemo riješiti primjenom odgovarajuće metode.

Na Slici 3.7. prikazan je kod koji uključuje unošenje potrebnih biblioteka i postavljanje potrebnih parametara za prikazivanje grafova. Zatim definirat ćemo varijable i konstruirati matricu A koristeći aproksimacije konačnim razlikama. Također, kod stvara vektor b na temelju određenih uvjeta. Linearne jednadžbe definirane s matricom A i vektorom b rješavaju se pomoću matematičkih operacija, rezultirajući vektorom rješenja y. Također, kod generira vremenski vektor t i prikazuje vrijednosti visine u odnosu na vrijeme. Konačno, graf se prikazuje, pružajući vizualni prikaz visine tijekom vremena. Na Slici 3.8. se nalazi grafički prikaz rješenja.

```python
import numpy as np
import matplotlib.pyplot as plt

def rjesavanje_tridijagonalnih_sustava(A, b):
    n = len(b)
    c_prime = np.zeros(n - 1)
    d_prime = np.zeros(n)
    y = np.zeros(n)

    c_prime[0] = A[0, 1] / A[0, 0]
    d_prime[0] = b[0] / A[0, 0]

    for i in range(1, n - 1):
        c_prime[i] = A[i, i + 1] / (A[i, i] - A[i, i - 1] * c_prime[i - 1])

    for i in range(1, n):
        d_prime[i] = (b[i] - A[i, i - 1] * d_prime[i - 1]) / (A[i, i] - A[i, i - 1] * c_prime[i - 1])

    y[n - 1] = d_prime[n - 1]
    for i in range(n - 2, -1, -1):
        y[i] = d_prime[i] - c_prime[i] * y[i + 1]

    return y
n = 10
h = (5 - 0) / n
A = np.zeros((n + 1, n + 1))
A[0, 0] = 1
A[n, n] = 1
for i in range(1, n):
    A[i, i - 1] = 1
    A[i, i] = -2
    A[i, i + 1] = 1

b = np.zeros(n + 1)
b[1:-1] = -9.8 * h**2
b[-1] = 50
# rješenje
y = rjesavanje_tridijagonalnih_sustava(A, b)
t = np.linspace(0, 5, 11)

plt.figure(figsize=(10, 8))
plt.plot(t, y)
plt.plot(5, 50, 'ro')
plt.xlabel('vrijeme (s)')
plt.ylabel('visina (m)')
plt.grid(True)
plt.show()
```

Slika 3.7. Python kod za rješavanje rubnog problema, Izvor: autor

Na Slici 3.8. vidimo kako bi izgledala putanja naše rakete, a crvena točka predstavlja uvjet zadatka koji je glasio da želimo da se ta naša raketa nalazi na 50 metara iznad tla nakon 5 sekundi od lansiranja, što smo uspješno postigli poštivajući zadani rubni uvjet pri rješavanju problema.



Slika 3.8. Vizualni prikaz rješenja zadatka, odnosno putanje rakete, Izvor: autor

Sada odredimo $y'(0)$. Iz aproksimacija konačnim razlikama znamo da je

$$y' \approx y_{i+1} - \frac{y_i}{2h}, \tag{3.22}$$

što znači da je za $i = 0$:

$$y'(0) \approx \frac{y_1 - y_{-1}}{2h}, \tag{3.23}$$

ali ne znamo što je y_{-1}. Zapravo, možemo izračunati y_{-1} jer znamo vrijednosti y na svakoj mrežnoj točki. Iz aproksimacije konačnim razlikama za drugu derivaciju, znamo da je:

$$y_{-1} - 2y_0 + y_1 = -gh^2, \tag{3.24}$$

stoga možemo riješiti za y_{-1} i zatim dobiti brzinu pri lansiranju. Sad možemo ubaciti (3.24) u (3.23):

$$y'(0) \approx \frac{(y_1 - (-gh^2 + 2y_0 - y_1))}{2h}, \tag{3.25}$$

i dobijemo rješenje koje glasi:

$$y'(0) \approx 34.5m/s.$$

Možemo primijetiti da dobivamo ispravnu brzinu pri lansiranju koristeći metodu konačnih razlika.

4. Problemi provođenja topline

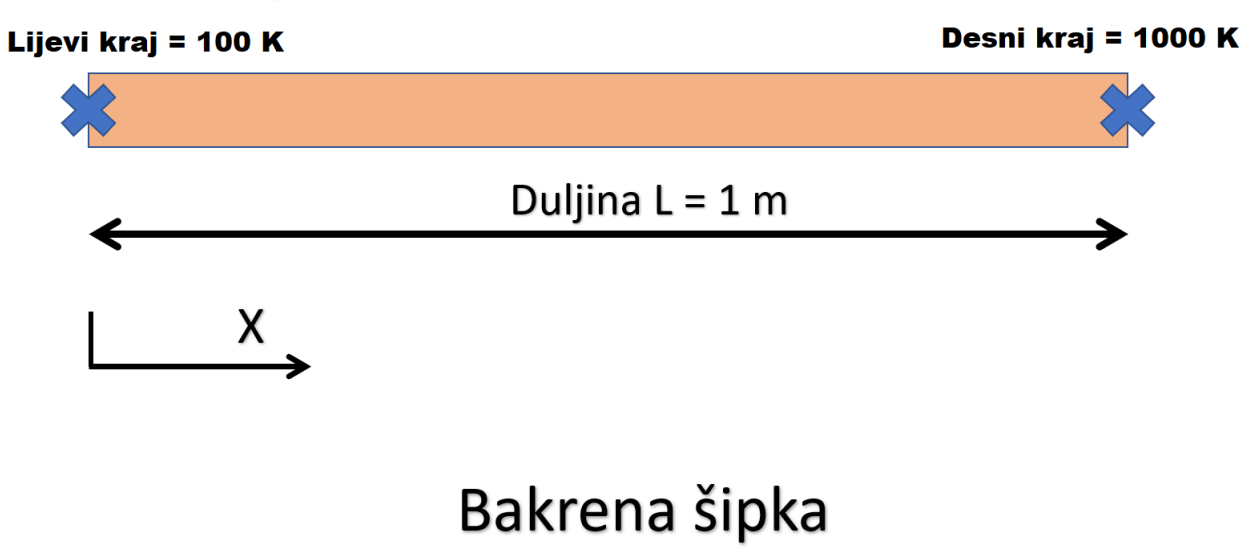
U ovom poglavlju primijenit ćemo metodu na nekoliko primjera iz inženjerstva, specifično, zadatke vezane uz problem provođenja topline.

4.1. Stacionarna jednadžba provođenja topline u jednoj dimenziji

Jednadžba provođenja topline opisuje ravnotežu topline u materijalu, gdje toplina protječe iz točaka s višom temperaturom prema točkama s nižom temperaturom, a toplinski izvori mogu dodavati ili oduzimati toplinu iz sustava [6].

Jednadžbu se može riješiti analitički u nekim posebnim slučajevima, ali u većini praktičnih primjena rješava se numeričkim metodama.

Uzet ćemo jednostavan primjer provođenja topline:



Slika 4.1. Vizualni prikaz zadatka provođenja topline po šipci, Izvor: autor

Kao što je prikazano na Slici 4.1, imamo šipku dugu $1m$ načinjenu od bakra. Na jednom kraju te šipke održavana je temperatura od $100K$, a dok drugi kraj je zagrijan na $1000K$. Naš cilj će biti pronaći temperature na različitim mjestima duž te šipke. Jednadžba koja opisuje proces glasi:

$$\frac{\partial^2 T}{\partial x^2} + \frac{g}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t}, \tag{4.1}$$

gdje je:

- α - termalna difuzivnost m^2/s,

- $\alpha = k/\rho c$,
- k - toplinska provodljivost materijala $W/(m \cdot K)$,
- ρ - gustoća materijala kg/m^3,
- c - specifična toplinska kapacitivnost materijala $J/(kg \cdot K)$,
- g - volumetrična stopa unutarnje generacije topline W/m^3.

Pretpostavljamo da varijacija temperature duž y i z smjerova nije značajna u odnosu na varijaciju duž x smjera [8]. U nastavku rješavamo jednadžbu (4.1) sa sljedećim dodatnim pretpostavkama:

- $g = 0$, što ukazuje na odsutnost unutarnje generacije topline,
- temperatura se ne mijenja s vremenom, što ukazuje na uvjet stacionarnog stanja.

Primjenom navedenih pretpostavki na jednadžbu (4.1) dobivamo $\frac{\partial^2 T}{\partial x^2} = 0$; gdje $T = T(x)$, a ova parcijalna diferencijalna jednadžba se može dalje pojednostaviti u običnu diferencijalnu jednadžbu. [8]

Dobili smo stacionarnu jednadžbu $T'' = 0$ koju možemo zbog njene jednostavnosti riješiti i analitički, međutim mi ćemo je riješiti i metodom konačnih razlika. Njeno analitičko rješenje dobijemo integriranjem polazne jednadžbe:

$$\begin{aligned} T''(x) &= 0, \\ \int T''(x)\,dx &= \int 0\,dx, \\ T'(x) &= C, \\ \int T'(x)\,dx &= \int C\,dx, \\ T(x) &= Cx + D, \end{aligned} \tag{4.2}$$

gdje su C i D konstante integracije.

U ovom posebnom slučaju potrebna su nam dva rubna uvjeta, te oba su nam dostupna:

- Temperatura (T) jednaka je $T_{kraj1} = 100K$ za $x = 0$m,
- Temperatura (T) jednaka je $T_{kraj2} = 1000K$ za $x = 1$m.

Početnu jednadžbu (4.2) zamijenimo s aproksimacijom druge derivacije:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = 0. \tag{4.3}$$

Zamijenit ćemo y sa T koji će nam predstavljati temperaturu te pomnožimo prethodnu jednadžbu (4.3) s h^2 da dobijemo još jednostavniji oblik:

$$T_{i+1} - 2T_i + T_{i-1} = 0. \tag{4.4}$$

I ovo je jednadžba koju smo dobili koristeći metodu konačnih razlika. Podijeliti ćemo našu cijev na jednake segmente, recimo u njih pet, $i = 1, 2, 3, 4, 5, 6$, za koje su nam za prvu i zadnju već poznate njihove pripadne temperature. Sad za svaki i raspišemo jednadžbu ubacujući ih u jednadžbu (4.4): za $i = 2$ dobivamo:

$$T_3 - 2T_2 + T_1 = 0, \tag{4.5}$$

$i = 3$:

$$T_4 - 2T_3 + T_2 = 0, \tag{4.6}$$

$i = 4$:

$$T_5 - 2T_4 + T_3 = 0, \tag{4.7}$$

konačno, za $i = 5$ imamo:

$$T_6 - 2T_5 + T_4 = 0. \tag{4.8}$$

T_1 i T_6 su nam poznati kao rubni uvjeti i njih možemo ubaciti u prethodne jednadžbe (4.5) i (4.8), te navedene jednadžbe prebacimo u matrični oblik:

$$\begin{pmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} T_2 \\ T_3 \\ T_4 \\ T_5 \end{pmatrix} = \begin{pmatrix} -100 \\ 0 \\ 0 \\ -1000 \end{pmatrix},$$

gdje su nam pripadna rješenja, odnosno temperature u zadanim točkama:

$$\begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{pmatrix} = \begin{pmatrix} 100K \\ 280K \\ 460K \\ 640K \\ 820K \\ 1000K \end{pmatrix}.$$

Problem možemo riješiti i korištenjem Pythona. Prvo, trebamo definirati koeficijente tridijagonalne matrice i desnu stranu sustava kao liste. Zatim, pozvat ćemo se na funkciju sa Slike 3.4. koju ćemo implementirali u Pythonu kako bismo dobili vektor rješenja. Ispisivanjem dobivenog vektora rješenja u Pythonu, provjeravamo je li ono isto kao rješenje koje smo dobili ručno.

Uspoređivanjem dobivenih rješenja s već poznatim rješenjima, potvrđujemo da se rješenje tridijagonalnog sustava izračunato u Pythonu podudara s već dobivenim rješenjima, kao što smo i prikazali na Slici 4.2.

```python
import numpy as np

def rjesavanje_tridijagonalnih_sustava(a, b, c, d):
    n = len(d)
    x = np.zeros(n)

    # Eliminacija unaprijed
    for i in range(1, n):
        m = a[i] / b[i-1]
        b[i] = b[i] - m * c[i-1]
        d[i] = d[i] - m * d[i-1]

    # Supstitucija unatrag
    x[n-1] = d[n-1] / b[n-1]
    for i in range(n-2, -1, -1):
        x[i] = (d[i] - c[i] * x[i+1]) / b[i]

    return x

# Koeficijenti tridijagonalnog sustava
a = [1, 1, 1, 1]
b = [-2, -2, -2, -2]
c = [1, 1, 1, 0]
d = [-100, 0, 0, -1000]

solution = rjesavanje_tridijagonalnih_sustava(a, b, c, d)
print("Rješenje:", solution)
```

```
Rješenje: [280. 460. 640. 820.]
```

Slika 4.2. Kod u Pythonu za rješenje gornjeg primjera, Izvor: autor

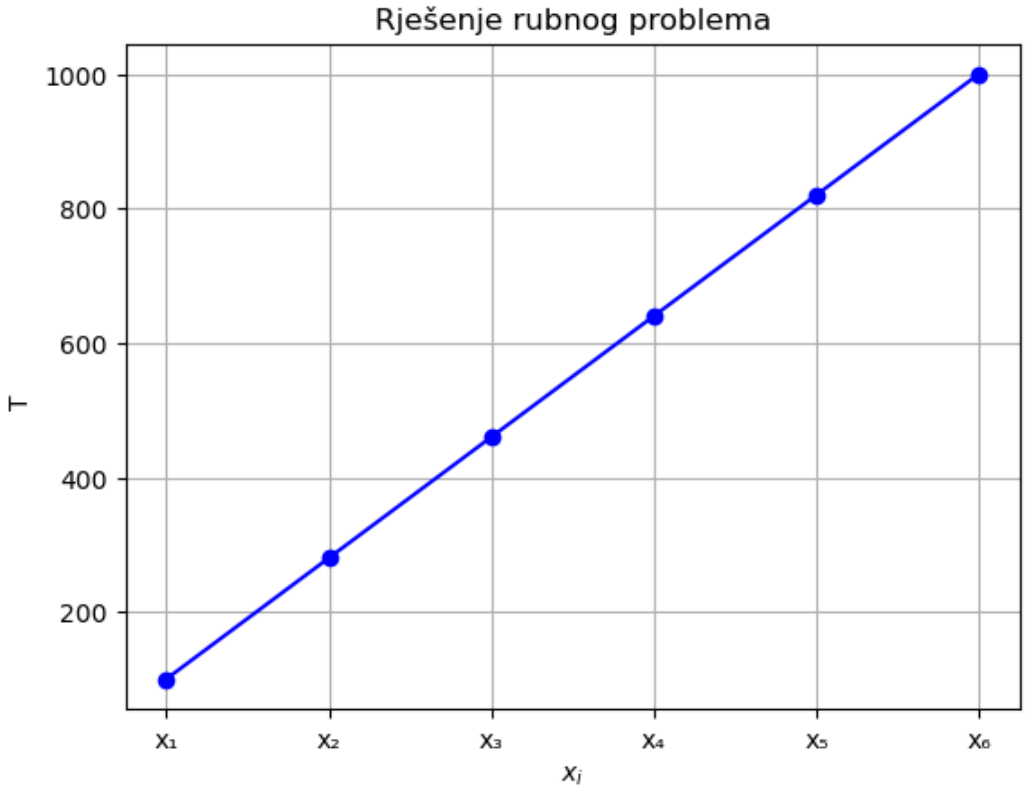
Također, korištenjem Pythona možemo prikazati dobiveno rješenje i grafički. Grafički prikaz rješenja koristan je jer nam pruža jasniju sliku o promjenama i trendovima u rješenju. Može nam pomoći u prepoznavanju važnih obrazaca, ekstrema ili glatkih prijelaza u vrijednostima. Također nam olakšava usporedbu različitih rješenja ili analizu utjecaja promjene koeficijenata na rezultat. Graf nas podržava u boljem razumijevanju i interpretaciji rješenja tridijagonalnog sustava.

Slika 4.3. prikazuje kod potreban za plotiranje dobivenih vrijednosti temperatura kao vizualnu reprezentaciju rješenja rubnog problema. Na osi apscisa imamo indekse ili pozicije u vektoru rješenja, dok se na osi ordinata prikazuju vrijednosti samog rješenja, odnosno dobivene temperature. Graf nam omogućuje da intuitivno shvatimo kako se vrijednosti mijenjaju u sustavu i kakav je oblik rješenja, koji u našem slučaju, je približno linearan.

4.2. Prijenos topline kroz rebrastu površinu

Razmotrimo sada malo složeniji primjer: prijenos topline kroz rebrastu površinu.

```
# Rješavanje tridijagonalnog sustava
solution = rjesavanje_tridijagonalnih_sustava(a, b, c, d)

# Dodavanje rubnih točaka na početak i kraj vektora rješenja
solution = np.insert(solution, 0, 100)
solution = np.append(solution, 1000)

# Generiranje x-koordinata
x_labels = ['x\u2081', 'x\u2082', 'x\u2083', 'x\u2084', 'x\u2085', 'x\u2086']

# Plotiranje rješenja
plt.plot(x_labels, solution, marker='o', linestyle='-', color='b')
plt.xlabel(r'$x_i$')
plt.ylabel('T    ')
plt.title('Rješenje rubnog problema')
plt.grid(True)
plt.show()
```



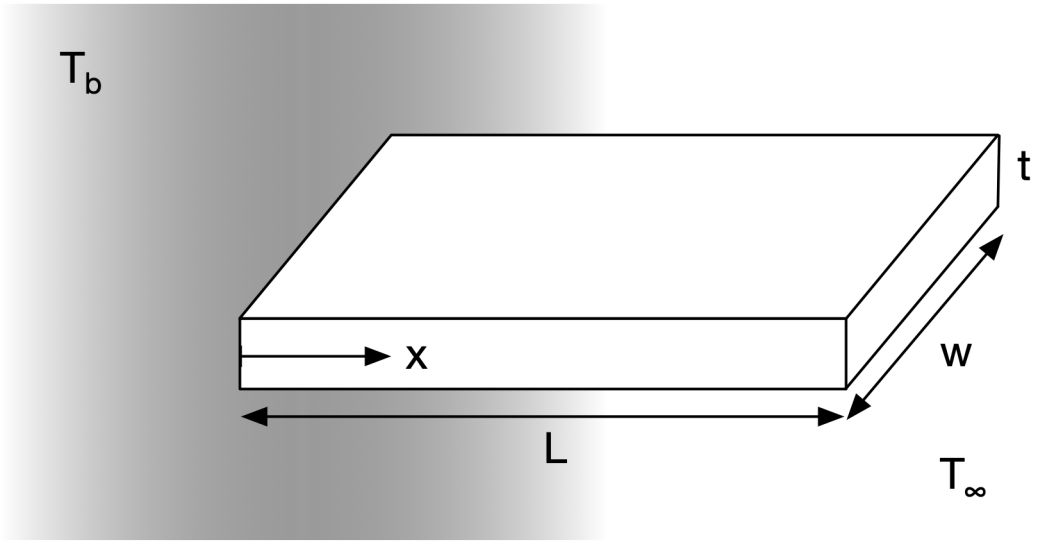
Slika 4.3. Kod i grafički prikaz u Pythonu za rješenje provođenja temperature, Izvor: autor

Prijenos topline kroz rebrastu površinu odvija se putem procesa konvekcije i kondukcije. Rebrasta površina se koristi kako bi se povećala površina koja dolazi u kontakt s okolinom te se na taj način povećala učinkovitost prijenosa topline. Konvekcija se odvija zbog strujanja fluida oko rebraste površine. Toplina se prenosi s rebraste površine na okolinu zahvaljujući strujanju fluida oko rebara, što povećava protok topline iz rebraste površine u okolinu. Kondukcija se odvija zbog toplinske provodljivosti materijala iz kojeg je napravljena rebrasta površina. Toplina se prenosi kroz rebraste materijale, što omogućava da toplina prođe kroz rebrastu površinu i prenese se na okolinu. Kombinacija konvekcije i kondukcije omogućuje učinkovit prijenos topline kroz rebrastu

površinu. U praksi, rebraste površine se često koriste u različitim uređajima, kao što su hladnjaci, izmjenjivači topline, i drugi uređaji u kojima je prijenos topline bitan [7].

Rebraste površine koriste se za poboljšanje prijenosa topline između tijela i okolne tekućine. To je osnovna metoda koja se često koristi kada se ne mogu promijeniti ovisni parametri kao što su temperatura tekućine i koeficijent prijenosa topline konvekcijom. Koeficijent prijenosa topline konvekcijom ovisi o svojstvima tekućine, režimu strujanja i brzini protoka.

U elektronici se rebraste površine koriste za održavanje komponenata unutar radnih temperatura uz pasivno ili aktivno hlađenje. Istraživanja se provode eksperimentalno, analitički i numerički kako bi se pronašli najpogodniji geometrijski oblici. Analiza prijenosa topline konvekcijom zahtijeva numeričko rješavanje više povezanih diferencijalnih jednadžbi čak i za jednostavne geometrijske oblike. Metoda konačnih razlika je široko korištena za analizu prijenosa topline s različitim geometrijama, veličinama i radnim uvjetima [6].



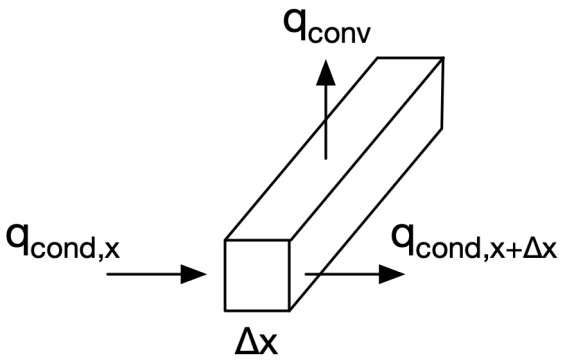
Slika 4.4. Geometrijski prikaz rebraste površine, Izvor: [6]

U ovoj situaciji na Slici 4.4. imamo općenito temperaturu $T = T(x)$ gdje je x smjer kretanja, temperaturu tijela T_b, temperaturu okolne tekućine T_∞, duljinu L, širinu w i debljinu površine t, toplinsku provodljivost materijala rebraste površine k, q predstavlja stopu prijenosa topline te koeficijent prijenosa topline konvekcijom h.

Rubni uvjeti mogu se definirati na različite načine, ali općenito možemo reći da je temperatura rebra na zidu jednaka temperaturi tijela, a rebra na kraju su izolirana. To nam daje sljedeće:

$$\begin{aligned} T(0) &= T_b, \\ q(L) = 0 \rightarrow \frac{dT}{dx}(0) &= 0. \end{aligned} \tag{4.9}$$

Naš cilj je dobiti temperature pri različitim udaljenostima $T(x)$. Da bismo to postigli, moramo odrediti diferencijalnu jednadžbu.



Slika 4.5. Kontrola volumena prijenosa topline kroz rebrastu površinu, Izvor: [6]

Za određeni volumenski presjek rebara možemo definirati stope prijenosa topline putem kondukcije kroz rebra i konvekcije s rebara na zrak kao što je prikazano na Slici 4.5.

$$\begin{aligned} q_{conv} &= hP(T - T_\infty)\Delta x, \\ q_{cond,x} &= -kA_c \left(\frac{dT}{dx}\right)_x, \\ q_{cond,x+\Delta x} &= -kA_c \left(\frac{dT}{dx}\right)_{x+\Delta x}, \end{aligned} \tag{4.10}$$

gdje je q_{conv} stopa prijenosa topline putem konvekcije s rebara na okolni zrak, q_{cond} stopa prijenosa topline putem kondukcije kroz rebara, P opseg (tako da je Pdx površina prijenosa topline na fluid), A_c je poprečni presjek i $\left(\frac{dT}{dx}\right)_x$ gradijent temperature u smjeru x.

Izvršavajući ravnotežu kroz kontrolni volumen dobivamo:

$$q_{cond,x+\Delta x} = q_{cond,x} - q_{conv}, \tag{4.11}$$

$$\begin{aligned} -kA_c \left(\frac{dT}{dx}\right)_{x+\Delta x} &= -kA_c \left(\frac{dT}{dx}\right)_x - hP(T - T_\infty)\Delta x, \\ -kA_c \frac{\left(\frac{dT}{dx}\right)_{x+\Delta x} - \left(\frac{dT}{dx}\right)_x}{\Delta x} &= -hP(T - T_\infty). \end{aligned} \tag{4.12}$$

Uzimanjem limesa kad $\Delta x \to 0$ u zadnjoj jednakosti u (4.12) dobivamo

$$-kA_c \left(\frac{d^2T}{dx^2}\right)_x = -hP(T - T_\infty). \tag{4.13}$$

Daljnim sređivanjem jednadžbe (4.13) gdje radi jednostavnijeg zapisa zamijenimo izraz $\frac{hP}{kA_c}$ s m^2 dolazimo do:

$$\begin{aligned} \frac{d^2T}{dx^2} &= \frac{hP}{kA_c}(T - T_\infty), \\ \frac{d^2T}{dx^2} &= m^2(T - T_\infty), \end{aligned} \tag{4.14}$$

čime smo dobili traženu jednadžbu.

Možemo dobiti egzaktno rješenje za tu ODJ. Radi praktičnosti, definirajmo novu varijablu, θ, koja je normalizirana temperatura $\theta = T - T_\infty$ pri čemu se derivacije θ i T podudaraju. Ubacujući nove varijable u jednadžbu (4.14) dobijemo sljedeću homogenu jednadžbu drugog reda:

$$\theta'' - m^2\theta = 0. \tag{4.15}$$

Egzaktno rješenje tada glasi:

$$\theta(x) = c_1 e^{-mx} + c_2 e^{mx}, \tag{4.16}$$

gdje su c_1 i c_2 konstante. Kada vratimo supstituciju θ dobijemo:

$$T(x) = T_\infty + c_1 e^{-mx} + c_2 e^{mx}. \tag{4.17}$$

Također, zbog već zadanih rubnih uvjeta možemo lako odrediti naše neodređene konstante. Za rubni uvjet $x = 0$, imamo $T(0) = T_b$, što nam daje sljedeću jednadžbu:

$$T_\infty + c_1 + c_2 = T_b. \tag{4.18}$$

Za drugi rubni uvjet $x = L$, imamo $\frac{dT}{dx}(L) = 0$, čime dobivamo drugu jednadžbu:

$$c_1(-m)e^{-mL} + c_2 m e^{mL} = 0. \tag{4.19}$$

Iz jednadžbe (4.18) možemo izraziti c_2:

$$c_2 = T_b - T_\infty - c_1, \tag{4.20}$$

te zatim ubacimo jednadžbu (4.20) u (4.19):

$$c_1(-m)e^{-mL} + (T_b - T_\infty - c_1)me^{mL} = 0, \tag{4.21}$$

i sad možemo riješiti za c_1:

$$\begin{aligned} c_1(me^{mL} - me^{-mL}) &= me^{mL}(T_b - T_\infty), \\ c_1(m(e^{mL} - e^{-mL})) &= me^{mL}(T_b - T_\infty), \\ c_1 &= \frac{me^{mL}(T_b - T_\infty)}{m(e^{mL} - e^{-mL})}. \end{aligned} \tag{4.22}$$

Sad kad imamo c_1, možemo je ubaciti nazad u početnu jednadžbu (4.20) za c_2:

$$c_2 = T_b - T_\infty - \frac{e^{mL}(T_b - T_\infty)}{e^{mL} - e^{-mL}}. \tag{4.23}$$

S određenim konstantama c_1 i c_2 možemo zapisati konačni oblik naše željene jednadžbe $T(x)$:

$$T(x) = T_\infty + \frac{e^{mL}(T_b - T_\infty)}{e^{mL} - e^{-mL}}e^{-mx} + T_b - T_\infty - \frac{e^{mL}(T_b - T_\infty)}{e^{mL} - e^{-mL}}e^{mx}. \tag{4.24}$$

Koristit ćemo ovo kako bismo provjerili točnost numeričkog rješenja.

Sada ćemo riješiti problem numerički koristeći metodu konačnih razlika. Zamijenit ćemo derivaciju u jednadžbi (4.15) s aproksimacijom konačne razlike:

$$\frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2} = m^2(T_i - T_\infty), \tag{4.25}$$

koju možemo preurediti u rekurzivnu formulu:

$$T_{i-1} + T_i(-2 - \Delta x^2 m^2) + T_{i+1} = -m^2 \Delta x^2 T_\infty. \tag{4.26}$$

Ovo nam daje jednadžbu za sve unutarnje čvorove. Možemo koristiti gornje rubne uvjete da dobijemo jednadžbe za rubne čvorove. Iz rubnih uvjeta (4.9) dobijemo dvije dodatne jednadžbe

$$T_1 = T_b, \tag{4.27}$$

$$\frac{T_n - T_{n-1}}{\Delta x} = 0 \to T_{n-1} - T_n = 0. \tag{4.28}$$

Kombinirajući sve prethodno navedene jednadžbe dobivamo odgovarajući sustav zapisan matrično $AT = b$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 - \Delta x^2 m^2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 - \Delta x^2 m^2 & 1 \\ 0 & \cdots & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{n-1} \\ T_n \end{bmatrix} = \begin{bmatrix} T_b \\ -m^2 \Delta x^2 T_\infty \\ \vdots \\ -m^2 \Delta x^2 T_\infty \\ 0 \end{bmatrix}. \tag{4.29}$$

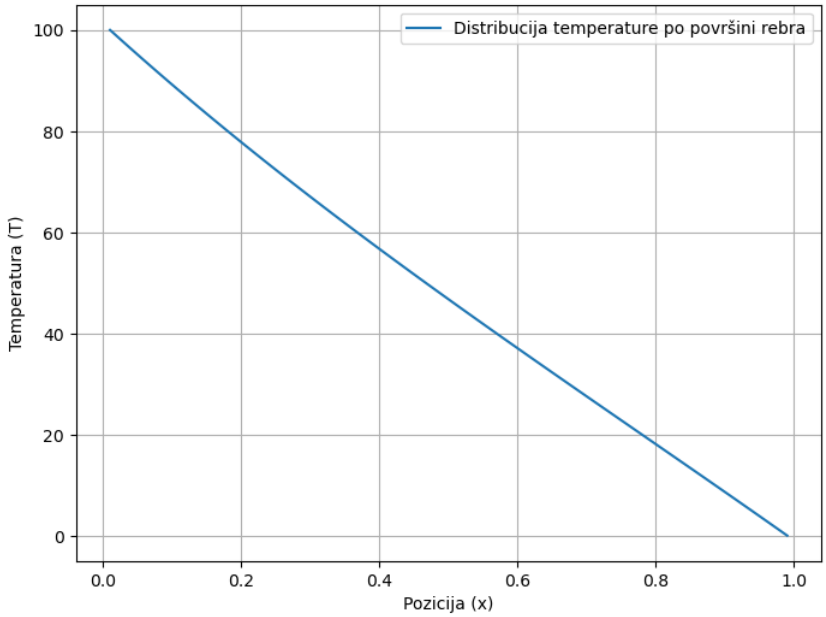
Da implementiramo ovaj problem u Pythonu, moramo izabrati neke proizvoljne parametre. Duljina rebara je definirana na početku zadatka, te ona iznosi $L = 1\text{m}$, diskretizirat ćemo našu mrežu na $n = 100$ podintervala, a korak među susjednih točaka mreže ćemo izračunati preko izraza $h = \frac{L}{n+1} = \frac{1}{101}$. Temperaturu tijela postavimo proizvoljno na $T_b = 100°\text{C}$, a temperaturu okoline na $T_\infty = 20°\text{C}$. Postavili smo da parametar m bude $m = 1$, koji predstavlja korijen vrijednosti $\frac{hP}{kA_c}$ radi jednostavnosti zapisa. Rubni uvjeti su već prije definirani koje ćemo sad koristiti u kodu.

Na slici 4.6. smo sve te parametre uveli u naš kod koji izbacuje rješenje tridijagonalnog sustava i opisuje promjenu temperature duž površine rebara. U kodu smo stavili da i dobivena rješenja prikazuje grafički da možemo vizualno zaključiti kako se mijenja temperatura kada se približavamo drugom kraju rebara, kao što je vidljivo na Slici 4.7. U praksi, morali bi prilagoditi puno više parametara da bi opisali sličan problem iz okoline, dok smo mi prikazali pojednostavljen slučaj.

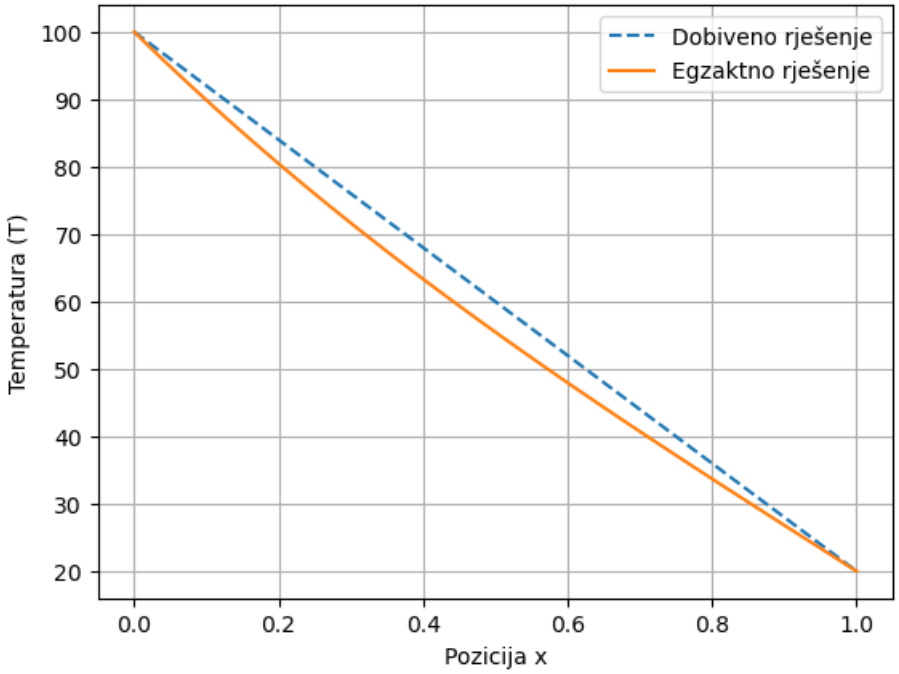
Slika 4.8. prikazuje odstupanje egzaktnog rješenja od rješenja dobivenog metodom konačnih razlika, te možemo zaključiti da su približno istih vrijednosti. Smanjenjem koraka i povećanjem diskretizacijske mreže, posljedično će se i smanjiti njena pogreška vidljiva na slici.

```python
import numpy as np
import matplotlib.pyplot as plt

# definiranje parametara
L = 1.0         # duljina rebra
n = 100         # broj podintervala
Tb = 100.0      # temperatura tijela
T_infinity = 20.0  # temperatura okoline
m = 1.0         # proizvoljni parametar m

# Diskretizacija domene
h = L / (n + 1)
x = np.linspace(0, L, n+2)

A = np.zeros((n, n))
diagonal = np.ones(n)
sub_diagonal = np.ones(n - 1)
super_diagonal = np.ones(n - 1)

for i in range(1, n - 1):
    A[i, i] = -2.0 - (h**2 * m**2)
    A[i, i - 1] = 1.0
    A[i, i + 1] = 1.0

A[0, 0] = 1.0
A[n-1, n-1] = -1.0

# desna strana, uključeni rubni uvjeti
b = np.zeros(n)
b[0] = Tb
b[1:] = -m**2 * h**2 * T_infinity

# rješenje tridijagonalnog sustava
T = np.linalg.solve(A, b)

# izrada grafa
plt.figure(figsize=(8, 6))
plt.xlabel('Pozicija (x)')
plt.ylabel('Temperatura (T)')

plt.plot(x[1:-1], T, label= 'Distribucija temperature po površini rebra')
plt.legend()
plt.grid(True)
plt.show()
```

Slika 4.6. Kod u Pythonu koji računa promjenu temperature duž rebara, Izvor: autor



Slika 4.7. Grafički prikaz temperature duž rebara, Izvor: autor



Slika 4.8. Odstupanje dobivenog rješenja od egzaktnog rješenja, Izvor: autor

5. Zaključak

U ovom radu obrađena je metoda konačnih razlika za rješavanje običnih diferencijalnih jednadžbi i pripadnih rubnih problema. U uvodnom dijelu rada objašnjen je sav matematički aparat neophodan za razumijevanje navedene metode. Tako je definirana derivacija, diferencijalna jednadžba, rubni problem i slično.

Osnova metode konačnih razlika je aproksimacija derivacije koju smo izveli pomoću Taylorovog razvoja funkcije. Na taj smo način izveli različite aproksimacijske sheme, kao što su razlike unaprijed, unazad i centralne razlike. Analizirali smo izvedene sheme na više primjera i usporedili ih s obzirom na točnost aproksimacije derivacije. Pokazali smo empirijski da se kod aproksimacija pojavi pogreška usporediva s onom koju su predvidjeli teorijski izvodi. Također smo pokazali da za točnu aproksimaciju trebamo odabrati značajno manji korak ako je prirast derivacije oko točke aproksimacije velik.

U nastavku smo primijenili metodu konačnih razlika na konkretne rubne probleme. Pokazali smo da se na taj način rješavanjem rubnih problema svodi na rješavanje linearnih sustava čija je matrica tridijagonalnog oblika. Objasnili smo kako se takvi sustavi mogu riješiti Thomasovim algoritmom.

Metodu konačnih razlika implementirali smo i u programskom jeziku Python-u. Drugim riječima, u programskom jeziku Python, napisali smo programski kod za Thomasov algoritam i primijenili ga na više primjera. Korištenje Pythona omogućila nam je i grafičko prikazivanje numeričkog rješenja, ali i usporedbe analitičkog i numeričkog rješenja.

U radu smo riješili nekoliko apstraktnih primjera rubnih problema, ali i nekoliko primjera s jasnom inženjerskom primjenom. Tako smo se bavili problemom vertikalnog hica i distribucije topline u ravnom štapu. U zadnjem dijelu rada primijenili smo metodu konačnih razlika na nešto složeniji problem provođenja topline kroz rebrastu površinu.

Možemo zaključiti da je metoda konačnih razlika relativno jednostavna za primjenu kod rješavanja rubnih problema. Može se uočiti da poteškoća može nastati u dva slučaja. Naime, aproksimacija derivacije može biti više i manje točna te tako numeričko rješenje ovisi o kvaliteti odabrane aproksimacijske sheme. Nadalje, pokazali smo da se traženje numeričkog rješenja rubnog problema svodi na rješavanje linearnog sustava što također može imati svoje izazove, posebice u slučaju većeg broja nepoznanica koje se dobije finijim diskretizacijama domene. Također, programski jezik Python pokazao se dobrim izborom za implementaciju opisane metode, posebno sa stajališta jednostavnosti i dostupnosti softvera.

Bibliografija

[1] Cassel, Kevin W., "Matrix, Numerical, and Optimization Methods in Science and Engineering", Cambridge University Press, UK, 2021.

[2] Burden, R. L., Faires, J. D., "Numerical Analysis (9th ed.)", Cengage Learning, USA, 2010.

[3] Heinzl, R., "Numerical discretization schemes", s Interneta, https://www.iue.tuwien.ac.at/phd/heinzl/node27.html, 05.11.2022

[4] Zill, Dennis G., "Differential Equations with Boundary-Value Problems", Cengage Learning, USA, 2017.

[5] Hartman, G., "Taylor series", s Interneta, https://math.libretexts.org/Bookshelves/Calculus/Calculus3e(Apex)/083A-Sequences-and-Series/8.073A-Taylor-Polynomials, 15.12.2022.

[6] Incropera, Frank P., DeWitt, David P., Bergman, Theodore L., Lavine, Adrienne S., "Fundamentals of Heat and Mass Transfer (sixth edition)", John Wiley and Sons, United states, 2007.

[7] Hlupić, Ines, "Prijenos topline", s Interneta, https://www.fkit.unizg.hr/download/repository/prijenosTopline.pdf, 30.03.2023.

[8] Engineering-stream group, "Numerical methods", s Interneta, https://engineering-stream.com/engineering-stream-Numerical-Methods.html, 15.04.2023.

[9] Scitovski, Rudolf, "Numerička matematika", s Interneta, https://www.mathos.unios.hr/pim/Materijali/Num.pdf, 05.05.2023.

[10] Hellevik, Leif R., "Numerical methods for engineers", s Interneta, https://folk.ntnu.no/leifh/teaching/tkt4140/.-main040.htmlch3:sec1, 01.07.2023.

[11] Kong, Q., Siauw, T., Bayen, A., "The shooting method", s Interneta, https://pythonnumericalmethods.berkeley.edu/notebooks/chapter23.02-The-Shooting-Method.html, 06.07.2023.

[12] Maplesoft, "Taylor's series", s Interneta, https://www.maplesoft.com/ns/math/taylor-series.aspx, 10.08.2023.

[13] Langtangen, Hans P., "Introduction to computing with finite difference methods", s Interneta, http://hplgit.github.io/INF5620/doc/pub/sphinx-decay/.-main-decay001.html, 20.08.2023.

[14] Dražić, I.; Inženjerska Matematika ET - interna skripta, Sveučilište u Rijeci, Tehnički fakultet, Rijeka, 2022.

Sažetak i ključne riječi

U ovom radu opisali smo metodu konačnih razlika za numeričko rješavanje rubnog problema običnih diferencijalnih jednadžbi. Metoda je bazirana na aproksimaciji derivacija podjeljenim razlikama. U radu su objašnjeni svi pojmovi nužni za razumijevanje metode (derivacija, diferencijalna jednadžba, rubni problem, Taylorov razvoj). Izvedeno je nekoliko aproksimacijskih shema derivacije i analazirana je njihova točnost. Nadalje, metoda konačnih razlika je primijenjena na više primjera s posebnim naglaskom na primjenu u inženjerstvu. Metoda je implementirana unutar programskog jezika Python.

Ključne riječi: metoda konačnih razlika, Taylorov polinom, derivacija, diferencijalna jednadžba, provođenje topline, Python

Summary and key words

In this paper, we described the finite difference method for numerically solving the boundary value problem of ordinary differential equations. The method is based on the approximation of derivatives by divided differences. The paper explains all terms necessary for understanding the method (derivation, differential equation, boundary value problem, Taylor development). Several approximate derivation schemes were derived and their accuracy was analyzed. Furthermore, the finite difference method is applied to several examples with a special emphasis on the application in engineering. The method is implemented within the Python programming language.

Keywords: finite difference method, Taylor polynomial, derivative, differential equation, heat conduction, Python