SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Lovre Grzunov

USPOREDBA METODA ZA PROMJENU RAZLUČIVOSTI SLIKE

Diplomski rad

Voditelj rada:
Doc. dr. sc. Tina Bosner

Zagreb, 2019.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Zahvaljujem se mentorici Doc. dr. sc. Tini Bosner, na ažurnosti, susretljivosti i svim savjetima koje mi je pružila prilikom izrade ovog rada.
Zahvaljujem se svim prijateljima i onima koji su me inspirirali na ovom putu.
Najveće zahvale roditeljima i sestri, koji su mi omogućili sve kako bi došao do ovog postignuća.
Posebne zahvale Matei na svom strpljenju i pozitivi kroz sve ove godine.

Sadržaj

Uvod

Digitalno doba u kojem živimo pruža nam nebrojene mogućnosti i tehnološke inovacije. Ponuda dimenzija zaslona koji omogućuju prikaz digitalne slike nikad nije bila veća, stoga je gotovo nemoguće da isti sadržaj bude prikazan u jednakim dimenzijama na svim ponuđenim zaslonima. Prilikom uvećanja ili smanjivanja slike, stoga posežemo za nekim od algoritama kako bi uspješno reprezentirali sliku na danom zaslonu. Osim smanjenja i povećanja slike, algoritmi za promjenu razlučivosti pogodni su i za ispravak nepravilnosti nastalih prostornim iskrivljenjem. Ovim radom obradit ćemo metode kubične konvolucijske interpolacije i kubične splajn interpolacije te otkriti pozadinu pojedinih algoritama, i otkriti očekivanu grešku. Osim njih dotaknut ćemo se i nešto jednostavnijih metoda, poput metode najbližeg susjeda i metode linearne interpolacije. Na samom kraju primijenit ćemo algoritme kubične konvolucijske interpolacije, kubične splajn interpolacije, najbližeg susjeda i linearne interpolacije, te usporediti dobivene rezultate nad realnim primjerima, i uz pomoć metode evaluacije algoritama.

Poglavlje 1

Kubična konvolucijska interpolacija

1.1 Interpolacijska funkcija

Želimo li neki fizički proces ili fizičku pojavu pojednostavniti nekakvim modelom koji ju dobro opisuje, kažemo da smo taj proces, odnosno pojavu, aproksimirali. Stoga, ako imamo potrebu jednu matematičku funkciju pojednostavniti nekom drugom koja ju dobro opisuje koristit ćemo se aproksimacijskom funkcijom. Poznato je više metoda, odnosno aproksimacijskih funkcija kojima se neka matematička funkcija može aproksimirati. Ovim radom razmotrit ćemo jednu od jednostavnijih, a to je interpolacija. Skripta, Numerička analiza [4], na 288. stranici, nudi nam uvid u pojam aproksimacijske funkcije i samu definiciju interpolacijske funkcije.

Neka su nam poznate informacije o funkciji $f(x)$, na nekom skupu $X \subseteq \mathbb{R}$. Na osnovu tih informacija, funkciju $f(x)$ želimo zamijeniti nekom drugom funkcijom $g(x)$ na istom zadanom skupu, tako da su te dvije funkcije bliske u nekom smislu. Kažemo da funkciju f aproksimiramo funkcijom g. Skup X je najčešće interval oblika $[a, b]$, ali može biti i diskretan skup točaka na tom intervalu. Budući da se naš problem zasniva na obradi digitalne slike, skup podataka koji ćemo koristiti bit će diskretan. Funkcija $g(x)$ može se izabrati na više načina, no to sve ovisi o problematici s kojom se susrećemo. U načelu, funkciju $g(x)$ biramo tako da nam računanje bude što jednostavnije. Ona obično ovisi o parametrima $a_k, k = 0, ...m$. Te parametre treba odrediti po nekom kriteriju

$$g(x) = g(x; a_0, a_1, ..., a_m). \tag{1.1}$$

Jedno od osnovnih svojstava interpolacijskih funkcija jest da se njezina vrijednost podudara s vrijednostima točaka skupa kojeg aproksimira. Te točke obično nazivamo čvorovima interpolacije. Prilikom traženja funkcije $g(x)$, od podataka o funkciji $f(x)$, koristimo samo informacije o njezinoj vrijednosti na skupu od $(n+1)$ točaka, tj. podaci oblika (x_k, f_k), gdje je $f_k = f(x_k)$, za $k = 0, ...n$. Parametri $a_0, ..., a_n$, kojih mora biti točno onoliko koliko je i

podataka, određuju se iz sljedećeg uvjeta

$$g(x_k; a_0, a_1, ..., a_n) = f_k, k = 0, ..., n. \tag{1.2}$$

Općenito se radi o nelinearnom sustavu jednadžbi, no ako je funkcija $g(x)$ linearna, tada za parametre a_k dobivamo sustav od $(n + 1)$ linearnih jednadžbi i s isto toliko nepoznanica. Zbog same problematike našeg slučaja, osim navedenih, vrijedit će još dva uvjeta. Sama analiza problema zasniva se na jednodimenzionalnom polju, ali sliku reprezentiramo kao dvodimenzionalno polje. Stoga, dvodimenzionalnu interpolaciju postižemo primjenom jednodimenzionalne interpolacije, jednom u svakoj od dvije dimenzije. Sljedeći bitan uvjet koji vrijedi za sve čvorove interpolacije u našem slučaju, jest da su ekvidistantni.

Keys [7], na stranici 1153, sa svim pripadajućim referencama, navodi kako se mnogo interpolacijskih funkcija, kojima su točke jednako udaljene, mogu napisati u obliku

$$g(x) = \sum_k c_k u\left(\frac{x - x_k}{h}\right). \tag{1.3}$$

Pritom je, $g(x)$ interpolacijska funkcija, h inkrement danom skupu, x_k točke skupa koje interpoliramo, c_k parametri koji ovise o promatranom skupu, a u interpolacijska jezgra. Parametri c_k su odabrani tako da za svaku točku skupa x_k vrijedi uvjet interpolacijske funkcije, odnosno, $f(x_k) = g(x_k)$. U (1.3) interpolacijska jezgra u, pretvara diskretan skup podataka u neprekidnu funkciju, operacijom sličnoj konvoluciji.

Prema [9], konvoluciju možemo opisati kao integralnu funkciju koja govori o količini preklapanja funkcije f dok se pomiče preko funkcije g. Obično se konvolucija promatra na neograničenom skupu podataka, $\langle -\infty, \infty \rangle$ no prema [10], vidimo kako se može restringirati i na oganičen, odnosno, diskretan skup podataka. Neka je $f : \mathbb{R} \to \mathbb{R}$ i $g : \mathbb{R} \to \mathbb{R}$, tada je diskretna konvolucija definirana na sljedeći način

$$(f \odot g)(n) = \sum_{k=-\infty}^{\infty} f(k)g(n - k). \tag{1.4}$$

Kako je operacija konvolucije komutativna,

$$f \odot g = g \odot f \tag{1.5}$$

vrijedi i sljedeći izraz

$$(f \odot g)(n) = \sum_{k=-\infty}^{\infty} f(n - k)g(k). \tag{1.6}$$

1.2 Kubična konvolucijska interpolacijska jezgra

Kubična konvolucijska interpolacijska jezgra sastavljena je, po dijelovima, od polinoma trećeg stupnja, definiranih na intervalima $(-2,-1)$, $(-1,0)$, $(0,1)$ i $(1,2)$. Vrijednost jezgre izvan intervala $(-2,2)$ jednaka je nuli. Broj podataka za evaluaciju interpolacijske funkcije zbog toga je sveden na samo četiri uzorka. Osim toga, naša jezgra ima i svojstvo simetričnosti, te ju možemo zapisati u sljedećem obliku

$$u(s) = \begin{cases} A_1|s|^3 + B_1|s|^2 + C_1|s| + D_1 & 0 < |s| < 1 \\ A_2|s|^3 + B_2|s|^2 + C_2|s| + D_2 & 1 < |s| < 2 \\ 0 & 2 < |s|. \end{cases} \tag{1.7}$$

Uvođenjem dodatnog uvjeta za vrijednosti interpolacijske jezgre, $u(0) = 1$ i $u(n) = 0$ za $n \in \mathbb{Z} \setminus \{0\}$, značajno utječemo na brzinu i točnost izračuna vrijednosti. Budući da je h inkrementirajuća vrijednost, razliku između dvije točke skupa x_j i x_k tada možemo napisati kao $(j-k)h$. Uvrstimo li x_j, umjesto x, za (1.3) dobivamo izraz

$$g(x_j) = \sum_k c_k u(j-k). \tag{1.8}$$

No kako, $u(j-k)$ poprima vrijednost 0, osim u slučaju $j = k$, prethodni se izraz (1.8) može napisati u obliku

$$g(x_j) = c_j. \tag{1.9}$$

Zbog osnovnog svojstava interpolacijske funkcije $f(x_k) = g(x_k)$, $\forall k \in [0,n]$, navedenog u (1.2), vrijedi

$$c_j = f(x_j). \tag{1.10}$$

Što znači, da u (1.3), parametar c_k zapravo poprima vrijednosti skupa kojeg aproksimiramo. Stoga je ovo značajano poboljšanje performansi, u odnosu na druge interpolacijske metode, poput metode kubičnog splajna. Primjerice, vrijednost jezgre interpolacijskog splajna ne poprima vrijednost 0, za cijele brojeve različite od 0. Što direktno utječe na složenost algoritma, budući da se vrijednost c_k mora odrediti rješavanjem pripadajućeg linearnog sustava.

Osim svojstva, kako u interpolacijskim čvorovima poprima vrijednost 0 ili 1, interpolacijska jezgra mora biti neprekidna i derivabilna. Zbog tih uvjeta, (1.7) nam pruža dodatna saznanja o koeficijentima. Naime, uvjet $u(0) = 1$ i $u(1) = u(2) = 0$ nam daje sljedeće jednadžbe:

$$\begin{aligned} 1 &= u(0) = D_1 \\ 0 &= u(1^-) = A_1 + B_1 + C_1 + D_1 \\ 0 &= u(1^+) = A_2 + B_2 + C_2 + D_2 \\ 0 &= u(2^-) = 8A_2 + 4B_2 + 2C_2 + D_2 \end{aligned} \tag{1.11}$$

Još tri jednadžbe slijede iz uvjeta derivabilnosti jezgre u u točkama 0, 1 i 2:

$$\begin{aligned} -C_1 &= u'(0^-) = u'(0^+) = C_1 \\ 3A_1 + 2B_1 + C_1 &= u'(1^-) = u'(1^+) = 3A_2 + 2B_2 + C_2 \\ 12A_2 + 4B_2 + C_2 &= u'(2^-) = u'(2^+) = 0 \end{aligned} \tag{1.12}$$

Posljedica korištenja svih navedenih uvjeta nad jezgrom interpolacije je sustav od sedam jednadžbi s osam nepoznanica. Što znači, da nam za jedinstvenost rješenja ovog sustava nedostaje još jedan uvjet. $A_2 = -1$ može biti jedan od odabranih uvjeta ali, na taj način odustaje se od ideje da funkcija g aproksimira funkciju f do najvećeg mogućeg stupnja. Za potrebe preciznije aproksimacije, pretpostavimo da je funkcija f mnogostruko derivabilna, kako bi nad njom mogli primijeniti Taylorov teorem srednje vrijednosti. Na taj način, možemo odabrati A_2 takav, da se funkcija kubične konvolucijske interpolacije, i Taylorov red funkcije f podudaraju u što većoj mjeri.

Na 3. stranici skripte *Numerička analiza*, [4], definiran je Taylorov teorem srednje vrijednosti.

Teorem 1.2.0.1 (Taylorov teorem srednje vrijednosti)**.** *Neka funkcija f ima neprekidne derivacije do reda $n+1$, $n+1 > 0$, na segmentu $[a,b]$. Ako su $x, x_0 \in [a,b]$, tada je*

$$f(x) = p_n(x) + R_{n+1}(x),$$

$$p_n(x) = f(x_0) + \frac{(x-x_0)}{1!}f'(x_0) + \frac{(x-x_0)^2}{2!}f''(x_0) + ... + \frac{(x-x_0)^n}{n!}f^{(n)}(x_0),$$

$$R_{n+1}(x) = \frac{1}{n!}\int_{x_0}^{x}(x-t)^n f^{(n+1)}(t)dt = \frac{(x-x_0)^{n+1}}{(n+1)!}f^{(n+1)}(\xi),$$

za neki ξ između x i x_0.

Pritom se polinom p_n naziva Taylorov razvoj polinoma n-tog reda za funkciju f u točki x_0, a R_{n+1}, n-ti ostatak funkcije f u točki x_0.

Neka je $A_2 = a$, odredimo ostale koeficijente na temelju prethodnih sedam jednadžbi. Interpolacijsku jezgru sada možemo napisati u sljedećem obliku

$$u(s) = \begin{cases} (a+2)|s|^3 - (a+3)|s|^2 + 1 & 0 < |s| < 1 \\ a|s|^3 - 5a|s|^2 + 8a|s| - 4a & 1 < |s| < 2 \\ 0 & 2 < |s|. \end{cases} \tag{1.13}$$

Pretpostavimo da je x bilo koja točka u kojoj će naš skup podataka biti interpoliran. Tada se x sigurno nalazi između dva uzastopna interpolacijska čvora, x_j i x_{j+1}. Neka je $s = \frac{x-x_j}{h}$. S obzirom da vrijedi $\frac{x-x_k}{h} = \frac{x-x_j+x_j-x_k}{h} = s + j - k$, (1.3) možemo zapisati kao

$$g(x) = \sum_k c_k u(s+j-k). \tag{1.14}$$

Budući da je vrijednost interpolacijske jezgre u jednaka nuli, osim na intervalu $(-2, 2)$, i $0 < s < 1$, članovi sume (1.14) kojima vrijednost nije jednaka nuli daju sljedeći izraz

$$g(x) = c_{j-1}u(s+1) + c_j u(s) + c_{j+1}u(s-1) + c_{j+2}u(s-2). \tag{1.15}$$

Prema tome, iz (1.13) slijedi

$$\begin{aligned}
u(s+1) &= a(s+1)^3 - 5a(s+1)^2 + 8a(s+1) - 4a \\
&= as^3 - 2as^2 + as \\
u(s) &= (a+2)s^3 - (a+3)s^2 + 1 \\
u(s-1) &= -(a+2)(s-1)^3 - (a+3)(s-1)^2 + 1 \\
&= -(a+2)s^3 + (2a+3)s^2 - as \\
u(s-2) &= -a(s-2)^3 - 5a(s-2)^2 - 8a(s-2) - 4a \\
&= -as^3 + as^2.
\end{aligned}$$

Uvrštavanjem dobivenih jednadžbi u (1.15), kubična konvolucijska funkcija se može napisati u sljedećem obliku

$$\begin{aligned}
g(x) = &- [a(c_{j+2} - c_{j-1}) + (a+2)(c_{j+1} - c_j)]s^3 \\
&+ [2a(c_{j+1} - c_{j-1}) + 3(c_{j+1} - c_j) + a(c_{j+2} - c_j)]s^2 \\
&- a(c_{j+1} - c_{j-1})s + c_j.
\end{aligned} \tag{1.16}$$

Pretpostavimo, neka je funkcija f barem klase C^3 na intervalu $[x_j, x_{j+1}]$, primjenom Taylorevog teorema, (1.2.0.1), slijedi

$$c_{j+1} = f(x_{j+1}) = f(x_j) + hf'(x_j) + \frac{h^2 f''(x_j)}{2} + 0(h^3) \tag{1.17}$$

$$R_3(x) = \frac{h^3 f^{(3)}(\xi_1)}{6},$$

pritom je, $h = x_{j+1} - x_j$. Izraz $0(h^3)$ predstavlja članove veličine reda h^3, koji teže nuli, proporcionalno h^3. Unatoč tome, možemo odrediti i stvarnu grešku, koja je određena izrazom $R_3(x)$, za neki ξ, koji se nalazi između x_j i x_{j+1}. Analogno za c_{j+2} dobivamo

$$c_{j+2} = f(x_j) + 2hf'(x_j) + 2h^2 f''(x_j) + 0(h^3), \tag{1.18}$$

s pripadajućom greškom

$$R_3(x) = \frac{4h^3 f^{(3)}(\xi_2)}{3}.$$

Te c_{j-1}, s pripadajućom greškom

$$c_{j-1} = f(x_j) - hf'(x_j) + \frac{h^2 f''(x_j)}{2} + 0(h^3), \tag{1.19}$$

$$R_3(x) = -\frac{h^3 f^{(3)}(\xi_{-1})}{6}.$$

Uvrstimo li dobivene Taylorove redove funkcije f u točkama c_{j+1}, c_{j+2} i c_{j-1}, (1.17), (1.18) i (1.19) u funkciju kubične konvolucije, (1.16), dobivamo sljedeći izraz

$$\begin{aligned} g(x) = &-(2a+1)[2hf'(x_j) + h^2 f''(x_j)]s^3 \\ &+ \left[(6a+3)hf'(x_j) + \frac{(4a+3)h^2 f''(x_j)}{2}\right]s^2 \\ &- 2ahf'(x_j)s + f(x_j) + 0(h^3), \end{aligned} \tag{1.20}$$

s pripadajućom greškom

$$\begin{aligned} R_3(x) = &-\frac{h^3 s^3}{3}[4af^{(3)}(\xi_2) + \frac{af^{(3)}(\xi_{-1})}{2} + \frac{af^{(3)}(\xi_1)}{2} + f^{(3)}(\xi_1)] \\ &+\frac{h^3 s^2}{3}[af^{(3)}(\xi_1) + af^{(3)}(\xi_{-1}) + 4af^{(3)}(\xi_2) + \frac{3f^{(3)}(\xi_1)}{2}] \\ &-\frac{ah^3 s}{6}[f^{(3)}(\xi_1) + f^{(3)}(\xi_{-1})]. \end{aligned}$$

Lako je izračunati kako je $sh = x - x_j$, stoga je Taylorov red funkcije $f(x)$ u točki x_j, s pripadajućom greškom R_3

$$f(x) = f(x_j) + shf'(x_j) + \frac{s^2 h^2 f''(x_j)}{2} + 0(h^3), \tag{1.21}$$

$$R_3(x) = \frac{s^3 h^3 f^{(3)}(\xi)}{6}. \tag{1.22}$$

Oduzmimo (1.20) i (1.21),

$$\begin{aligned} f(x) - g(x) = &(2a+1)[2hf'(x_j) + h^2 f''(x_j)]s^3 \\ &- (2a+1)[3hf'(x_j) + h^2 f''(x_j)]s^2 \\ &+ (2a+1)shf'(x_j) + 0(h^3). \end{aligned} \tag{1.23}$$

Vidimo kako greška aproksimacijske funkcije $g(x)$, u (1.22), ovisi o ξ_1, ξ_{-1} i ξ_2 koji nisu ekvivalentni. Ne pomaže ni činjenica kako je Taylorov red funkcije $f(x)$ razvijen za neki ξ koji se nalazi između x i x_j, prema tome direktan izračun greške za $f(x) - g(x)$ nije

trivijalan, ali je ipak moguć zahvaljujući činjenici da su funkcije klase barem C^3. Stoga, neka je $H(x) = f(x) - g(x)$, tada razvojem funkcije $H(x)$ u Taylorov red oko točke x_j dobivamo grešku oblika

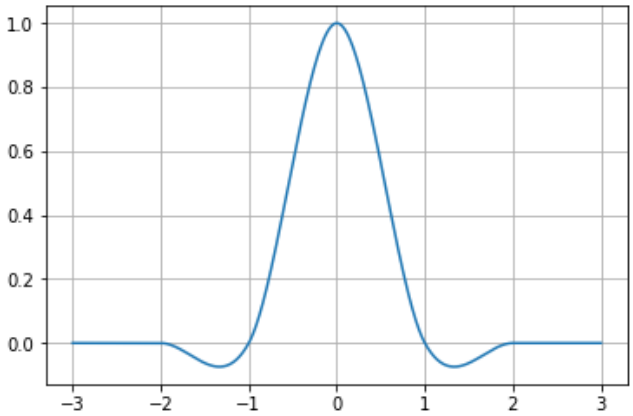
$$R_3(x) = -\frac{1}{6} f^{(3)}(\eta) h^3 s(2s^2 - 3s + 1),$$

pritom je $0 < s < 1$. Iz (1.23) očito je kako će razvoj funkcije $f(x)$ u Taylorov red trećeg stupnja, odgovarati interpolacijskoj funkciji $g(x)$, ako je vrijednost parametra a jednaka $-\frac{1}{2}$. Time smo odredili posljednji traženi uvjet, kojime možemo jedinstveno odrediti ostale vrijednosti parametara jezgre interpolacije. Dakle, za $A_2 = a = -\frac{1}{2}$ vrijedi

$$f(x) - g(x) = 0(h^3). \tag{1.24}$$

Izraz (1.24) nam zapravo ukazuje na veličinu greške interpolacije. Greška interpolacije uniformno teži nuli, proporcionalno h^3. Jednostavnije rečeno, funkcija g je aproksimacija trećeg reda funkcije f. Dobiveni uvjet, $A_2 = -\frac{1}{2}$, jedini pruža takvu mogućnost, za svaki drugi odabir parametra A_2, aproksimacijska funkcije je najviše prvog reda. Korištenjem i konačno dobivenog uvjeta, kubična konvolucijska interpolacijska jezgra se može napisati u sljedećem obliku

$$u(s) = \begin{cases} \frac{3}{2}|s|^3 - \frac{5}{2}|s|^2 + 1 & 0 < |s| < 1 \\ -\frac{1}{2}|s|^3 + \frac{5}{2}|s|^2 - 4|s| + 2 & 1 < |s| < 2 \\ 0 & 2 < |s|. \end{cases} \tag{1.25}$$



Slika 1.1: Graf funkcije kubične konvolucijske jezgre (1.25)

1.3 Rubne vrijednosti

Prilikom definiranja funkcije f, u potpoglavlju 1.1, naveli smo kako je funkcija definirana na nekom skupu $X \subseteq \mathbb{R}$, ali prilikom praktičnog djelovanja, funkciju f promatramo na ograničenom skupu $[a, b]$. Prema tome, čvorovi interpolacije x_k, isključivo pripadaju intervalu $[a, b]$. Neka je $x_k = x_{k-1} + h$, za $k = 1, ...N$, gdje je $x_0 = a$, $x_N = b$ i $h = \frac{(b-a)}{N}$, za neki proizvoljno velik prirodan broj N. U našem slučaju broj N predefiniran je dimenzijom slike koju obrađujemo, to jest brojem piksela u retku, odnosno stupcu. Potpoglavlje 1.1, spominje kako su naši interpolacijski čvorovi ekvidistantni, te zbog toga, ova definicija za x_k ne utječe na prethodni izračun.

Na intervalu $[a, b]$, kubična konvolucijska interpolacija se može zapisati na sljedeći način

$$g(x) = \sum_{k=-1}^{N+1} c_k u\left(\frac{x - x_k}{h}\right), \tag{1.26}$$

a potrebne su nam sljedeće vrijednosti c_k za $k = -1, 0, 1..., N + 1$, kako bi mogli odrediti funkciju g za svaki $x \in [a, b]$. Od prije nam je poznato, kako za $k = 0, 1..., N$, $c_k = f(x_k)$, no što je s vrijednostima za $k = -1$ i $k = N + 1$? x_{-1} i x_{N+1} ne pripadaju definiranom skupu, stoga su nam njihove vrijednosti nepoznate. Dakle, vrijednosti dodijeljene parametrima c_{-1} i c_{N+1} su naši rubni uvjeti. Imajući na umu činjenicu da je naša funkcija $g(x)$ aproksimacijska funkcija, gdje je dopuštena greška reda $0(h^3)$, moramo odrediti takve granične vrijednosti da aproksimiraju funkciju $f(x)$ za svaki x iz danog intervala $[a, b]$, greškom reda $0(h^3)$. Kako bi pronašli lijevu rubnu vrijednost, pretpostavimo da je x element podintervala $[x_0, x_1]$. Za tako definiran x, interpolacijska funkcija je oblika

$$g(x) = c_{-1}u(s + 1) + c_0u(s) + c_1u(s - 1) + c_2u(s - 2), \tag{1.27}$$

pritom je $s = \frac{(x-x_0)}{h}$. Koristeći se interpolacijskom jezgrom iz (1.25), interpolacijska funkcija ima sljedeći oblik

$$g(x) = \frac{s^3[c_2 - 3c_1 + 3c_0 - c_{-1}]}{2} - \frac{s^2[c_2 - 4c_1 + 5c_0 - 2c_{-1}]}{2} + \frac{s[c_1 - c_{-1}]}{2} + c_0. \tag{1.28}$$

No, ako je greška aproksimacijske funkcije g reda $0(h^3)$, tada koeficijent uz s^3 mora biti jednak nuli. Stoga, c_{-1} mora biti izabran tako da vrijedi $c_{-1} = c_2 - 3c_1 + 3c_0$, ili

$$c_{-1} = f(x_2) - 3f(x_1) + 3f(x_0). \tag{1.29}$$

Uvrštavanjem (1.29) u (1.28), interpolacijska funkcija g poprima sljedeći oblik

$$g(x) = \frac{s^2[f(x_2) - 2f(x_1) + f(x_0)]}{2} + \frac{s[-f(x_2) + 4f(x_1) - 3f(x_0)]}{2} + f(x_0). \tag{1.30}$$

Ne preostaje nam ništa drugo, nego pokazati kako je (1.30) aproksimacijska funkcija, trećeg reda, funkcije $f(x)$. Za početak raspišimo Taylorove redove funkcija $f(x_2)$ i $f(x_1)$ oko točke x_0.

$$f(x_2) = f(x_0) + 2f'(x_0)h + 2f''(x_0)h^2 + 0(h^3) \tag{1.31}$$

$$f(x_1) = f(x_0) + f'(x_0)h + \frac{f''(x_0)}{2}h^2 + 0(h^3) \tag{1.32}$$

Uvrštavanjem dobivenih Taylorovih redova za funkcije,(1.31) i (1.32), u (1.30), aproksimacijska funkcija g poprima sljedeći oblik

$$g(x) = f(x_0) + f'(x_0)sh + \frac{f''(x_0)s^2h^2}{2} + 0(h^3). \tag{1.33}$$

Kako je $sh = x - x_0$, razvoj Taylorovog reda funkcije $f(x)$ oko točke x_0 tada je oblika

$$f(x) = f(x_0) + f'(x_0)sh + \frac{f''(x_0)s^2h^2}{2} + 0(h^3). \tag{1.34}$$

Oduzmimo (1.33) od (1.34)

$$f(x) - g(x) = 0(h^3), \tag{1.35}$$

$$R_3(x) = -\frac{1}{6}f^{(3)}(\eta)h^3s(2s^2 - 3s + 1), \tag{1.36}$$

pritom je s $R_3(x)$ zadani oblik greške, za $0 < s < 1$.

Stoga, rubna vrijednost određena izrazom (1.29) rezultira aproksimacijom trećeg reda za $f(x)$ kad $x_0 < x < x_1$.

Analogno, za c_{N+1}, ako je x iz intervala $[x_{N-1}, x_N]$, desna rubna vrijednost

$$c_{N+1} = 3f(x_N) - 3f(x_{N-1}) + f(x_{N-2}) \tag{1.37}$$

daje nam aproksimaciju trećeg reda funkcije $f(x)$ za $x_{N-1} < x < x_N$.

Koristeći se interpolacijskom jezgrom definiranom u (1.25) i rubnim uvjetima (1.29) i (1.37), imamo potpunu definiciju kubične konvolucijske interpolacije.

Neka je $x_k < x < x_{k+1}$, funkcija kubične konvolucijske interpolacije glasi

$$g(x) = \frac{c_{k-1}(-s^3 + 2s^2 - s)}{2} + \frac{c_k(3s^3 - 5s^2 + 2)}{2} + \frac{c_{k+1}(-3s^3 + 4s^2 + s)}{2} + \frac{c_{k+2}(s^3 - s^2)}{2}, \tag{1.38}$$

gdje je, $s = \frac{(x-x_k)}{h}$ i $c_k = f(x_k)$, za $k = 0, 1, 2, ..., N$. Pritom su rubni uvjeti sljedećeg oblika $c_{-1} = 3f(x_0) - 3f(x_1) + f(x_2)$ i $c_{N+1} = 3f(x_N) - 3f(x_{N-1}) + f(x_{N-2})$.

1.4 Usporedba s drugim metodama

Prilikom uspoređivanja različitih interpolacijskih metoda, potrebno je u obzir uzeti mnoge faktore. Ključno svojstvo za usporedbu svakako bi bila preciznost pojedine metode. Neke indikacije preciznosti su vrste funkcija koje se mogu egzaktno rekonstruirati. Primjerice, kubična konvolucijska interpolacijska funkcija egzaktno rekonstruira sve polinome drugog stupnja. Razlog leži u tome što je treća derivacija polinoma drugog stupnja jednaka nuli. Stoga je i greška aproksimacije također jednaka nuli, što je vidljivo u izrazu (1.36), jer je jedan od faktora derivacija trećeg stupnja, pa je samim time i cijeli izraz jednak nuli ukoliko je derivacija trećeg stupnja jednaka nuli. S druge strane, linearna interpolacija rekonstruira funkciju s aproksimacijskom greškom jednakoj nuli samo za polinome stupnja jedan. Relativnu preciznost različitih interpolacijskih metoda možemo promatrati gledajući njihovu brzinu konvergencije. Odnosno, time dobivamo mjeru kojom brzinom se greška aproksimacije približava nuli dok smanjujemo inkrement. Prilikom izvoda algoritma kubične konvolucijske interpolacije uočava se kako greška aproksimacije sadrži članove reda veličine h^3, gdje je h dani inkrement. U tom slučaju, aproksimacijska greška teži nuli barem brzinom kojom h^3 teži nuli. Stoga, kažemo da je mjera brzine konvergencije kubične konvolucijske interpolacije jednaka $0(h^3)$. Imajući na umu činjenicu da ovu problematiku želimo primijeniti u praksi potrebno je voditi računa i o njezinoj efikasnost, odnosno performansi pojedinog algoritma temeljenog na odabranoj metodi. Stoga promotrimo jednu od najbržih i najjednostavnijih metoda, koja je zbog svoje efikasnosti često korištena u praksi unatoč nešto nižoj preciznosti, algoritam zvan *nearest neighbour* (najbliži susjed). Definiciju ovog algoritma možemo pronaći u članku G. R. Dunlopa, [3] na 349. stranici.

Metoda nearest neghbour

Neka je $f(x)$, definirana na intervalu $[a, b]$, neprekidna funkcija koju želimo aproksimirati nad ekvidistantnim intervalima, čija je udaljenost jednaka h. Tada i-ti čvor interpolacije možemo napisati u obliku $x_i = a + ih$, a njegovu vrijednost računamo na zadani način

$$f_i = f(a + ih) \tag{1.39}$$

Za sve točke x, koje se nalaze između čvorova interpolacije vrijednost se određuje na sljedeći način. Neka je $f_{nn}(x)$ vrijednost izračunata upotrebom algoritma najbližeg susjeda, tada

$$f_{nn}(x) = f_i. \tag{1.40}$$

Način na koji računamo i-ti čvor dan je sljedećom definicijom

$$i \leq \frac{x - a}{h} + 0.5 < i + 1. \tag{1.41}$$

Jednostavnije rečeno, ideja ovog algoritma jest da točke koje se nalaze između čvorova interpolacije poprime vrijednost najbližeg čvora, vidljivo na slici 1.2. Zbog toga, konstantna funkcija jest jedina vrsta funkcije koju ovaj algoritam može egzaktno rekonstruirati. Prema tome, $0(h)$ je mjera brzine algoritma najbližeg susjeda.

Teorem 1.4.0.1 (Lagrangeov teorem srednje vrijednosti)**.** *Neka je f realna funkcija neprekidna na konačnom segmentu $[a, b]$ i diferencijabilna na otvorenom intervalu (a, b). Tada postoji barem jedna točka $\xi \in (a, b)$ takva da je*

$$f(b) - f(a) = f'(\xi)(b - a).$$

Prema iskazanom Teoremu 1.4.0.1 pokazat ćemo da je mjera kojom algoritam najbližeg susjeda konvergira nuli iznosi $0(h)$. Neka f ima neprekidnu derivaciju na intervalu $[x, x_j]$, tada postoji točka $m \in [x, x_j]$ takva da

$$f(x) = f(x_j) + f'(m)sh \tag{1.42}$$

gdje je $s = \frac{(x-x_j)}{h}$. Ako je točki x najbliži interpolacijski čvor x_j, tada je vrijednost $f(x)$ jednaka vrijednosti interpolacijske funkcije najbližeg susjeda, odnosno jednaka je $f(x_j)$. Greška aproksimacijske funkcije u (1.42) proporcionalna je inkrementu h, što znači da je mjera konvergencije jednaka $0(h)$.

Linearna interpolacija

Sljedeća jednostavna metoda korištena prilikom digitalne obrade slike je linearna interpolacija. Neka je $f(x)$ neprekidna funkcija, definirana na intervalu $[a, b]$, koju želimo aproksimirati koristeći metodu linearne interpolacije. Kao i u prethodnoj metodi, metodi najbližeg susjeda, funkciju želimo aproksimirati nad ekvidistantnim intervalima čija je međusobna udaljenost jednaka h. Čvorove interpolacije i njihove vrijednosti odabiremo na način definiran izrazom (1.39). Za sve točke x, koje se nalaze između interpolacijskih čvorova vrijednost se određuje na sljedeći način. Neka je vrijednost funkcije f u točki x jednaka $f(x)$, tada je aproksimacijka vrijednost $f_{li}(x)$, dobivena korištenjem metode linearne interpolacije, definirana kao

$$f_{li}(x) = (i + 1 - \frac{x}{h})f_i + (\frac{x}{h} - i)f_{i+1}, \tag{1.43}$$

pritom je indeks i definiran izrazom $x_i \leq x < x_{i+1}$. Drugim riječima, aproksimacijske vrijednosti točaka između dva interpolacijska čvora nalaze se na pravcu koji spaja ta dva čvora, vidljivo na slici 1.2. U uvodnom dijelu ovog potpoglavlja spomenuli smo kako metoda linearne interpolacije egzaktno rekonstruira polinome stupnja jedan. Odnosno, mjera brzine konvergencije linearne interpolacije iznosi $0(h^2)$. Kako bi potkrijepili tu tvrdnju, pretpostavimo da je x točka između interpolacijskih čvorova x_j i x_{j+1}. Te neka je

$s = \frac{x-x_j}{h}$. Ako funkcija ima neprekidnu drugu derivaciju na intervalu $[x_j, x_{j+1}]$, tada primjenom Tayloreva teorema, (1.2.0.1), vrijedi

$$f(x_j) = f(x) - f'(x)sh + 0(h^2). \tag{1.44}$$

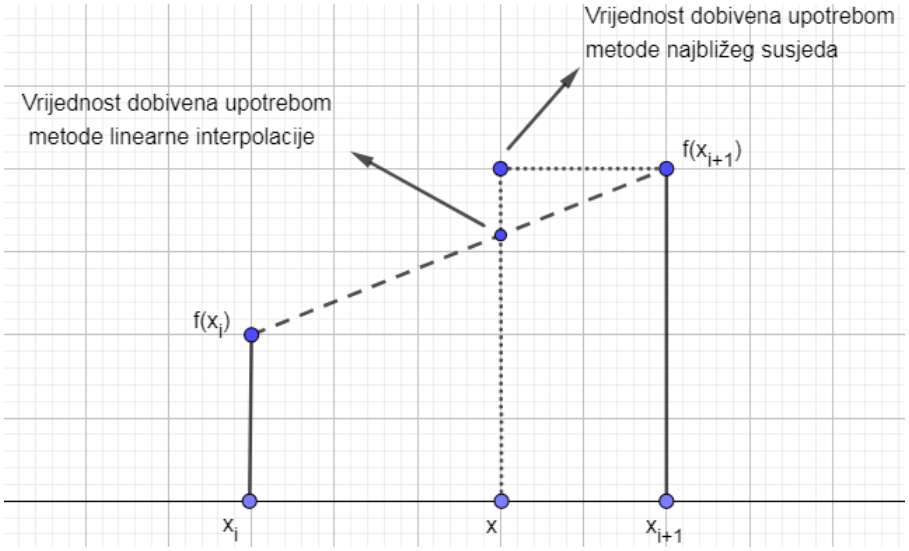
Pritom je aproksimacijska greška dana izrazom $0(h^2)$. No, kako je $x_{j+1} - x = (1 - s)h$, primjenom Tayloreva teorema

$$f(x_{j+1}) = f(x) + f'(x)(1 - s)h + 0(h^2). \tag{1.45}$$

Pomnožimo izraz (1.44) s $(1 - s)$ i izraz (1.45) s s, te ih zbrojimo,

$$(1 - s)f(x_j) + sf(x_j + 1) = f(x) + 0(h^2). \tag{1.46}$$

Vidimo kako je lijeva strana (1.46) zapravo linearna interpolacija funkcije $f(x)$. Druga strana jednakosti (1.46) ukazuje nam kako je greška aproksimacijske funkcije proporcionalna h^2. Prema tome, pokazali smo kako je mjera brzine konvergencije linearne interpolacije jednaka $0(h^2)$. Kako smo u dosadašnjem radu pokazali da je mjera brzine konvergencije kubične konvolucijske funkcije jednaka $0(h^3)$, u globalu od nje možemo očekivati preciznije rezultate prilikom aproksimacije u odnosu na metodu linearne interpolacije.



Slika 1.2: Grafički prikaz aproksimacijskih vrijednosti dobivenih upotrebom metode linearne interpolacije i metodom najbližeg susjeda

Poglavlje 2

Kubična splajn interpolacija

Riječ splajn dolazi od engleske riječi *"spline"*, a označava savitljivo ravnalo, koje se upotrebljavalo pri izgradnji brodova, ili prilikom crtanja različitih glatkih oblika. Usprkos svojoj savitljivosti, ravnalo nije moglo samo zadržavati dani oblik, te bi se zbog toga učvrstilo na predodređenim mjestima dodatnim alatima. Prema tome, ravnalo bi u velikoj mjeri pratilo željeni oblik, dok bi u fiksnim točkama zadržavalo traženu vrijednost, odnosno poziciju. Stoga, sam proces crtanja, odnosno taj aspekt gradnje brodova, zbog fiksnih točaka kojima je linija predodređena uvelike podsjeća na metodu interpolacije. Ako želimo to ravnalo opisati nekom matematičkom funkcijom, zbog svoje savitljivosti i gipkosti gotovo sigurno bi se morali koristiti kakvim polinomom višeg stupnja. Zbog gipkog svojstva splajna, možemo reći da taj proces spada u domenu polinomne interpolacije. Time je zapravo primjena splajn interpolacije poznata i prije svoje teoretske obrade. Tema ovog poglavlja obrađena je na petnaestom predavanju [2], kolegija Numerička analiza, koji se održava na doktorskom studiju matematike u Zagrebu.

2.1 Interpolacija polinomima i po dijelovima polinomna interpolacija

Ulazeći u sferu polinoma viših stupnjeva otvaraju nam se mogućnosti aproksimacije raznih funkcija, no to krije i poneku zamku. Naime, ukoliko nismo sigurni da dani polinom vrlo dobro aproksimira funkciju na cijelom skupu, moguće su anomalije, odnosno velika odstupanja između dvije točke interpolacije koje nisu u blizini. Zbog svojih loših svojstava, uglavnom se u praksi izbjegava interpolacija polinomom visokog stupnja. Kako bi ipak iskoristili poželjna svojstva polinoma, da pomoću njih možemo rekonstruirati izrazito zahtjevne funkcije, interval na kojem ju želimo aproksimirati, podijelimo na podintervale te na svakom od njih koristimo polinom fiksnog, nižeg stupnja. Ako grafički predočimo

takve funkcije sigurno bi se radilo o nekakvoj krivulji, stoga ne čudi ideja o posezanju za polinomima. Splajn interpolacija, pripada grupi interpolacija koja kao svoju aproksimirajuću funkciju koristi polinome na svakom od podintervala. No, prije same definicije splajn interpolacije, potrebno je ponešto reći i o po dijelovima polinomnoj interpolaciji.

Neka je na nekom intervalu $[a, b]$ dana funkcija f, te neka je na danom skupu definirano i $n + 1$ čvorova interpolacije, koji čine particiju ili mrežu, a označavamo ih sa $a = x_0 < x_1 < ... < x_n = b$. Tada za potrebe po dijelovima polinomne interpolacije, na svakom podintervalu određenog čvorovima interpolacije, $[x_{j-1}, x_j]$, koristimo polinom fiksnog stupnja m, takav da je

$$\varphi\big|_{[x_{j-1},x_j]} = p_j,$$

za $j = 1, ..., n$, gdje je $p_j \in \mathcal{P}_m$. Pritom je $\mathcal{P}_m$ standardna oznaka za skup svih polinoma stupnja m. Svaki polinom p_j, stupnja je m te je određen s $m + 1$ koeficijenata. Budući da je n podintervala, te na svakom podintervalu se nalazi jedinstveni polinom fiksnog stupnja m, stoga je $(m+1)\cdot n$ koeficijenata koje moramo odrediti. Imajući na umu činjenicu da funkciju interpoliramo, odmah su nam poznate vrijednosti funkcije φ u čvorovima interpolacije, odnosno

$$\varphi(x_j) = f(x_j),$$

gdje je $j = 0, ..., n$. Prema tome, poznata nam je početna i odredišna vrijednost svakog polinoma na podintervalu $[x_{j-1}, x_j]$, to jest

$$\begin{aligned} p_j(x_{j-1}) &= f(x_{j-1}) \\ p_j(x_j) &= f(x_j), \end{aligned}$$

za $j = 1, ..., n$. Dakle zahvaljujući interpolaciji ukupno imamo $2n$, od mogućih $(m + 1) \cdot n$ uvjeta. Osim samih uvjeta interpolacije, valja napomenuti kako je time osigurana i neprekidnost funkcije φ, jer svaki unutarnji čvor interpolacije element je domene točno dva polinoma

$$p_{j-1}(x_{j-1}) = p_j(x_{j-1}),$$

za $j = 2, ..., n$. Proizlazi da za $m = 1$, imamo dovoljan broj uvjeta za jednoznačno određivanje svakog od $2n$ koeficijenata. Znamo da polinom prvog stupnja zapravo odgovara linearnoj funkciji, stoga imamo sve elemente za primjenu po dijelovima linearne interpolacije. Naravno, kroz ovaj rad smo imali prilike uvjeriti se kako je linearna interpolacija jedna od jednostavnijih metoda, a eventualno poboljšanje bi se moglo postići povećanjem ukupnog broja interpolacijskih čvorova, ali time direktno utječemo na složenost algoritma što je dovoljna indikacija da je potrebno pronaći precizniju metodu. Povećanjem stupnja polinoma, $m > 1$, zasigurno bi riješili taj problem, no time raste i ukupan broj koeficijenata koje je potrebno odrediti. Stoga je nužno dodati uvjete na glatkoću funkcije φ u unutarnjim čvorovima interpolacije. Promotrimo za sada metodu po dijelovima kubične interpolacije, kojom želimo aproksimirati istu funkciju f nad intervalom $[a, b]$, sa pripadajućim

čvorovima interpolacije koji čine mrežu, a označavamo ih s $a = x_0 < x_1 < ... < x_n = b$. Tada na svakom podintervalu $[x_{j-1}, x_j]$ koristimo kubični polinom

$$\varphi\big|_{[x_{j-1},x_j]} = p_j,$$

za $j = 1, ..., n$, gdje je $p_j \in \mathcal{P}_3$. Pritom je $\mathcal{P}_3$ standardna oznaka za skup svih polinoma stupnja 3. Uobičajena je praksa polinome p_j zapisivati relativno obzirom na početnu točku podintervala x_{j-1}, oblika

$$p_j(x) = c_{0,j} + c_{1,j}(x - x_{j-1}) + c_{2,j}(x - x_{j-1})^2 + c_{3,j}(x - x_{j-1})^3, \tag{2.1}$$

za $x \in [x_{j-1}, x_j]$, i za $j = 1, ..., n$. Kao i do sada, promatramo n kubičnih polinoma na isto toliko podintervala te je na svakom od podintervala potrebno odrediti 4 koeficijenta. Prema tome ukupno je potrebno odrediti $4n$ koeficijenata. Uvjet interpolacije nudi nam vrijednosti $2n$ koeficijenata, jer svaki kubični polinom p_j mora interpolirati funkciju f u rubovima svog podintervala $[x_{j-1}, x_j]$, odnosno

$$\begin{aligned} p_j(x_{j-1}) &= f(x_{j-1}) \\ p_j(x_j) &= f(x_j), \end{aligned}$$

za $j = 1, ..., n$, čime je osigurana neprekidnost funkcije φ. Pretpostavimo da je naša interpolacijska funkcija φ barem klase $C^1[a, b]$, odnosno da je njena derivacija neprekidna i u čvorovima interpolacije. Tada, za svaki kubični polinom p_j dobivamo još točno $2n$ uvjeta, odnosno vrijedi

$$\begin{aligned} p_j'(x_{j-1}) &= s_{j-1}, \\ p_j'(x_j) &= s_j, \end{aligned}$$

za $j = 1, ..., n$, pri čemu su s_j neke, za sada nepoznate vrijednosti. Postoji više načina na koji se vrijednosti s_j mogu odrediti, a svaka od njih zapravo čini zasebnu metodu. Zanimljivo je naglasiti da sama notacija s_j nije slučajna, već dolazi od engleske riječi za nagib, *"slope"*. Osim toga, zbog

$$p_{j-1}'(x_{j-1}) = p_j'(x_{j-1}) = s_{j-1},$$

za $j = 1, ..., n$, osigurali smo i dodatan uvjet, neprekidnost prve derivacije funkcije φ u svim unutrašnjim interpolacijskim čvorovima.

2.2 Metoda kubične splajn interpolacije

Konačno, promotrimo i metodu kubične splajn interpolacije. Za potrebe ove metode pretpostavimo da funkcija φ ima neprekidnu drugu derivaciju u unutarnjim čvorovima, koje

označavamo s $x_1, x_2, ..., x_{n-1}$, odnosno da je klase C^2 na danom skupu $[a, b]$. Tada imamo mogućnost jednoznačno odrediti vrijednosti $s_0, ..., s_n$. Usprkos tome, još uvijek nemamo sve potrebno za jednoznačno određivanje svih potrebnih koeficijenata. Naime, kao i do sada potrebno je izračunati koeficijente za n polinoma. Svakom polinomu potrebno je odrediti 4 koeficijenta, što znači da je potrebno ukupno odrediti $4n$ koeficijenata. Za sada, zbog interpolacije osigurano je $2n$ uvjeta. Znamo da na skupu $[a, b]$ imamo $n - 1$ unutarnjih čvorova, stoga takozvanim lijepljenjem prve derivacije u svim unutarnjim čvorovima na raspolaganju imamo još dodatnih $n - 1$ uvjeta. S obzirom na to da je funkcija φ klase C^2, postupak lijepljenja možemo ponoviti i za drugu derivaciju u unutarnjim čvorovima, i priskrbiti si još $n - 1$ uvjeta. Proizlazi da raspolažemo s $4n - 2$ uvjeta, dok je ukupno potrebno odrediti $4n$ koeficijenata. Sve u svemu nedostaju nam još 2 uvjeta kako bi sve koeficijente mogli jednoznačno odrediti. Prema tome poslužit ćemo se sljedećom metodom, lijepljenjem prvih i drugih derivacija.

Proces lijepljenja prve derivacije postiže se zahtjevom

$$\varphi'(x_j) = s_j,$$

pritom ne ulazeći u dublje razmatranje o vrijednosti s_j. Uvjet lijepljenja druge derivacije nešto je drugačiji, i zahtjeva sljedeći uvjet

$$p_j''(x_j) = p_{j+1}''(x_j),$$

za $j = 1, ..., n - 1$. Pridržavamo li se načina zapisa polinoma, prikazano u (2.1), relativno s obzirom na početnu točku podintervala, tada je

$$p_j''(x) = 2c_{2,j} + 6c_{3,j}(x - x_{j-1})$$
$$p_{j+1}''(x) = 2c_{2,j+1} + 6c_{3,j+1}(x - x_j).$$

Poslužimo li se čvorovima interpolacije x_j i podijelimo li uvjet lijepljenja s 2 dobivamo sljedeći izraz

$$c_{2,j} + 3c_{3,j}(x_j - x_{j-1}) = c_{2,j+1}. \tag{2.2}$$

Poželjno je koeficijente $c_{i,j}$ ispisati u terminima f_j i s_j. Da bi to postigli, prvo ćemo razmotriti interpolaciju u kojoj ne zahtijevamo samo interpolaciju zadane vrijednosti funkcije već i interpolaciju njezine prve derivacije, takav oblik interpolacije poznat je kao Hermiteova interpolacija. Stoga ju koristimo nad svakim polinomom p_j. S obzirom na činjenicu da ćemo se poslužiti Hermiteovom metodom, nije na odmet promisliti o zapisu samog polinoma p_j. Metoda kojom se interpolacijskom polinomu dodaju nove točke interpolacije, odnosno kojom se povećava stupanj interpolacijskog polinoma zapisuje se u obliku Newtonovog interpolacijskog polinoma. Interpolacijski polinom stupnja 0 može se opisati konstantom koja interpolira funkciju f u točki x_0, očito je

$$p_0(x) = f_0.$$

Prema tome, za interpolacijski polinom stupnja 1 dodajemo još jedan čvor interpolacije, x_1, tada polinom p_1 možemo zapisati kao polinom p_0 sa pripadajućom korekcijom $r_1(x)$, odnosno

$$p_1(x) = p_0(x) + r_1(x).$$

Pritom je pripadajuća korekcija $r_1(x)$ stupnja 1. Iz uvjeta interpolacije u x_0 slijedi

$$f_0 = p_1(x_0) = p_0(x_0) + r_1(x_0) = f_0 + r_1(x_0)$$

to jest, slijedi da $r_1(x_0) = 0$, pa je r_1 oblika

$$r_1(x) = a_1(x - x_0).$$

Nadalje, iz uvjeta interpolacije u točki x_1 imamo

$$f_1 = p_1(x_1) = p_0(x_1) + r_1(x_1) = f_0 + r_1(x_1),$$

odnosno, vidimo da je $r_1(x_1) = f_1 - f_0$, pa se koeficijent a dobiva kao podijeljena razlika čvorova x_0 i x_1, odnosno

$$a_1 = \frac{f_1 - f_0}{x_1 - x_0}.$$

Kao i u prethodnom slučaju, interpolacijski polinom višeg stupnja, odnosno stupnja 2, dobivamo dodavanjem još jednog čvora interpolacije, čvor x_2. Tada polinom p_2 možemo napisati kao zbroj polinoma p_1 i korekcije r_2,

$$p_2(x) = p_1(x) + r_2(x).$$

Vidljivo je da r_2 mora biti stupnja 2, a iz uvjeta interpolacije čvorova x_0 i x_1 slijedi

$$f_j = p_2(x_j) = p_1(x_j) + r_2(x_j) = f_j + r_2(x_j), \quad j = 0, 1.$$

Prema tome, $r_2(x_j) = 0$, odnosno r_2 je oblika

$$r_2(x) = a_2(x - x_0)(x - x_1),$$

te se koeficijent a_2 računa iz uvjeta interpolacije u x_2. Nastavimo li s ovim postupkom, dolazimo do općeg oblika Newtonovog interpolacijskog polinoma

$$p_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n \prod_{j=0}^{n-1} (x - x_j).$$

Valja napomenuti kako je standardna oznaka koeficijenta $a_j = f[x_0, ..., x_j]$, te ćemo se nje i pridržavati u nastavku ovog rada. Prema [1] na 62. stranici vidimo kako za podijeljene razlike vrijedi sljedeća rekurzija

$$f[x_0, ..., x_n] = \frac{f[x_1, ..., x_n] - f[x_0, ..., x_{n-1}]}{x_n - x_0}, \tag{2.3}$$

s tim da je

$$f[x_j] = f_j.$$

Prikažimo i tablicu koja sadrži sve koeficijente, odnosno podijeljene razlike. Za potrebe računanja Newtonovog polinoma koristimo samo podijeljene razlike koje nalaze na gornjoj dijagonali, a zbog lakše uočljivosti obojane su crvenom bojom.

x_j	$f[x_j]$	$f[x_j, x_{j+1}]$	$f[x_j, x_{j+1}, x_{j+2}]$	...	$f[x_0, ..., x_n]$
x_0	$f[x_0]$				
		$f[x_0, x_1]$			
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$		
		$f[x_1, x_2]$		$\ddots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$f[x_0, ..., x_n]$
		$f[x_{n-2}, x_{n-1}]$		$\iddots$	
x_{n-1}	$f[x_{n-1}]$		$f[x_{n-2}, x_{n-1}, x_n]$		
		$f[x_{n-1}, x_n]$			
x_n	$f[x_n]$				

Tablica 2.1: Tablica svih podijeljenih razlika za Newtonov polinom

Dakle, možemo napisati i konačan opći izgled Newtonovog polinoma

$$\begin{aligned} p_n(x) =& f[x_0] + f[x_0, x_1](x - x_0) \\ &+ f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &+ \cdots + f[x_0, x_1, ..., x_n](x - x_0) \cdots (x - x_{n-1}), \end{aligned}$$

uz pripadajuću grešku, u nekoj točki $x \in \langle a, b \rangle$, koja je različita od svih interpolacijskih točaka, $x_0, ..., x_n$,

$$f(x) - p_n(x) = \frac{\omega(x)}{(n+1)!} f^{(n+1)}(\xi), \tag{2.4}$$

za neki $\xi \in (x_{min}, x_{max})$. Pritom je $x_{min} = min\{x_0, ..., x_n, x\}$, dok je $x_{max} = max\{x_0, ..., x_n, x\}$. Napomenimo kako se oznaka $\omega(x)$ odnosi na polinom čvorova, to jest

$$\omega(x) = \prod_{j=0}^{n} (x - x_j).$$

Izraz greške (2.4), otkriva nam i sljedeću formulu

$$f[x_0, x_1, ..., x_n, x] = \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

koja vrijedi i u slučaju kad čvorovi međusobno nisu različiti.

Zbog neposredno iskazanih činjenica, koristit ćemo Newtonov oblik interpolacijskog polinoma p_j, u takozvanim dvostrukim čvorovima x_{j-1} i x_j. U svakom od čvorova zadajemo po dva podatka, vrijednost funkcije i vrijednost njene derivacije. Pritom, razmak između dva različita susjedna čvora označavamo s $h_j = x_j - x_{j-1}$, za $j = 1, ..., n$. Neka je $f[x_j, x_j + h]$ podijeljena razlika ta dva čvora, i ako se oni međusobno približavaju, odnosno ako $h \to 0$, tada je

$$\lim_{h\to 0} f[x_j, x_j + h] = \lim_{h\to 0} \frac{f(x_j + h) - f(x_j)}{h} = f'(x_j), \tag{2.5}$$

ali, uz uvjet da funkcija f ima derivaciju u točki x_j. Dakle, vrijedi

$$f[x_j, x_j] = f'(x_j).$$

Kako se u našem slučaju derivacija u točki x_j zadaje sa s_j, tada je

$$f[x_j, x_j] = s_j$$

Stoga, tablica podijeljenih razlika za Hermiteov interpolacijski polinom p_j kojemu su x_{j-1} i x_j dva dvostruka čvora, ima sljedeći oblik

t_j	$f[t_j]$	$f[t_j, t_{j+1}]$	$f[t_j, t_{j+1}, t_{j+2}]$	$f[t_j, t_{j+1}, t_{j+2}, t_{j+3}]$
x_{j-1}	f_{j-1}			
		s_{j-1}		
x_{j-1}	f_{j-1}		$\frac{f[x_{j-1},x_j]-s_{j-1}}{h_j}$	
		$f[x_{j-1}, x_j]$		$\frac{s_j+s_{j-1}-2f[x_{j-1},x_j]}{h_j^2}$
x_j	f_j		$\frac{s_j-f[x_{j-1},x_j]}{h_j}$	
		s_j		
x_j	f_j			

Tablica 2.2: Tablica podijeljenih razlika za Hermiteov interpolacijski polinom

Prema tome, sada možemo napisati i Newtonov oblik Hermiteovog interpolacijskog polinoma p_j kojemu su x_{j-1} i x_j dva dvostruka čvora

$$\begin{aligned} p_j(x) =& f[x_{j-1}] + f[x_{j-1}, x_{j-1}] \cdot (x - x_{j-1}) \\ &+ f[x_{j-1}, x_{j-1}, x_j] \cdot (x - x_{j-1})^2 \\ &+ f[x_{j-1}, x_{j-1}, x_j, x_j] \cdot (x - x_{j-1})^2 (x - x_j), \end{aligned} \tag{2.6}$$

pritom je

$$\begin{aligned} f[x_{j-1}, x_{j-1}] &= s_{j-1}, \\ f[x_{j-1}, x_{j-1}, x_j] &= \frac{f[x_{j-1}, x_j] - s_{j-1}}{h_j}, \\ f[x_{j-1}, x_{j-1}, x_j, x_j] &= \frac{s_j + s_{j-1} - 2f[x_{j-1}, x_j]}{h_j^2}. \end{aligned}$$

Uvrstimo li čvorove x_j i x_{j-1} u (2.6), jasno je da vrijedi

$$\begin{aligned} p_j(x_{j-1}) = f_{j-1}, \quad p_j'(x_{j-1}) = s_{j-1}, \\ p_j(x_j) = f_j, \quad p_j'(x_j) = s_j. \end{aligned}$$

Prema tome, pronašli smo oblik traženog polinoma p_j, na svakom podintervalu $[x_{j-1}, x_j]$, za $j = 1, ..., n$. Kako smo cijeli ovaj proces proveli kako bi koeficijente $c_{i,k}$ ispisati u terminima f_j i s_j, raspišimo Newtonov oblik polinoma p_j u terminima potencija $(x - x_{j-1})$. Za trenutak promotrimo zapis posljednjeg člana polinoma p_j, odnosno $(x - x_{j-1})^2(x - x_j)$, njega možemo zapisati u sljedećem obliku

$$\begin{aligned} (x - x_{j-1})^2(x - x_j) &= (x - x_{j-1})^2(x - x_{j-1} + x_{j-1} - x_j) \\ &= (x - x_{j-1})^2(x - x_{j-1} - h_j) \\ &= (x - x_{j-1})^3 - h_j(x - x_{j-1})^2 \end{aligned}$$

Dakle, polinom p_j tada možemo zapisati u sljedećem obliku

$$\begin{aligned} p_j(x) =& f[x_{j-1}] + f[x_{j-1}, x_{j-1}] \cdot (x - x_{j-1}) \\ &+ (f[x_{j-1}, x_{j-1}, x_j] - h_j f[x_{j-1}, x_{j-1}, x_j, x_j]) \cdot (x - x_{j-1})^2 \\ &+ f[x_{j-1}, x_{j-1}, x_j, x_j] \cdot (x - x_{j-1})^3. \end{aligned}$$

Usporedimo li koeficijente uz odgovarajuće potencije binoma $(x - x_{j-1})$, dobijemo sljedeće vrijednosti

$$\begin{aligned} c_{0,j} &= p_j(x_{j-1}) = f_{j-1}, \\ c_{1,j} &= p_j'(x_{j-1}) = s_{j-1}, \\ c_{2,j} &= \frac{p_j''(x_{j-1})}{2} = f[x_{j-1}, x_{j-1}, x_j] - h_j f[x_{j-1}, x_{j-1}, x_j, x_j], \\ c_{3,j} &= \frac{p_j^{(3)}(x_{j-1})}{6} = f[x_{j-1}, x_{j-1}, x_j, x_j]. \end{aligned} \tag{2.7}$$

Obratimo li pozornost na oblik posljednja dva izraza, logika nalaže da prvo izračunamo vrijednost koeficijenta $c_{3,j}$, te dobiveni rezultat iskoristimo pri određivanju koeficijenta $c_{2,j}$. Prema tome, zapišimo i konačne izraze za izračun koeficijenata $c_{i,j}$, u standardnom zapisu polinoma p_j, i to redom kojim ih računamo

$$\begin{aligned} c_{0,j} &= f_{j-1}, \\ c_{1,j} &= s_{j-1}, \\ c_{3,j} &= \frac{s_j + s_{j-1} - 2f[x_{j-1}, x_j]}{h_j^2}, \qquad (2.8) \\ c_{2,j} &= \frac{f[x_{j-1}, x_j] - s_{j-1}}{h_j} - h_j c_{3,j}, \qquad (2.9) \end{aligned}$$

za $j = 1, ..., n$. Stoga, iskoristimo prethodno izračunate izraze (2.9) i (2.8), te ih uvrstimo u uvjet lijepljenja (2.2), koji nakon dodatnog sređivanja ima sljedeći oblik

$$\frac{-3f[x_{j-1}, x_j] + s_{j-1} + 2s_j}{h_j} = \frac{3f[x_j, x_{j+1}] - 2s_j - s_{j+1}}{h_{j+1}},$$

ali ovog puta za $j = 1, ...n - 1$, jer uvjet lijepljenja provodimo isključivo na unutarnjim čvorovima. Sredimo li prethodni izraz množenjem s $h_j h_{j+1}$ i grupiranjem članova koji sadrže element s_j, proizlazi sljedeće

$$h_{j+1}s_{j-1} + 2(h_j + h_{j+1})s_j + h_j s_{j+1} = 3(h_{j+1}f[x_{j-1}, x_j] + h_j f[x_j, x_{j+1}]). \qquad (2.10)$$

Prema tome novonastali linearni sustav sadrži $n + 1$ nepoznanicu i $n - 1$ jednadžbu. No, zadavanjem vrijednosti početnog i krajnjeg nagiba, odnosno s_0 i s_n, naš sustav broji $n - 1$ nepoznanicu. Matričnim prikazom tog sustava vidimo da se radi o tridijagonalnoj matrici

$$\begin{bmatrix} 2(h_1 + h_2) & h_1 & & & & \\ h_3 & 2(h_2 + h_3) & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-1} & 2(h_{n-2} + h_{n-1}) & h_{n-2} & \\ & & & h_n & 2(h_{n-1} + h_n) & \end{bmatrix}.$$

Promatrajući retke matrice, uočavamo i sljedeće svojstvo koje vrijedi

$$2(h_j + h_{j+1}) > h_j + h_{j+1},$$

što znači da je riječ o strogo dijagonalno dominantnoj matrici, pa je samim time matrica i regularna. Prema tome, ovaj sustav sasvim sigurno ima jednoznačno rješenje $s_1, ..., s_{n-1}$. Kako bi odredili rješenja koristit ćemo se takozvanom LR faktorizacijom bez pivotiranja.

Pretpostavimo da je A neka tridijagonalna matrica, nad kojom je moguće provesti metodu LR faktorizacije bez pivotiranja,

$$A = \begin{bmatrix} d_1 & e_1 & & & \\ c_2 & d_2 & e_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & d_{n-1} & e_{n-1} \\ & & & c_n & d_n \end{bmatrix}.$$

Tada su matrice L i R oblika

$$L = \begin{bmatrix} 1 & & & & \\ l_2 & 1 & & & \\ & \ddots & \ddots & & \\ & & l_{n-1} & 1 & \\ & & & l_n & 1 \end{bmatrix},$$

$$R = \begin{bmatrix} r_1 & e_1 & & & \\ & r_2 & e_2 & & \\ & & \ddots & \ddots & \\ & & & r_{n-1} & e_{n-1} \\ & & & & r_n \end{bmatrix}.$$

Pritom su dijagonale iznad glavne dijagonale jednake za matrice A i R. Ostali se elementi matrica L i R računaju sljedećim rekurzijama

$$\begin{aligned} r_1 &= d_1, \\ l_i &= \frac{c_i}{r_{i-1}}, \\ r_i &= d_i - l_i e_i, \end{aligned}$$

za $i = 2, ..., n$. Primijetimo kako s_j više nisu međusobno nezavisni, već ovise jedan o drugome. Proizlazi da aproksimacija više nije lokalna, već se promjenom vrijednosti samo jedne točke mijenjaju i svi polinomi.

2.3 Rubne vrijednosti

Prilikom određivanja vrijednosti koeficijenata polinoma, funkciju smo promatrali u unutarnjim čvorovima, te smo od nje zahtijevali da je u tim čvorovima klase C^2. Time smo

priskrbili dodatnih $n - 1$ uvjeta, te smo imali mogućnost primijeniti metodu Hermiteove interpolacije i odrediti vrijednosti koeficijenata $c_{i,j}$. Stoga, vrijednosti s_0 i s_n su ostale nedefinirane, i upravo one nedostaju kako bi u cijelosti jednoznačno odredili metodu kubične splajn interpolacije. Njihove vrijednosti se ne zadaju direktno, već se zadaju rubni uvjeti na funkciju φ iz kojih slijede vrijednosti s_0 i s_n.

Prvi način na koji možemo zadati rubne uvjete naziva se potpuni (kompletni) splajn. Kako bi mogli primijeniti ovu metodu potrebna nam je vrijednost prve derivacije funkcije f u njezinim rubovima, koju pridružimo vrijednostima s_0 i s_n. Dakle,

$$\begin{aligned} s_0 &= f'(x_0), \\ s_n &= f'(x_n). \end{aligned}$$

Pritom je greška aproksimacije u funkcijskoj vrijednosti jednaka $0(h^4)$. Napomenimo, ako nam oblik same funkcije f nije poznat, no poznate su nam vrijednosti funkcije f u konačno mnogo točaka, vrijednost prve derivacije funkcije f očigledno nije moguće odrediti direktnom derivacijom, stoga vrijednost prve derivacije funkcije f aproksimiramo korištenjem metode numeričke derivacije.

Drugi način na koji možemo odrediti vrijednosti s_0 i s_n zahtijeva vrijednost druge derivacije funkcije f u njezinim rubovima. Zadaje se

$$\begin{aligned} f''(x_0) &= \varphi''(x_0) = p_1''(x_0), \\ f''(x_n) &= \varphi''(x_n) = p_n''(x_n). \end{aligned}$$

Preostaje još drugu derivaciju polinoma p_1 i p_n izraziti u terminima s_0 i s_1, odnosno s_{n-1} i s_n. Zbog (2.7) znamo da vrijedi

$$c_{2,1} = \frac{p_1''(x_0)}{2} = \frac{f''(x_0)}{2},$$

odnosno,

$$\frac{3f[x_0, x_1] - 2s_0 - s_1}{h_1} = \frac{f''(x_0)}{2}.$$

Dodatnim sređivanjem slijedi

$$2s_0 + s_1 = 3f[x_0, x_1] - \frac{h_1}{2} f''(x_0).$$

Ovu jednadžbu dodajemo u naš linearni sustav, i to na prvo mjesto. Time sada naš linearni sustav broji n nejednadžbi i $n + 1$ nepoznanicu. Preostaje još odraditi posljednju jednadžbu kako bi imali sustav s istim brojem jednadžbi i nepoznanica. Slično kao i za p_1, vrijedi

$$p_n''(x_n) = 2c_{2,n} + 6c_{3,n}h_n,$$

uvrstimo li izraze za $c_{2,n}$ i $c_{3,n}$ slijedi

$$s_{n-1} + 2s_n = 3f[x_{n-1}, x_n] + \frac{h_n}{2} f''(x_n). \tag{2.11}$$

Uvrstimo li (2.11) u dobiveni linearni sustav, na posljednje mjesto, postigli smo traženo, sustav s $n + 1$ jednadžbom i isto toliko nepoznanica, što znači da je moguće jednoznačno odrediti rješenje. Kao i u prethodnom slučaju greška u funkcijskoj vrijednosti je $0(h^4)$.

Sljedeća metoda naziva se prirodni splajn. Koristi se takozvanim slobodnim krajevima, točnije rečeno zahtijeva da vrijednost druge derivacije funkcije φ bude jednaka nuli, odnosno

$$\varphi''(x_0) = \varphi''(x_n) = 0.$$

Prema tome jednadžbe koje uvrštavamo u linearni sustav tada su oblika

$$\begin{aligned} 2s_0 + s_1 &= 3f[x_0, x_1], \\ s_{n-1} + 2s_n &= 3f[x_{n-1}, x_n]. \end{aligned}$$

Greška funkcije ovog puta ovisi o dva slučaja. Ako funkcija f nema drugu derivaciju na rubovima jednaku 0, tada je greška u funkcijskoj vrijednosti jednaka $0(h^2)$. Za razliku od toga, ukoliko funkcija f ipak ima drugu derivaciju u rubnim vrijednostima jednaku 0, tada je greška u funkcijskoj vrijednosti kao i u prethodnoj metodi, odnosno $0(h^4)$.

Sljedeća metoda koja se koristi je takozvani *"not-a-knot"* (nije čvor) splajn. Za razliku od prethodnih metoda, nisu nam potrebne informacije o derivaciji funkcije f na rubovima. Umjesto aproksimacije, koristi se nije čvor uvjet, odnosno parametre s_0 i s_n odabiremo tako da su prva dva i posljednja dva kubična polinoma jednaka, to jest

$$p_1 = p_2, \quad p_{n-1} = p_n.$$

Drugim riječima, u čvoru x_1 zalijepi se i treća derivacija polinoma p_1 i p_2, a u čvoru x_{n-1} zalijepi se treća derivacija polinoma p_{n-1} i p_n. Opisane zahtjeve možemo zapisati na sljedeći način

$$p_1^{(3)}(x_1) = p_2^{(3)}(x_1), \quad p_{n-1}^{(3)}(x_{n-1}) = p_n^{(3)}(x_{n-1}).$$

Prethodni zahtjev, $p_1^{(3)}(x_1) = p_2^{(3)}(x_1)$, zapravo znači da su vodeći koeficijenti polinoma p_1 i p_2 jednaki, to jest

$$c_{3,1} = c_{3,2}.$$

Iskoristimo li taj zahtjev, zahtjevu lijepljenja druge derivacije,

$$c_{2,1} + 3c_{3,1}h_j = c_{2,2},$$

slijedi

$$\frac{f[x_0, x_1] - s_0}{h_1} + 2\frac{s_1 + s_0 - 2f[x_0, x_1]}{h_1} = \frac{f[x_1, x_2] - s_1}{h_2} - h_2\frac{s_1 + s_0 - 2f[x_0, x_1]}{h_1^2}.$$

Dodatnim sređivanjem izraza dobivamo prvu jednadžbu linearnog sustava

$$h_2 s_0 + (h_1 + h_2)s_1 = \frac{(h_1 + 2(h_1 + h_2))h_2 f[x_0, x_1] + h_1^2 f[x_1, x_2]}{h_1 + h_2}$$

Na sličan se način dobije i posljednja jednadžba linearnog sustava

$$(h_{n-1} + h_n)s_{n-1} + h_{n-1}s_n = \frac{(h_n + 2(h_{n-1} + h_n))h_{n-1} f[x_{n-1}, x_n] + h_n^2 f[x_{n-2}, x_{n-1}]}{h_{n-1} + h_n}.$$

Pritom je greška aproksimacije za funkcijske vrijednosti jednaka $0(h^4)$ Važno je napomenuti kako prilikom korištenja svake od navedenih metoda odabira rubnih uvjeta zadržana tridijagonalna struktura linearnog sustava za nepoznate parametre s_j.

2.4 Greška kubične splajn interpolacije

Neka je funkcija f klase C^2 na intervalu $[a, b]$. Pretpostavimo da funkcija f ima ograničenu i integrabilnu četvrtu derivaciju na svakom podintervalu $[x_{j-1}, x_j]$. Tada grešku funkcije φ u odnosu na funkciju f možemo opisati kao

$$\begin{aligned}
\|f(x) - \varphi(x)\|_\infty &\leq \frac{5}{384}h^4\|f^{(4)}\|_\infty, \\
\|f'(x) - \varphi'(x)\|_\infty &\leq \frac{1}{24}h^3\|f^{(4)}\|_\infty, \\
\|f''(x) - \varphi''(x)\|_\infty &\leq \frac{3}{8}h^2\|f^{(4)}\|_\infty, \\
\|f^{(3)}(x) - \varphi^{(3)}(x)\|_\infty &\leq \frac{1}{2}\left(\frac{1}{\beta} + \beta\right)h\|f^{(4)}\|_\infty,
\end{aligned}$$

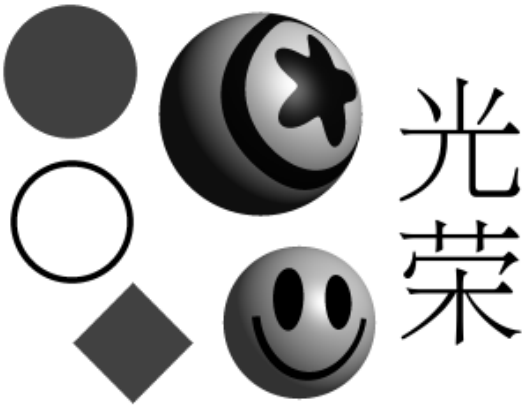
gdje je $\beta = \frac{(max_j h_j)}{(min_j h_j)}$, definirana kao mjera neuniformnosti mreže. Budući da su čvorovi interpolacije u našem slučaju ekvidistantni, i sama mreža je uniformna. Prema tome, $\beta = 1$, stoga je greška oblika

$$\|f^{(3)}(x) - \varphi^{(3)}(x)\|_\infty \leq h\|f^{(4)}\|_\infty.$$

Poglavlje 3

Primjena metoda

Slika 3.1: Barbara



Slika 3.2: Shapes

Želimo li primijeniti obrađene metode, potrebna je konverzija slike u odgovarajući format. Slike odabrane za primjenu metoda su crno-bijele, stoga je svaki piksel reprezentiran 8-bitnom vrijednosti, u intervalu [0 – 255]. Prema tome, vrijednosti svakog pojedinog piksela spremamo u dvodimenzionalno polje, čime smo si osigurali diskretan skup podataka za primjenu metoda. Kako smo naveli u potpoglavlju (1.1), dvodimenzionalnu interpolaciju postižemo interpolacijom u jednoj dimenziji, odnosno po retcima dvodimenzionalnog polja, a zatim u drugoj dimenziji, to jest, po njegovim stupcima. Napomenimo kako su programska rješenja realizirana koristeći se *Python* programskim jezikom, te su za konverziju slika korištene biblioteke *Numpy* i *cv2*. Slike 3.1, 3.2, 3.3 i 3.4 uobičajeni su primjeri

nad kojima se provode testiranja. Sve slike uvećali smo koeficijentom 2, te ćemo prikazati detalje na kojima se mogu uočiti razlike prilikom primjena različitih metoda.

Slika 3.3: Monarch

Slika 3.4: Lena

3.1 Najbliži susjed

Unatoč tome što metoda najbliži susjed ne spada domenu interpolacija, primjena algoritma u dvije dimenzije provodi se na analogan način. Dakle, algoritam za početak primjenjujemo u jednoj dimenziji, po retcima, a nakon toga i u drugoj dimenziji, po stupcima. Ova metoda je očekivano dala najlošije rezultate. na primjerima 3.5 i 3.6 možemo uočiti kako zakrivljene linije imaju nazubljen oblik.

Ovdje se nalazi i programski kod korišten prilikom uvećanja slike.

```
import cv2
import numpy as np

def nearest(data,n):
    new_width = n * len(data[0]) #novi broj stupaca
    new_height = n * len(data) #novi broj redaka
    #korak između nove dvije točke po retcima
    h_width = (len(data[0])-1)/(new_width - 1)
    #korak između nove dvije točke po stupcima
```

Slika 3.5: Barbara

Slika 3.6: Shapes

```
h_height = (len(data)-1)/(new_height - 1)
data_temp = []
row_temp = []
data_new = []

#jednodimenzionalna interpolacija po retcima
for i in range(len(data)):
    for j in range(new_width):
        #posljednji čvor interpolacije dodajemo naknadno
        if(j != new_width-1):
            # index interpolacijskog čvora
            x_j = int(h_width*j)
            x = j*h_width
            #ispitujemo kojem interpolacijskom
            # čvoru je točka bliža
            # i na taj način određujemo vrijednost
            if(x - x_j <= x_j+1 - x ):
                row_temp.append(data[i][x_j])
            else:
                row_temp.append(data[i][x_j+1])

    row_temp.append(data[i][len(data[0])-1])
```

```
        data_temp.append(row_temp)
        row_temp = []

    #kreiranje dodimenzionalnog polja dimenzija nove slike
    for i in range(new_height):
        for j in range(new_width):
            row_temp.append(None)
        data_new.append(row_temp)
        row_temp = []

    #primjena algoritma i u drugoj dimenziji, po stupcima
    for i in range(new_width):
        for j in range(new_height):
            # posljednji čvor interpolacije dodajemo naknadno
            if(j != new_height - 1):
                # index interpolacijskog čvora
                x_j = int(h_height*j)
                x = j*h_width

                #ispitujemo kojem interpolacijskom
                # čvoru je točka bliža
                # i na taj način određujemo vrijednost
                if(x - x_j <= x_j+1 - x ):
                    data_new[j][i] = data_temp[x_j][i]
                else:
                    data_new[j][i] = data_temp[x_j+1][i]
            else:
                data_new[j][i] = data_temp[len(data)-1][i]

    return(data_new)

#vrijednosti pixela slike spremamo u dvodimenzionalno polje
#slike su crno-bijele stoga je vrijednost pixela integer [0-255]
data =cv2.imread('shapes.png',0)

#pokretanje funkcije nearest neighbour s danim faktorom,
#i kreiranje nove slike
cv2.imwrite('shapes_nearest.png',np.uint8(nearest(data,2)))
```

3.2 Linearna interpolacija

Metoda linearne interpolacije pružila je ipak nešto bolje rezultate, kao što je vidljivo na primjerima 3.7 i 3.8, zakrivljene linije ovog su puta nešto manje zrnate.

Slika 3.7: Barbara

Slika 3.8: Shapes

Ovdje se nalazi i programski kod korišten prilikom uvećanja slike.

```
    import cv2
import numpy as np

def linearna_interpolacija(data,n):
    new_width = n * len(data[0]) #novi broj stupaca
    new_height = n * len(data) #novi broj redaka

    #korak između nove dvije točke po retcima
    h_width = (len(data[0])-1)/(new_width - 1)
    #korak između nove dvije točke po stupcima
    h_height = (len(data)-1)/(new_height - 1)
    data_temp = []
    row_temp = []
    data_new = []

    #jednodimenzionalna interpolacija po retcima
```

```
    for i in range(len(data)):
        for j in range(new_width):
            #posljednji čvor interpolacije dodajemo naknadno
            if(j != new_width-1):
                # index interpolacijskog čvora
                x_j = int(h_width*j)
                x = j*h_width
                #računanje vrijednosti svake točke,
                # i konverzija u tip int
                li = (x_j + 1 - x)*data[i][x_j]
                    + (x - x_j)*data[i][x_j+1]
                row_temp.append(li)
            else:
                row_temp.append(data[i][len(data[0])-1])
        data_temp.append(row_temp)
        row_temp = []

    #kreiranje dodimenzionalnog polja dimenzija nove slike
    for i in range(new_height):
        for j in range(new_width):
            row_temp.append(None)
        data_new.append(row_temp)
        row_temp = []

    #primjena algoritma i u drugoj dimenziji, po stupcima
    for i in range(new_width):
        for j in range(new_height):
            # posljednji čvor interpolacije dodajemo naknadno
            if(j != new_height - 1):
                # index interpolacijskog čvora
                x_j = int(h_height*j)
                x = j*h_width

                li = (x_j+1-x)*data_temp[x_j][i]
                    +(x-x_j)* data_temp[x_j+1][i]
                data_new[j][i] = int(li+0.5)
            else:
                data_new[j][i] = int(data_temp[len(data)-1][i]+0.5)
    return(data_new)
```

```
#vrijednosti pixela slike spremamo u dvodimenzionalno polje
#slike su crno-bijele stoga je vrijednost pixela integer [0-255]
data =cv2.imread('shapes.png',0)

#pokretanje funkcije kubicne konvolucijske interpolacije
# s danim faktorom, i kreiranje nove slike
cv2.imwrite('shapes_LI.png',np.uint8(linearna_interpolacija(data,2)))
```

3.3 Kubična konvolucijska interpolacija

Metoda kubične konvolucijske interpolacije rezultira nešto jasnijim slikama, te na primjerima 3.9 i 3.10 vidimo kako su zakrivljene linije lijepo zaobljene.

Slika 3.9: Barbara

Slika 3.10: Shapes

Ovdje se nalazi i programski kod korišten prilikom uvećanja slike.

```
import numpy as np
import cv2

#funkcija koja računa vrijednost pojedinog elementa,
#zadana formulom kubične konvolucijske interpolacije
def g(c_1,c_2,c_3,c_4,s):
    s_2 = s*s
```

```
    s_3 = s*s*s
    return((c_1 * (-s_3+2*s_2-s))/2 + (c_2*(3*s_3-5*s_2+2))/2 +
           (c_3*(-3*s_3 + 4*s_2 + s))/2 + (c_4*(s_3-s_2))/2)

# funkcija vraća integer iz domene 0-255
def domena(a):
    if(a + 0.5 < 0 ):
        return 0
    if(a + 0.5 > 255):
        return 255
    return int(a+0.5)

def kki(data, n):
    new_width = n * len(data[0]) #novi broj stupaca
    new_height = n * len(data) #novi broj redaka
    #dvodimenzionalno polje u kojeg spremamo podatke
    #nakon jednodimenzionalne interpolacije
    data_temp =[]
    row_temp = []
    data_new = [] #dvodimenzionalno polje, završni podaci
    #korak između nove dvije točke po retcima
    h_width = (len(data[0])-1)/(new_width-1)
    #korak između nove dvije točke po stupcima
    h_height = (len(data)-1)/(new_height -1)

    #jednodimenzionalna interpolacija po retcima
    for i in range(len(data)):
        for j in range(new_width):
            #posljednji čvor interpolacije dodajemo naknadno
            if(j != new_width-1):

                x_j = int(h_width*j) # index interpolacijskog čvora
                s = (j*h_width - x_j)

                if(x_j == 0): #lijevi rubni uvjet
                    c_1 = data[i][2] - 3*data[i][1] + 3*data[i][0]
                else:
                    c_1 = data[i][x_j-1]
```

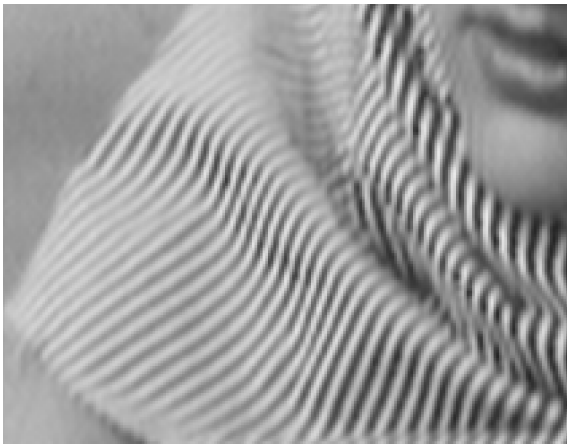
```
                if(x_j >= (len(data[0])-2)): #desni rubni uvjet
                    c_4 = 3*data[i][x_j+1] - 3*data[i][x_j]
                        + data[i][x_j-1]
                else:
                    c_4 = data[i][x_j+2]

                c_2 = data[i][x_j]
                c_3 = data[i][x_j+1]

                #izračunatu vrijednost dodajemo
                #u privremeno jednodimenzionalno polje
                row_temp.append(g(c_1,c_2,c_3,c_4,s))
            else:
                #dodajemo posljednji interpolacijski
                # čvor u privremeno polje
                row_temp.append(data[i][len(data[0])-1])

        #dvodimenzionalno polje koje
        #sadrži vrijednosti interpolacije po retcima
        data_temp.append(row_temp)
        row_temp =[]

    #kreiranje dodimenzionalnog polja dimenzija nove slike
    for i in range(new_height):
        for j in range(new_width):
            row_temp.append(None)
        data_new.append(row_temp)
        row_temp = []

    #primjena algoritma i u drugoj dimenziji, po stupcima
    for i in range(new_width):
        for j in range(new_height):
            # posljednji čvor interpolacije dodajemo naknadno
            if(j != new_height - 1):
                # index interpolacijskog čvora
                x_j = int(h_height*j)
                s = (j*h_height-x_j)

                if(x_j == 0): #lijevi rubni uvjet
```

```
                    c_1 = data_temp[2][i]
                    - 3*data_temp[1][i] + 3*data_temp[0][i]
                else:
                    c_1 = data_temp[x_j-1][i]

                if(x_j >= len(data_temp)-2): #desni rubni uvjet
                    c_4 = 3*data_temp[x_j+1][i]
                        - 3*data_temp[x_j][i] + data_temp[x_j-1][i]
                else:
                    c_4 = data_temp[x_j+2][i]

                c_2 = data_temp[x_j][i]
                c_3 =data_temp[x_j+1][i]

                #spremanje vrijednosti u novu sliku,
                #pritom pazeci na domenu [0-255]
                data_new[j][i] = domena(g(c_1,c_2,c_3,c_4,s))
            else:
                data_new[j][i] = domena(data_temp[len(data)-1][i])

    return(data_new)

#vrijednosti pixela slike spremamo u dvodimenzionalno polje
#slike su crno-bijele stoga je vrijednost pixela integer [0-255]
data =cv2.imread('shapes.png',0)

#pokretanje funkcije kubicne konvolucijske
#interpolacije s danim faktorom, i kreiranje nove slike
cv2.imwrite('shapes_kki.png',np.uint8(kki(data,2)))
```

3.4 Kubična splajn interpolacija

Prilikom primjene ove metode korišten je poznati *Thomasov algoritam*, [11], za rješavanje linearnih sustava sa tridijagonalnim matricama sa sljedeće stranice [12]. Spomenuli smo u potpoglavlju (2.3) kako postoji više načina za odabir rubnih vrijednosti te je ovaj algoritam baziran na prirodnom splajnu. Na primjerima 3.11 i 3.12 vidimo kako su kao i kod prethodnog algoritma, kubične konvolucijske interpolacije, zakrivljene linije lijepo zaobljene.

Ovdje se nalazi i programski kod korišten prilikom uvećanja slike.



Slika 3.11: Barbara

Slika 3.12: Shapes

```
import cv2
import numpy as np

# funkcija vraća integer iz domene 0-255
def domena(a):
    if(a + 0.5 < 0 ):
        return 0
    if(a + 0.5 > 255):
        return 255
    return int(a+0.5)

#prva podijeljena razlika
#razmak je uvijek 1 pa nema potrebe za nazivnikom
def podijeljene(a):
    x=[]
    for i in range(len(a)-1):
        #članovi polja a su tipa uint8, odnosno [0-255],
        #stoga je potrebna konverzija u tipa int
        x.append(int(a[i+1])-int(a[i]))
    return x

#slobodni vektor
```

```
def vektor(x_i):
    x = []
    x.append(3*x_i[0]) #lijevi rubni uvjet
    for i in range(len(x_i)-1):
        x.append(3*x_i[i] + 3* x_i[i+1])
    x.append(3*x_i[len(x_i)-1]) #desni rubni uvjet
    return(x)

#kreiranje polja koje sadrži vrijednosti dijagonala
#cvorovi su ekvidistantni pa su vrijednosti
# na dijagonalama predefinirane
def dijagonala(m,n):
    x = []
    #ako je m == 1 riječ je o sporednoj dijagonali,
    # koja je duzine n-1
    if (m == 1):
        for i in range(n-1):
            x.append(1)
    #ako je m == 4 riječ je o glavnoj dijagonali duzine n
    if (m == 4):
        for i in range(n):
            x.append(4)
    return x

#poziv algoritma za rješenje problema tridijagonalne matrice
# a donja dijagonala, b glavna dijagonala,
# c gornja dijagonala, d slobodni vektor
def thomas(a,b,c,d):
    x = len(b)
    c.append(0.0)
    p =[]
    q = []
    p.append(c[0]/b[0])
    q.append(d[0]/b[0])
    for j in range(1,x):
        pj = c[j]/(b[j]-a[j-1]*p[j-1])
        qj = (d[j]-a[j-1]*q[j-1])/(b[j]-a[j-1]*p[j-1])
        p.append(pj)
        q.append(qj)
```

```
    s =[]
    s.append(q[x-1])
    for j in range(x-2,-1,-1):
        sj= q[j] - p[j]*s[0]
        s.insert(0,sj)
    return s

def cub_spl(data, n):
    new_width = n * len(data[0]) #novi broj stupaca
    new_height = n * len(data) #novi broj redaka

    #korak između nove dvije točke po retcima
    h_width = (len(data[0])-1)/(new_width - 1)
    #korak između nove dvije točke po stupcima
    h_height = (len(data)-1)/(new_height - 1)
    data_temp = []
    row_temp = []
    data_new = []
    #dijagonale koje se koriste prilikom računanja
    #Thomasovog algoritma, interpolacija po retcima
    a = dijagonala(1,len(data[0]))
    b = dijagonala(4,len(data[0]))
    c = dijagonala(1,len(data[0]))

    for i in range(len(data)):
        #lista prvih podijeljenih razlika
        x_i = podijeljene(data[i])
        vect = vektor(x_i) # elementi slobodnog vektora
        s = thomas(a,b,c,vect) # vrijednosti s_0, ... , s_n
        for j in range(new_width):
            if(j != new_width - 1):
                # index interpolacijskog čvora
                x_j = int(h_width*j)
                x = h_width*j #čvor kojeg određujemo
                #koeficijenti polinoma c_0, c_1, c_2, c_3
                c_0 = data[i][x_j]
                c_1 = s[x_j]
                c_3 = s[x_j+1] + s[x_j] - 2*x_i[x_j]
                c_2 = x_i[x_j] - s[x_j] - c_3
```

```
            #vrijednost točke x određena polinomom p_j
            w = (x-x_j)
            p_j = c_0 + c_1*w + c_2 *(w*w) + c_3*(w*w*w)
            row_temp.append(p_j)
        else:#dodavanje posljednjeg interpolacijskog čvora
            row_temp.append(data[i][len(data[0])-1])
            data_temp.append(row_temp)
    row_temp =[]

#kreiranje dodimenzionalnog polja dimenzija nove slike
for i in range(new_height):
    for j in range(new_width):
        row_temp.append(None)
    data_new.append(row_temp)
    row_temp = []

#dijagonale koje se koriste prilikom računanja
#Thomasovog algoritma, interpolacija po stupcima
a = dijagonala(1,len(data))
b = dijagonala(4,len(data))
c = dijagonala(1,len(data))

for i in range(new_width):
    #interpolacijske cvorove po stupcima spremamo u listu
    x_cvorovi =[]
    for z in range(len(data_temp)):
        x_cvorovi.append(data_temp[z][i])
    #lista prvih podijeljenih razlika
    x_i = podijeljene(x_cvorovi)
    vect = vektor(x_i) # elementi slobodnog vektora
    s = thomas(a,b,c,vect) # vrijednosti s_0, ... , s_n
    x_cvorovi =[]

    for j in range(new_height):

        if(j != new_height - 1):
            # index interpolacijskog čvora
            x_j = int(h_height*j)
            x = h_height*j #čvor kojeg određujemo
```

```
                #koeficijenti polinoma c_0, c_1, c_2, c_3
                c_0 = data_temp[x_j][i]
                c_1 = s[x_j]
                c_3 = s[x_j+1] + s[x_j] - 2*x_i[x_j]
                c_2 = x_i[x_j] - s[x_j] - c_3
                #vrijednost točke x određena polinomom p_j
                w = (x-x_j)
                p_j = c_0 + c_1*w + c_2 *(w*w) + c_3*(w*w*w)

                data_new[j][i] = domena(p_j)

            else:#dodavanje posljednjeg interpolacijskog čvora
                data_new[j][i] = domena(data_temp[len(data)-1][i])

    return(data_new)

#vrijednosti pixela slike spremamo u dvodimenzionalno polje
#slike su crno-bijele stoga je vrijednost pixela integer [0-255]
data = cv2.imread('shapes.png',0)

#pokretanje funkcije kubicne splajn interpolacije s danim faktorom,
#i kreiranje nove slike
cv2.imwrite('shapes_spline.png',np.uint8(cub_spl(data,2)))
```

3.5 Evaluacija metoda

Primjenom metoda, i proučavanjem dobivenih rezultata, osim najlošije metode najbližeg susjeda, teško je golim oko uočiti razlike, te odrediti koja metoda je dala najbolji rezultat. Stoga se koristimo standardnom metodom za procjenu učinkovitosti pojedinog algoritma, metoda PSNR, odnosno *Peak signal-to-noise ratio*. Kako bi primijenili PSNR metodu, originalnu sliku smanjili smo koeficijentom 2, na način da je vrijednost svakog piksela umanjene slike dobivena računanjem aritmetičke sredine četiriju susjednih piksela. Zatim smo sliku uvećali istim koeficijentom koristeći neku od obrađenih metoda, te odredili PSNR rezultat. Za potrebe određivanja PSNR rezultata, potrebno je odrediti MSE koeficijent, odnosno *Mean squared error*, i to na sljedeći način

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2.$$

Pritom je I izvorna slika dimenzija $m \times n$, dok je slika K jednakih dimenzija, ali je nastala smanjenjem izvorne, a zatim uvećanjem primjenom neke od opisanih metoda. Nakon toga, vrlo lako se odredi i rezultat PSNR metode, zadan formulom

$$PSNR = 20 \cdot log_{10}(MAX_I) - 10 \cdot log_{10}(MSE), \tag{3.1}$$

gdje je MAX_I maksimalna moguća vrijednost piksela. Budući da smo slike reprezentirali 8-bitnom vrijednosti, maksimalna vrijednost svakog pojedinog piksela je 255. Općenito, ovisno o reprezentaciji slike, maskimalna vrijednost piksela računa se kao $2^B - 1$, gdje je B broj bitova pojedinog uzorka, odnosno piksela.

Metoda	PSNR	Rank
Najbliži susjed	22.17785077145027	4
Linearna interpolacija	22.716538904414268	3
Kubična konvolucijska interpolacija	23.62869996193549	2
Kubična splajn interpolacija	24.05050052087546	1

Tablica 3.1: Tablica PSNR vrijednosti pojedine metode na primjeru *shapes.png*

Metoda	PSNR	Rank
Najbliži susjed	25.63587481087763	1
Linearna interpolacija	24.963655541367505	4
Kubična konvolucijska interpolacija	25.150324966291986	2
Kubična splajn interpolacija	24.984872916301367	3

Tablica 3.2: Tablica PSNR vrijednosti pojedine metode na primjeru *barbara.png*

Metoda	PSNR	Rank
Najbliži susjed	31.54453107944488	4
Linearna interpolacija	32.14700000077637	3
Kubična konvolucijska interpolacija	33.029289240752945	2
Kubična splajn interpolacija	33.55458811765598	1

Tablica 3.3: Tablica PSNR vrijednosti pojedine metode na primjeru *lena.png*

Metoda	PSNR	Rank
Najbliži susjed	31.48143200637883	3
Linearna interpolacija	29.7031778596786	4
Kubična konvolucijska interpolacija	31.037284397076505	2
Kubična splajn interpolacija	31.48143200637883	1

Tablica 3.4: Tablica PSNR vrijednosti pojedine metode na primjeru *monarch.png*

Primjenom PSNR metode za evaluaciju algoritama obrade slika, nad različitim primjerima uočavamo kako su karakteristike slika bitno utjecale na konačan rezultat. Tako je slika *shapes.png*, 3.2, u prosjeku dala najlošiji rezultat. S druge strane, slika *lena.png*, 3.4 pružila je najbolje rezultate. Zanimljiva je i činjenica kako je na primjeru *barbara.png*, 3.1, najjednostavnija metoda pružila nejbolji rezultat. Usporedbom svih rezultata, možemo zaključiti kako je metoda kubične splajn interpolacije pružila najbolje rezultate.

Ovdje se nalazi i programski kod korišten prilikom određivanja PSNR rezultata za metodu kubične konvolucijske interpolacije. Iz formule (3.1) vidljivo je kako nam je za izračun vrijednosti potrebna funkcija logaritma, stoga je upotrijebljena biblioteka *math*, koja nam pruža mogućnost korištenja iste. Analogno i za ostale metode.

```
import math
data =cv2.imread('shapes.png',0)

x = []
I = []
#kreiranje dvostruko manje slike
for i in range(0,len(data),2):
    for j in range(0,len(data[0]),2):
        x.append((int(data[i][j]) + int(data[i+1][j])
        + int(data[i][j+1]) + int(data[i+1][j+1]))/4)
    I.append(x)
    x=[]

#dvodimenzionalno polje dobiveno
#primjenom algoritma spremamo u polje K
K = kki(I,2)

z = 0
#petlja kojom računamo MSE vrijednost
for i in range(len(K)):
    for j in range(len(K[0])):
```

```
        z = z + (int(data[i][j])
        -int(K[i][j]))*(int(data[i][j])-int(K[i][j]))
nazivnik = len(K)*len(K[0])
MSE = (1/nazivnik)*z
PSNR = 20*math.log10(255) - 10*math.log10(MSE)
```

Bibliografija

[1] N. Bosner, S. Singer: *Numerička analiza, 13. predavanje*, dostupno na: https://web.math.pmf.unizg.hr/~nela/nad.html str. 55 – 65 (travanj 2019.)

[2] N. Bosner, S. Singer: *Numerička analiza, 15. predavanje, dodatak*, dostupno na: https://web.math.pmf.unizg.hr/~nela/nad.html (travanj 2019.)

[3] G. R. Dunlop: *A rapid computational method for improvements to nearest neighbour interpolation*, Camp. and Maths. with Applr., 6(3), str. 349–353 (1980)

[4] Z. Drmač i dr.: *Numerička analiza: osnovni udžbenik*, skripta PMF-Matematičkog odjela, 2003, dostupno na: https://web.math.pmf.unizg.hr/nastava/unm/materijali.php str. 290–291 (travanj 2019.)

[5] B. Guljaš: *Matematička analiza I & II*, skripta PMF-Matematičkog odjela, 2018, dostupno na: https://web.math.pmf.unizg.hr/nastava/analiza/materijali.php str. 105 (travanj 2019.)

[6] H. S. Hou, H. C. Andrews: *Cubic Splines for image Interpolation and Digital Filtering*, IEEE Trans Acoust, 26 (6), str. 508–517 (1978)

[7] R. G. Keys: *Cubic Convolution Interpolation for Digital Image Processing*, IEEE Trans Acoust, 29 (6), str. 1153–1160 (1981)

[8] I. J. Schoenberg: *Splines and histograms*, in Spline Function and Approximation Theory, vol. 21 of ISNM, Birkhaäuser 630 Verlag Basel und Stuttgart, str. 277–358 (1973)

[9] *Convolution and applications of convolution*, dostupno na: http://www.math.tamu.edu/~roquesol/Convolution_Paper.pdf (travanj 2019.)

[10] *Discrete Time Convolution*, dostupno na: http://pilot.cnxproject.org/content/collection/col10064/latest/module/m10087/latest (travanj 2019.)

[11] J. D. Hoffman: *Numerical Methods for Engineers and Scientists, Second Edition, Revised and explained, Marcel Dekker Inc., New York, 2001.*

[12] Python Engineer: Thomas algorithm*, dostupno na:* `http://pythonengineer.blogspot.com/2012/03/thomas-algorithm.html` *(lipanj 2019.)*

Sažetak

Cilj ovog rada je upoznati se s metodama za promjenu razlučivosti slike, kako bi ih mogli primijeniti na konkretnim primjerima i usporediti dobivene rezultate. Metode koje smo obradili su metoda najbližeg susjeda, linearne interpolacije, kubične konvolucijske interpolacije i kubične splajn interpolacije. Metode smo testirali na crno-bijelim slikama, standardnim za ovakvu vrstu primjene. Evaluacija dobivenih rezultata pojedinih metoda obavljena je *Peak signal-to-noise radio* formulom, na četiri različite slike. Očekivano, metoda kubične splajn interpolacije, u većini slučajeva je ponudila najbolja rješenja, dok je najjednostavnija metoda najbližeg susjeda na jednom od četiri primjera neočekivano pružila najbolji rezultat.

Za svaku od metoda priložen je i programski realiziran algoritam, nastao korištenjem programskog jezika *Python*

Summary

The main goal of this thesis is to analize methods for digital image processing, and use acquired knowledge for implementation on real examples which will be evaluated. Nearest neighbour, linear interpolation, cubic convolution interpolation and cubic spline interpolation are the methods analyzed in this thesis. All methods are implemented on grayscale images, standard for this kind of problem. Evaluation of methods is accomplished by *Peak signal-to-noise radio* formula, on four different images. Predictably, in most cases, cubic spline method provided highest PSNR score. Interestingly, tho most plane method, nearest neighbour, stepped up in one exapmle, and provided the best PSNR score.

All algorithms are developed using *Python*.

Životopis

Rođen sam 9. ožujka 1993. godine u Zadru. Pohađao sam i završio Osnovnu školu Šimuna Kožičića Benje, a potom i gimnaziju Franje Petrića u Zadru. Nakon završetka srednjoškolskog obrazovanja, 2011. godine upisujem preddiplomski studij Matematike, smjer: nastavnički, na Prirodoslovno - matematičkom fakultetu u Zagrebu. Godine 2016. stječem titulu univ. bacc. educ. math te iste godine upisujem Diplomski sveučilišni studij Matematika i informatika; smjer: nastavnički, na Prirodoslovno-matematičkom fakultetu u Zagrebu.