



Univerzitet u Novom Sadu
Prirodno - matematički fakultet,
Departman za fiziku

UPOTREBA PROGRAMSKI JEZIK PYTHON U ANALIZI METEOROLOŠKIH PODATAKA

-MASTER RAD-

Mentor: Prof.dr Ilija Arsenić

Kandidat: Martin Petraš

Novi Sad, 2018.

Sadržaj

1	Korišćeni alati	3
1.1	Python	3
1.1.1	Korišćenje biblioteka	3
1.1.2	Scipy	4
1.1.3	Numpy	4
1.1.4	Pandas	4
1.1.5	Matplotlib	5
1.1.6	MetPy	5
1.2	Django	5
1.2.1	Django arhitektura	6
2	Postupak za instaliranje	8
2.1	Instaliranje Python3 verzije	8
2.2	Podešavanje virtuelnog okruženja	9
2.3	Pokretanje Django projekta	9
3	Interpolacija	11
3.1	Linearna interpolacija	11
3.2	Natural neighbor interpolacija	11
3.3	Barnes i Cressman interpolacija	13

Zahvalnica:

mami,tati,.....

Uvod

Predmet ovog rada je izrada web aplikacije u Django razvojnom okruženju za analizu meteoroloških podataka. Web aplikacija koristi podatke sa Carpatoclim projekta, projekat koji je prikupljao meteorološke podatke u vremenskom periodu od 1961-2010 godine. Obuhvaćena oblast, kao što i samo ime projekta kaže, je oblast pružanja planinskog lanca Karpatskih planina. Aplikacija će korisniku omogućiti prikaz podataka u vizuelnom grafičkom obliku ili u vidu izlazne dadoteke sa podacima za navedeni vremenski interval.

1 Korišćeni alati

1.1 Python

Python je dinamički i objektno orijentisan programski jezik. Spada u interpreterske programske jezike visokog nivoa. Nastao je krajem devedesetih godina prošlog veka i njegov autor je Gvido van Rosum. Broj funkcija u samom jeziku je skroman, pa zahteva relativno malo uloženog vremena i napora kako bi se napravio prvi programi. Pythonova sintaksa je dizajnirana da bude čitljiva i jednostavna, što ga čini idealnim nastavnim jezikom i omogućava početnicima brzo napredovanje. Programeri provode više vremena razmišljajući o problemu koji pokušavaju da reše, a manje vremena razmišljaju o kompleksnosti jezika. Python se može izvoditi na svim važnijim operativnim sistemima (Windows, Linux/Unix, OS X), na Linux i OS X sistemima nije potrebna posebna instalacija. Za pokretanje Python programa potreban je Python interpreter, koji je pisan u programskom jeziku C. Najosnovnije korišćenje Pythona je kao jezik skriptovanja i automatizacije, koristi se za nauku o podacima i mašinsko učenje, za web usluge, metaprogramiranje. Njegova popularnost, sem čiste i pregledne sintakse, ogleda se i u velikom broju standardnih biblioteka koje su najčešće pisane u C i samom Pythonu. Što se tiče nedostataka istakli bi njegovu brzinu, međutim brzina kojom se može napisati funkcionalan program je daleko brži nego u nekom drugom jeziku.

1.1.1 Korišćenje biblioteka

Mnoge python funkcije su sadržane u specijalizovanim bibliotekama, tzv. modulima. Učitavanje modula se postiže naredbom *import*. Ova mogućnost je pythonu donela popularnost jer postoje mnogi moduli koji jako pomažu prilikom rada. Python je probo svoju popularnost stekao pri izradi veb programa međutim usavršavanjem podrške za dodatne module otvorilo je vrata pythonu i u drugim oblastima. Moduli poput *numpy* i *pandas*, koji se koriste za analizu i vizuelno prikazivanje podataka, su python svrstali uz rame sa ostalim kako komercijalnim programima tako i sa programima otvorenog koda kao što su R, MATLAB, SAS, Stata i ostali. Kada pogledamo da python spada u besplatni programski jezik, njegova popularnost nije slučajnost. Jedan deo ove popularnosti u obradi podataka pripada i laka integracija sa C, C++ i FORTRAN kodom. Pošto su moduli jako značajni za python istaći ćemo neke od naznačajnijih. Samo da napomene, sve nabrojane mogućnosti koje pružaju biblioteke je moguće izvesti koristeći isključivo samo Pythonov kod. Prednost biblioteka je u efikasnosti i lakoći kojim se postižu isti rezultati, uz manje utrošenog vremena. Jedna od značajnih biblioteka otvorenog koda je i SciKit. Ona sadrži različite klase objekata za klasifikaciju, algoritme mašinskog učenja (engl. *Machine Learning*). U SciKit paketu se nalaze NumPy i

SciPy biblioteke koje sadrže klase za razne standardne matematičke i numeričke funkcije.

1.1.2 Scipy

Scipy (engl. *Scientific Python*) proširuje funkcionalnost Numpy-a sa značajnom zbirkom algoritama za Fourijeovu transformaciju, regresiju i druge matematičke tehnike. Sadrži neke od sledećih modula :

- `scipy.integrate` - integracija funkcija
- `scipy.special` - specijalne funkcije
- `scipy.optimize` - optimizacija
- `scipy.interpolate` - interpolacija

Nama od značaja će biti modul za interpolaciju.

1.1.3 Numpy

NumPy (*Numerical Python*) predstavlja fundamentalnu open source Python biblioteku kada su obzir uzmu numerički proračuni. Sadrži matematičke funkcije zadužene za operaciju na podacima, koje se veoma brzo i efikasno izvršavaju. Poziva se komandom :

```
import numpy as np
```

Za numeričke proračune, nizovi kod NumPy su dosta efikasniji prilikom njihove manipulacije nego bilo koja druga izgrađena struktura unutar Pythona. Biblioteka je pisana u C-u i Fortranu.

1.1.4 Pandas

Pandas je dizajniran kako bi se ubrzao rad sa strukturiranim ili tabelarnim podacima i kao takav, postao jako moćna i produktivna alatka za analizu podataka. Ono što krasi ovu biblioteku je i velika podrška od strane zajednice, njen aktivna razvoj, odličan rad sa ostalim bibliotekama, izgrađena je na osnovama *numpy* pa je takođe krasi brzina. Neke od mogućnosti koje pruža ova biblioteka su:

- ulaz-izlaz podataka u različitim formatima (csv, txt, SQL...)
- indeksiranje, sortiranje, rangiranje
- čišćenje podataka

- grupisanje
- vizuelizacija

Sačinjenja je od dve strukture podataka, *dataframe* i *series*. *Series* predstavljaju jednodimenzioni objekat sačinjen od tabele sa vrednostima i njihovim indeksima. Skraćenica za *pandas* je *pd*, poziva se korišćenjem komande:

```
import pandas as pd
```

1.1.5 Matplotlib

Najviše korišćena i najstarija biblioteka koja se koristi za vizuelno predstavljanje podataka. U kombinaciji sa *numpy* i *pandas* čine jako moćne alate, koji se mogu nositi sa komercijalnim i dosta skupim alatima kao što su *Matlab* i *Mathematica*. Mana ove biblioteke je potreba za pisanjem većeg broja linija koda kako bi se dobila naprednija vizuelizacija podataka. Potreba za povećanim kvalitetom vizuelizacije, nastale su savremenije biblioteke koje se u većoj ili manjoj meri oslanjaju na *matplotlib*. Jedna od takvih biblioteka je *seaborn* čiji je fokus na atraktivnijoj izradi statističkih grafika. *Seaborn* se potpuno oslanja na *matplotlib*. Naveo bih još i *bokeh* i *altair* koje nisu zavisne od *matplotlib*, a takođe se koriste za vizuelizaciju podataka.

1.1.6 MetPy

Metpy predstavlja kolekciju alata u Pythonu za pregled, vizuilizaciju i proračuna meteoroloških podataka. Paket je nastao od strane *Unidata* udruženja, koja se bavi izradom alata za naučne svrhe. MetPy paket sadrži dosta opcija, koristi se za vizelizaciju radarskih slika, rad sa netCDF podacima, vizuelizaciju radio-sondažnih merenja, vizuelizaciju meteoroloških podataka i ostalo. Mi ćemo koristiti ovu biblioteku u svrhu interpolacije polja temperature. Za instaliranje MetPy u terminal kucamo:

```
conda install -c -conda-forge metpy
```

Instaliranjem ovog paketa dobijamo razne alate, nama će najviše značiti mogućnost interpolacije za polje temperature kao i negove vizuelizacije. Za vizuilizaciju se koristi *Cartopy*, Pythonov paket za crtanje mapa.

1.2 Django

Nagla ekspanzija interneta praćenja je i povećanjem popularnosti web aplikacija. Google, Facebook, YouTube su samo od nekih popularnih web aplikacija koje

čine svakodnevnicu pretraživanja. Web aplikacija je u stvari program, kojim pristupamo pomoću internet pretraživača(browser). Ovakav pristup omogućava veću bezbednost, sigurnost bez potrebu za instaliranjem na operativni sistem. Sve što vam je potrebno kako bi pristupili programu je uređaj koji ima pristup internetu. Usled spomenutog naglog rasta popularnosti web aplikacija, javila je potreba za brzim i automatizovanim načinom njihove izrade. Tako je nastao Django, web framework napisan u Python-u i služi za izradu web stranica i aplikacija.

Razvijan je od 2003. godine od strane novinara, koji su zahtevali brz razvoj web stranica. Kao slobodni softver(eng. *open source*) objavljen je 2005. godine, kada je i dobio ime Django po jazz gitaristi Djangu Reinhardt. Od 2008. godine za razvoj Django frameworka zadužena je *Django Software Foundation*. Django je zapravo skup alata i biblioteka koje programerima web aplikacija olakšavaju izradu i brigu o povezivanju svih njenih delova. Dizajniran je kako bi se skratilo vreme potrebno za izradu web aplikacije, držeći se strogih zahteva iskusnih web programera koji su ga stvorili. Web programiranje zahteva ponavljanje isti zadataka kao što su :

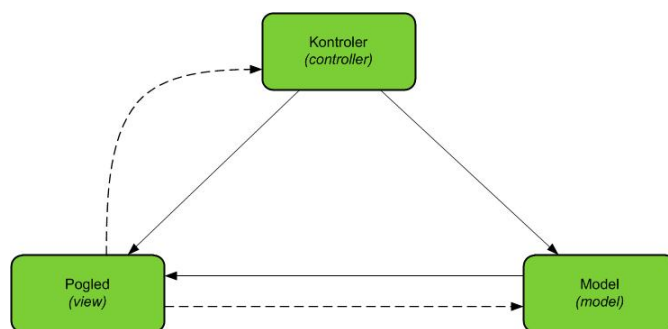
- korisnička prava
- registracija korisnika
- konfigurisanje URL-ova
- validacija unesenih podataka
- administracija sajta

Pošto se Django bazira na Python-u pisanje ovakvog koda sadrži nekoliko linija. U drugim programskim jezicima takvi zadaci predstavljaju mukotrpan proces zahtevajući dosta vremena.

1.2.1 Django arhitektura

Kao što je rečeno, Django služi za brzu i jednostavnu izradu web stranica i aplikacija. Ovu osobinu mu omogućava arhitektura MVC(*model-view-controller*) na kojoj se bazira, i koja razdvaja logiku same aplikacije od prikazanog dela aplikacije. Možemo je podeliti na tri dela:

- Model
- View (Pogled)
- Controller(Kontroler)



Slika 1. Koncept modela MVC

Pune linije predstavljaju vezu izmedju navedenih elemenata za razmenu podataka, dok isprekidane linije predstavljaju vezu kod koje neki drugi objekt vrši tu funkciju. Django koristi drugačije nazive elemenata MVC-a. Tako *view* predstavlja kontroler, a *template* (šablon) predstavlja *view* iz MVC-a pa implementaciju MVC-a možem za Django zvati kao MTV (*model-template-view*). Ova promena je objašnjena na takav način što kod Djanga pogled predstavlja podatke koji se prezentuju dok šablon način na koji se podaci prikazuju.

Model je zadužen za komunikaciju sa bazom podataka. Model je u Django okruženju klasa (engl. *class*) i određuje varijable i metode pridružene određenim tipovima podata. Pridružene varijable predstavljaju kolone u tablici dok metode definišu relacije između varijabli. Ulog modela je da dohvati tražene podatke i prosledi ih ka pogledu. Model nema informacija o šablonima i funkcijama izvedenih u pogledima.

Uloga pogleda je prikazivanje podataka dohvaćenih iz baze podataka u internet pretraživaču. Priliko izrade web stranice, svaka kreirana aplikacija ima svoj pogled. Pogled u stvari sačinjavaju funkcije u Python programskom jeziku. Ove funkcije imaju ulogu prenosnika podataka koji će biti prikazani.

Pregled je HTML stranica s dodatnim strukturama koje omogućavaju prikaz podataka koji su poslani od pogleda. Uloga pregleda je da sadržaj poslat od strane pogleda ugradi u HTML kod, koji će se prikazati u internet pretraživaču.

2 Postupak za instaliranje

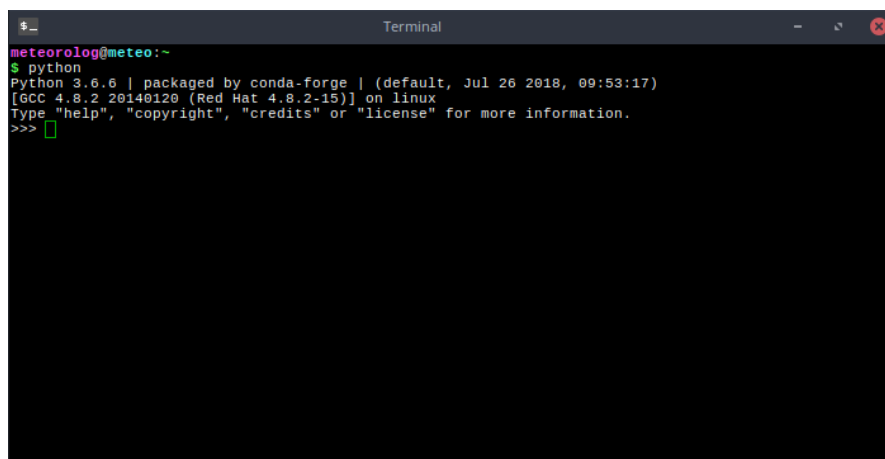
2.1 Instaliranje Python3 verzije

Django je pisan u Python-u i kako bi ga koristili potrebno je imati instaliran Python. Python kao programski jezik postoji za sve tri popularne systemske platforme Windows, Linux i OSX. Ovde će biti prikazan postupak instaliranja potrebnih paketa za rad pod Linux operativnim sistemom. Gotov sve distribucije linuxa dolaze sa instaliranim Python-om, razlika može biti u verziji. Postoje dve verzije, starija 2.7 verzija koja je podržana do kraja 2020. godine, i novija verzija 3.6. U radu je korišćena verzija 3.5. Proveru verzije radimo tako što otvorimo terminal i kucamo sledeću komandu :

```
python3 --version
```

Ukoliko je Python verzije 3.6 instaliran, pojavi se sledeće

```
Python 3.6.6
```



Slika 2. Python verzije 3.6 pokrenut iz terminala

Ukoliko nemate instaliran Python verzije 3.6, možete ga instalirati :

```
sudo apt install python3.6  
sudo dnf install python3  
sudo zypper install python3
```

gde je prva komanda za Debian, Fedoru i openSuse redom.

2.2 Podešavanje virtuelnog okruženja

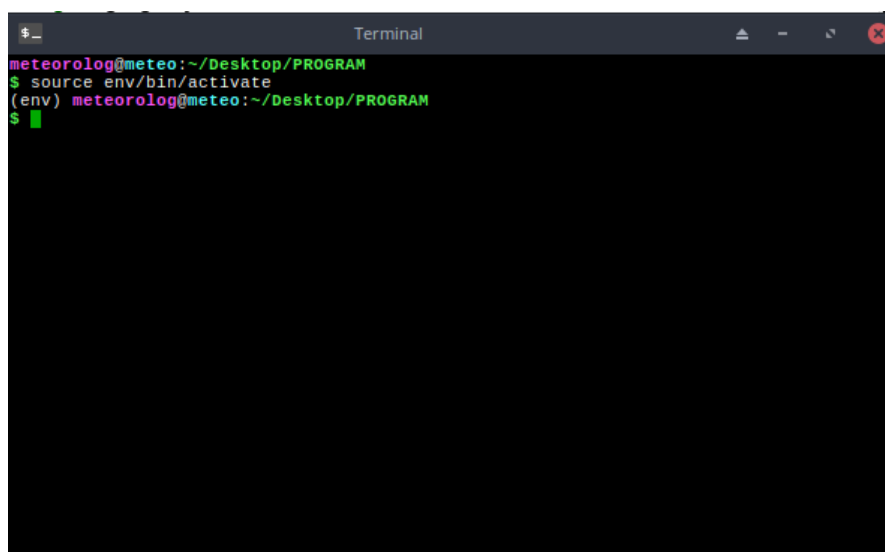
Pre nego što instaliramo Django pokazaćemo kako instalirati veoma zahvalnu alatku, koja nam omogućava održavanje čistoće paketa na sistemu. Virtuelno okruženje (engl. *virtual environment*) izoluje projekte jedan od drugog, držeći pakete vezane za svaki projekat zasebno. Kako bi kreirali virtuelno okruženje zvano *env*, u direktorijum u kome smo započeli projekat, u terminal kucamo :

```
python3 -m virtualenv env
```

Kada ovo odradimo, u dokumentu se pojavi dokument *env* u koji će se instalirati svi paketi koje u buduću budemo koristili. Kada ovo sve odradimo, ostalo nam je još da ovo virtuelno okruženje aktiviramo. Aktiviranje virtuelno okruženja radimo na sledeći način :

```
source env/bin/activate
```

nako čega se u terminalu pojavljuje nastavak (*env*). Ovaj nastavak nam potvrđuje da je virtuelno okruženje aktivirano. Kako bi zatvorili virtuelno okruženje, u terminal kucamo komandu *deactivate*.



Slika 3. Postupak aktiviranje virtuelnog okruženja

2.3 Pokretanje Django projekta

Kada smo podesili virtuelno okruženje možemo instalirati Django. Instaliranje radimo pomoću *pip* menadžera-. Prvo uradimo nadogradnju *pip* paketa, kako bi imali poslednju verziju. To radimo na sledeći način :

```
python3 -m pip install --upgrade pip
```

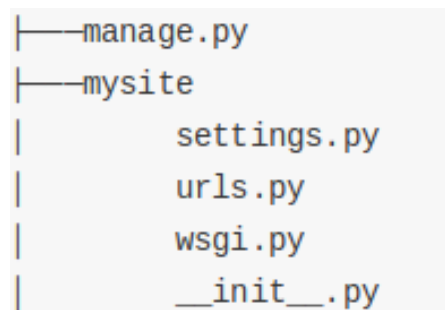
Kada smo uradili nadogradnju pip-a, Django instaliramo :

```
pip install django
```

Kada ovo odradimo možemo uspešno pokrenuti naš prvi Django projekat. Izrada projekta podrazumeva pokretanje Django skripte koja će izgraditi kostur potrebnih datoteka. Kucamo sledeću komandu :

```
django-admin startproject mysite .
```

i u stvorenom direktorijum dobijamo strukturu kao sa slike 4. U kreiranoj datoteci *manage.py* skripta predstavlja upravitelja-menadžera projekta pomoću kojeg se pokreće web server. Skripta *settings.py* služi za konfigurisanje našeg sajta. *urls.py* sadrži šablone(preglede) koje koristimo. Kako bi pokrenuli web server u terminal



```
├──manage.py
├──mysite
│   ├──settings.py
│   ├──urls.py
│   ├──wsgi.py
│   └──__init__.py
```

Slika 4. Izgled kostura-struktura django projekta

kucamo :

```
python manage.py runserver
```

nako čega u internet pretraživač upišemo ovu adresu *http : //127.0.0.1 : 8000/*. Ako ste sve uradili kako treba u internet pretraživaču bi trebalo da se pojavi čestitka o uspešnom pokretanju web sajta.

3 Interpolacija

Podaci sa meteoroloških stanica su veoma značajni za prognozu vremena. U prognozi vremena ovi podaci se koriste kao početni uslovi za jednačine dok se kod analize koriste za vizuelni prezentaciju polja promenljivih. Cilj je što vernije predstaviti stanje varijabli na određenom nivou i mestu. Ako bi postojala merenja u svim tačkama prostora i uz to dovoljno tačna, problem tačnosti ne bi postojao. Međutim, broj raspoloživih stanica koje šalju podatke i dalje nije dovoljan jer na nekim mestima, poput okeana ili neprisupačnim predela, ne postoji mogućnost postavljanja mernih uređaja. Jedan način rešenje ovog problema su satelitska merenja a drugi pristupačniji je uz pomoć interpolacije.

Interpolacija predstavlja metod procene vrednosti varijable na mestima gde oni nisu određeni, koristeći poznate vrednosti iz lokalnih lokacija. Interpolacija je moćna metoda za analizu podataka kod kojih su merenja rasuta. Problem interpolacije je njena implementacija. Za nekoliko poznatih primera, koristeći različite metode interpolacije dobićemo različite rezultate. Zbog toga, aplikacija će omogućivati prikaz četiri metoda interpolacije :

- Linearna interpolacija
- Interpolacija najbližeg suseda
- Barnes interpolacija
- Cressman interpolacija

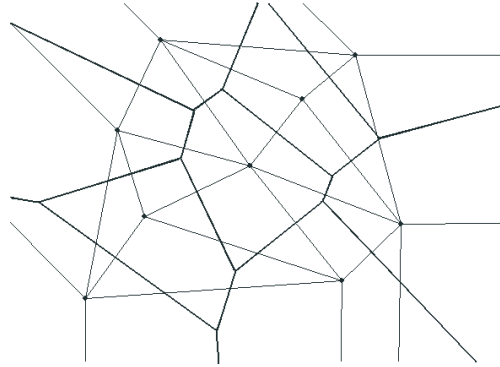
Metode interpolacije možemo podeliti u dve kategorije u zavisnosti koje tačke su uzete za definisanje interpolacije u određenoj tački. Ove dve kategorije su globalna i lokalna interpolacija.

3.1 Linearna interpolacija

3.2 Natural neighbor interpolacija

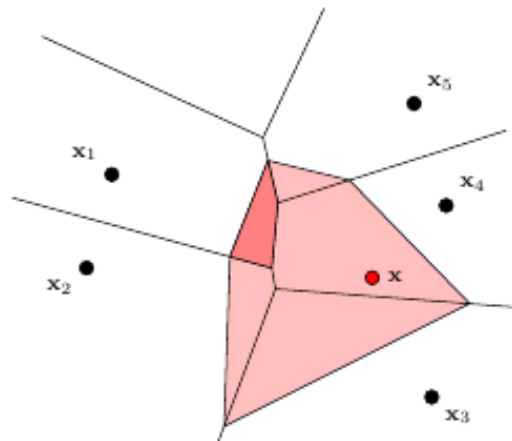
Postoje razne implementacije natural neighbor interpolacija. Ovde će biti obrađena lokalna metoda koja se bazira na Voronojevom dijagramu i Deloneovoj trijangulaciji. Pokazala se kao vremenski malo zahtevna metoda, jer se bazira na ideji ponovnog korišćenja rezultata tokom proračuna u što većem obimu. Prvo ćemo definisati teoriju Voronoj dijagrama i Delonove trijangulacije. Za dati skup tačaka u ravni, koje ćemo u daljem tekstu nazvati centrima, konstrukcija Voronoj dijagrama podrazumeva podelu ravni na ćelije, tako da svaka ćelija sadrži jedan centar. Deloneov graf nad datim skupom tačaka u ravni je dualan graf Voronoj dijagramu. Drugim rečima, čvorovi ovog grafa su centri, a grane povezuju čvorove,

koji odgovaraju centrima susednih Voronoi ćelija. Na slici 5 su prikazani Voronoi dijagram (podebljane linije) i Deloneova triangulacija (obične linije) skupa od devet centara. Ako dve Voronoi ćelije dele zajedničke ivice onda za njih kažemo



Slika 5. Voronoi dijagram i Deloneova triangulacija

da su prirodni susedi (natural neighbor). Kako bi se definisali susedi za određene tačke interpolacije x , prvo formiramo Voronoi ćeliju u kojoj se ova tačka nalazi. Formiranjem ćelije definišemo zapreminu $V(x)$, koja će imati nove susede. Na slici 6 možemo videti da su za tačku x prirodni susedi x_1, x_2, x_3, x_4, x_5 , odnosno da Voronoi ćelija $V(x)$ deli ivice sa $V(x_1), V(x_2), V(x_3), V(x_4), V(x_5)$. Vrednost



Slika 6. Najbliži susedi za tačku x

interpolacije u tački x dobijamo tako što usrednjimo vrednosti ovih definisanih okolnih, susednih tačaka.

3.3 Barnes i Cressman interpolacija

Barnes i Cressman spadaju u grupu sukcesivno korekcionih metoda, kod kojih se polje varijabli modifikuje koristeći osmotrene podatke iterativnim postupkom. Metod se zasniva na prolazu kroz svaku tačku mreže, ažuriranj varijabli za svaku od njih na osnovu osmotrenih podataka koje obuhvataju tu tačku mreže. Nakon prvog prolaza kroz zadato područje, radi se drugi prolaz, modifikujući polje svake tačke mreže u odnosu na okolne osmotrene tačke. Ove interacije se rade sve dok razlike između dve iteracije ne budu manje od zadatog praga. Polje mreže u tački i se ažurira na sledeći način:

$$f_i^{m+1} = f_i^m + \frac{\sum_{k=1}^K w_{ik}^m (O_k - f_k^m)}{\sum_{k=1}^K w_{ik}^m} \quad (1)$$

gde f_i^m je vrednost varijable u i -toj tački mreže za m iteraciju. O_k je vrednost k -te osmotrene vrednosti u okolini tačke mreže, a w_{ik}^m je težinska funkcija i određuje se na sledeći način:

$$w_{ik}^m = \exp\left(\frac{-r_{ik}^2}{2R_m^2}\right) \quad (2)$$

Za Cressman šemu težinska funkcija zavisi od radijusa delovanja R_m kao:

$$w_{ik}^m = \frac{R_m^2 - r_{ik}^2}{R_m^2 + r_{ik}^2} \quad \text{za} \quad r_{ik}^2 \leq R_m^2 \quad (3)$$

$$w_{ik}^m = 0 \quad \text{za} \quad r_{ik}^2 > R_m^2 \quad (4)$$

gde je r_{ik}^2 dužina između osmotrenih vrednosti i tačke mreže. Kod ove metode, osmotrene vrednosti izvan radijusa dešejstva se ne uzimaju u ažuriranje vrednosti polja.