UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA RAČUNARSTVO I INFORMATIKU

Algoritmi za višedimenzionalnu interpolaciju

SEMINARSKI RAD
- PRVI CIKLUS STUDIJA -

Student:
Amer Mujalo

Sažetak

U ovom seminarskom radu istražujemo različite algoritme za višedimenzionalnu interpolaciju, fokusirajući se na analizu i implementaciju u programskom jeziku Julia. Interpolacija je ključna metoda u numeričkom računanju koja se koristi za procjena vrijednosti funkcija između poznatih tačaka. Ovaj rad obuhvata analizu i implementaciju nekoliko različitih algoritama, uključujući Krigingovu, Shepardova kao i radijalnu baznu funkciju (RBF) interpolaciju. Svaki algoritam je implementiran u Julia jeziku, uz prikaz testnih slučajeva i rezultata. Rad se završava analizom primene svakog algoritma u praksi i diskusijom o njihovim performansama.

Abstract

This seminar paper explores various algorithms for multidimensional interpolation, focusing on their analysis and implementation in the Julia programming language. Interpolation is a crucial method in numerical computing used for estimating function values between known points. This paper covers the analysis and implementation of different algorithms, including Radial Basis Function (RBF), Kriging, and Shepard interpolation. Each algorithm is implemented in Julia, with test cases and results presented. The paper concludes with an analysis of each algorithm's practical applications and a discussion of their performance.

Uvod

Interpolacija je tehnika koja omogućava procenu vrijednosti funkcija između poznatih tačaka u višedimenzionalnom prostoru. Ova metoda je ključna u različitim oblastima kao što su statistika, inženjering i nauke o podacima. U ovom radu analiziramo i implementiramo nekoliko različitih algoritama za višedimenzionalnu interpolaciju koristeći Julia programski jezik. Svaki algoritam pruža različite prednosti i primjene, koje ćemo detaljno istražiti.

Implementacija algoritma

Shepardova interpolacija

Uvod

Shepardova interpolacija, poznata i kao **Interpolacija obrnuto ponderiranim udaljenostima** (eng. Inverse Distance Weighting, IDW), jednostavna je tehnika višedimenzionalne interpolacije koja se temelji na ideji da tačke bliže mjestu interpolacije imaju veći utjecaj na procijenjenu vrijednost. Ovaj pristup koristi prostornu konfiguraciju poznatih podataka kako bi procijenio vrijednosti u nepoznatim tačkama. Shepardova interpolacija često se koristi u prostornim analizama, poput geostatistike i analize terena, gdje je prostorna autokorelacija između tačaka.

Osnovne formule

Osnovna formula Shepardove interpolacije temelji se na **ponderiranoj sredini** poznatih vrijednosti, gdje su težine obrnuto proporcionalne udaljenosti između interpolacijske tačke i poznatih tačaka. Neka su (x_i,y_i) poznate tačke s pripadajućim vrijednostima $f(x_i,y_i)$, a (x,y) je tačka gdje interpoliramo vrijednost. Shepardova interpolacija definira procijenjenu vrijednost f(x,y) kao:

$$f(x,y) = \frac{\sum_{i=1}^{N} \frac{f(x_i,y_i)}{d(x,y,x_i,y_i)^p}}{\sum_{i=1}^{N} \frac{1}{d(x,y,x_i,y_i)^p}}$$

gdje je:

- ***N*** broj poznatih tačaka,
- **$d(x,y,x_i,y_i)$** Euklidska udaljenost između tačke (x,y) i poznate tačke (x_i,y_i),
- **p** eksponent koji kontrolira koliko brzo težina opada s udaljenošću (najčešće je **p=2**).

U ovoj formuli, što je tačka dalje od tačke interpolacije, to je njezin doprinos manji. Ako je udaljenost vrlo mala ili nula, procjena postaje vrlo bliska vrijednosti u poznatoj tački.

Primjena

Shepardova interpolacija našla je široku primjenu u poljima kao što su:

- **Geostatistika**: interpolacija podataka o tlu, analize površina i topografije.
- **Meteorologija**: prostorna interpolacija meteoroloških podataka kao što su temperatura i tlak.
- **GIS (Geografski Informacijski Sistemi)**: modeliranje i interpolacija prostorno distribuiranih podataka.
- **Računarska grafika**: generiranje i interpolacija tekstura na mrežama.

IDW je popularan zbog jednostavne implementacije i intuitivnog koncepta, ali njegova primjena može biti ograničena u situacijama gdje prostorna autokorelacija nije tako jednostavna ili gdje su potrebne sofisticiranije metode, poput **Kriging**-a.

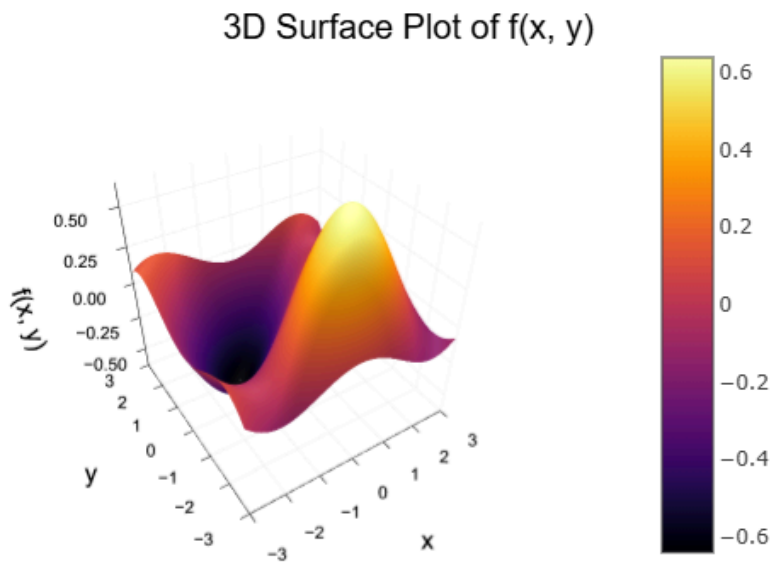
Pseudokod Shepardove interpolacije

U nastavku je primjer implementacije Shepardove interpolacije:

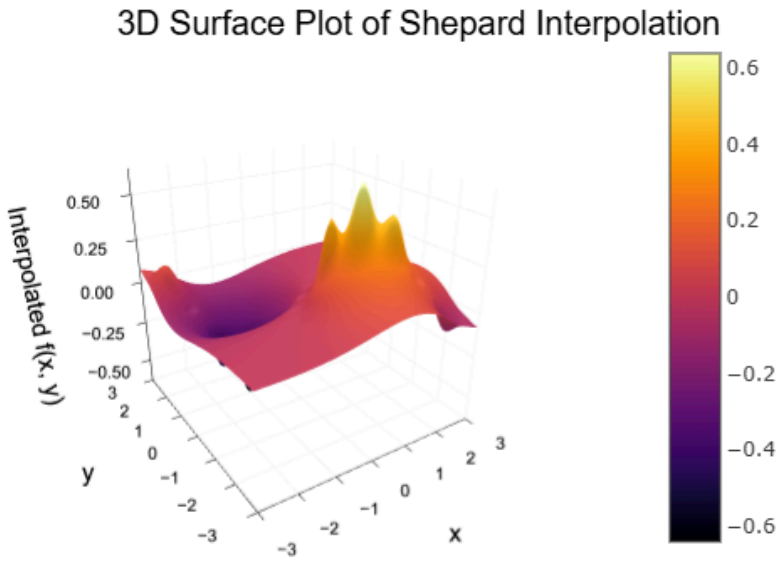
```
# Shepard's interpolation function
function shepard_interpolation(x, y, x_known, y_known, f_known, p=2)
    N = length(f_known)
    numerator = 0.0
    denominator = 0.0
    for i in 1:N
        dist = euclidean_distance(x, y, x_known[i], y_known[i])
        if dist == 0.0
            return f_known[i]  # Vrati poznatu vrijednost
        end
        weight = 1.0 / dist^p
        numerator += weight * f_known[i]
        denominator += weight
    end
    return numerator / denominator
end
```

Za f(x,y) = `sin((x - y) / 2) * cos(sqrt(x^2 + y^2) / 2)`

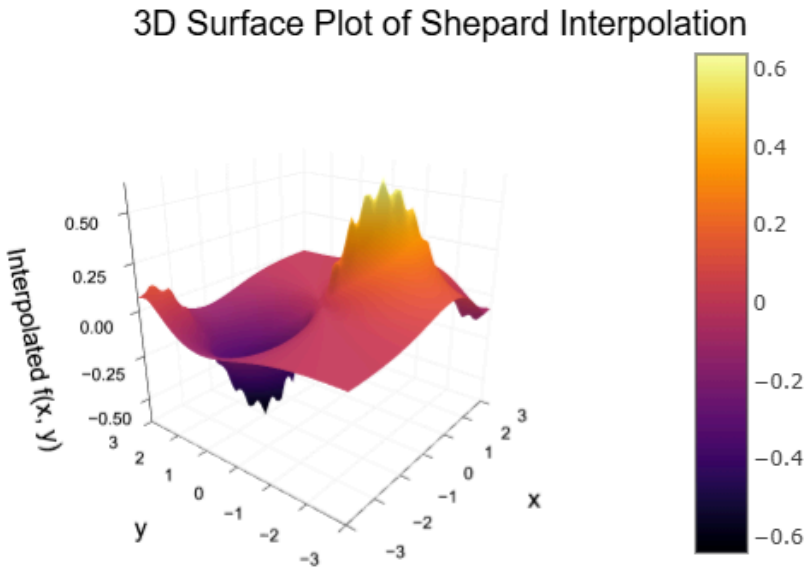
Originalna funkcija u rangu od -3 do 3 dobijamo:



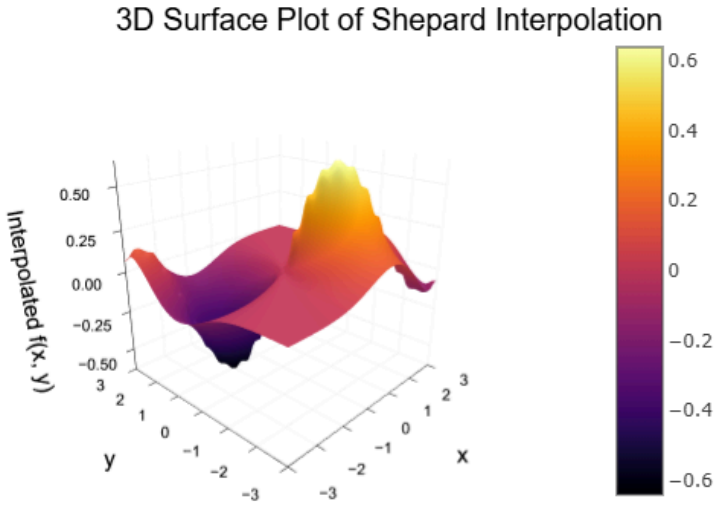
A kada pokušamo interpolirati za p=2, uzimajući 26 tačaka, dobijamo:



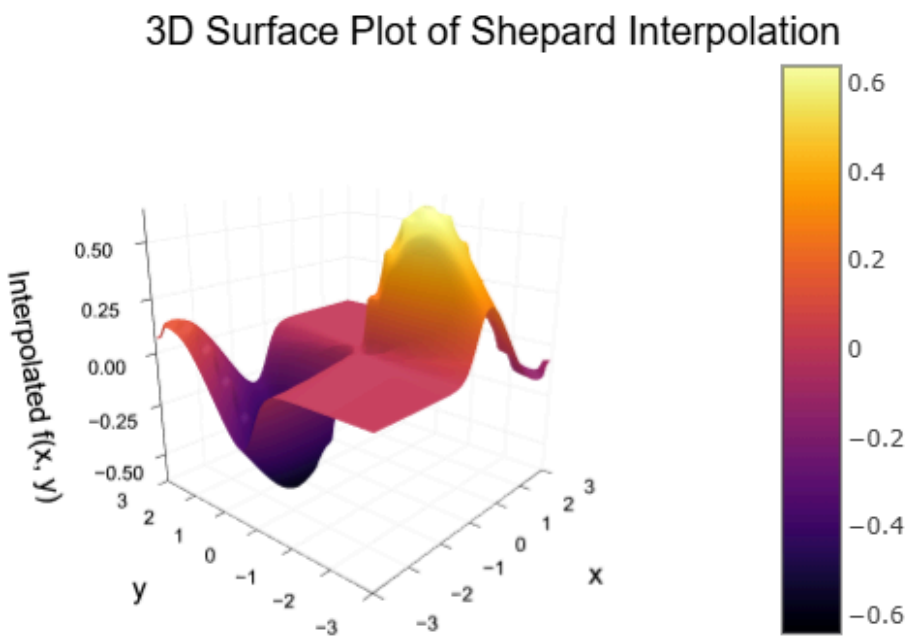
A kada pokušamo interpolirati za p=2, uzimajući 52 tačke, dobijamo:



A kada pokušamo povećati za p=5, uzimajući istih broj tačaka, dobijamo:



i za p=10 dobijamo:




```
for p in [1, 2, 3, 4, 5, 10]
    Z_interpolated = [shepard_interpolation(xi, yi, x_known, y_known, f_known, p) for xi in x_grid, yi in y_grid]
    Z_actual = [f(xi, yi) for xi in x_grid, yi in y_grid]
    relative_error = abs.(Z_interpolated .- Z_actual) ./ (abs.(Z_actual) .+ 1e-10)
    error_results[p] = relative_error

end

for (p, error_matrix) in error_results
    println("Srednja relativna greska za p = $p:")
    println(mean(error_matrix))
end
```

```
Srednja relativna greska za p = 10:
0.9468715056582867
Srednja relativna greska za p = 5:
1.0076664588917437
Srednja relativna greska za p = 4:
1.0498902886821724
Srednja relativna greska za p = 3:
1.114165259866315
Srednja relativna greska za p = 2:
1.182305987733621
Srednja relativna greska za p = 1:
1.155529425375526

Srednja apsolutna Greska za p = 10:
0.09887524145053835
Srednja apsolutna Greska za p = 5:
0.09524388084113898
Srednja apsolutna Greska za p = 4:
0.10033443876512224
Srednja apsolutna Greska za p = 3:
0.11279182636663714
Srednja apsolutna Greska za p = 2:
0.14026012554823875
Srednja apsolutna Greska za p = 1:
0.19078407418575052

Za tačku (-0.45, 0.45):
Interpolirana vrijednost: -0.44047373727323175
Stvarna vrijednost: -0.41313057286996635
Razlika: 0.027343164403265396

Za tačku (-0.25, 0.25):
Interpolirana vrijednost: -0.19049796594752566
Stvarna vrijednost: -0.24354832880354732
Razlika: 0.05305036285602166

Za tačku (-0.1, 0.1):
Interpolirana vrijednost: -0.016667504915751735
Stvarna vrijednost: -0.09958393708102278
Razlika: 0.08291643216527105
```

Greške u interpolaciji za tačke **(−0.45,0.45)**, **(−0.25,0.25)** i **(−0.1,0.1)** mogu se objasniti sljedećim faktorima:

1. **Blizina Poznatih Tačaka:** Tačke za koje se vrši interpolacija nalaze se između poznatih tačaka **(−0.5, 0.5)** i **(0.0, 0.0)**. Ako su interpolacijske tačke daleko od poznatih tačaka, greška se povećava. Na primjer, tačka **(−0.1,0.1)** je najbliža tački **(0.0,0.0)**, ali razlika je veća jer je dalje od ostatka poznatih tačaka.

greške su veće zbog udaljenosti tačaka za interpolaciju od poznatih tačaka i same distribucije tih tačaka na mreži.

Analiza Grešaka Shepardove Interpolacije

Visoke greške u Shepardovoj interpolaciji mogu se objasniti sljedećim razlozima:

1. **Raspodjela Poznatih Tačaka:** Ako su poznate tačke rijetke ili loše raspoređene, interpolacija može biti netačna.
2. **Parametar p:** Različite vrijednosti ppp mogu značajno utjecati na rezultate. Loš izbor može povećati greške.
3. **Numerička Stabilnost:** Velike razlike u udaljenostima mogu uzrokovati probleme sa preciznošću.
4. **Složenost Funkcije:** Funkcija koju interpoliramo može biti previše složena za određene parametre ili raspodjelu tačaka.

Prednosti i Mane Shepardove Interpolacije

Prednosti:

1. **Jednostavnost:** Algoritam je jednostavan za implementaciju i razumijevanje.
2. **Fleksibilnost:** Može se prilagoditi različitim problemima putem parametra p, koji kontrolira uticaj udaljenosti na interpolaciju.
3. **Lokalni Uticaj:** Svaka tačka u mreži utiče samo na lokalni dio prostora, što može biti korisno za nepravilne raspodjele podataka.

Mane:

1. **Osetljivost na Distribuciju Tačaka:** Ako su poznate tačke nepravilno raspoređene, interpolacija može biti netačna.
2. **Numerička Nestabilnost:** Velike ili male udaljenosti mogu uzrokovati numeričke probleme i preciznost može biti pogođena.
3. **Visoki Računarski Troškovi:** Može biti spor za velike skupove podataka zbog velikog broja proračuna težina.

Kriging (Geostatistička interpolacija)

Uvod

Kriging je moćna metoda geostatističke interpolacije koja se koristi za predikciju nepoznatih vrijednosti na temelju poznatih podataka u prostoru. Metoda je dobila ime po pioniru geostatistike, Danie G. Krigeu, i razvijena je kako bi se primijenila u rudarskoj industriji za procjenu sadržaja ruda između sondi. Kriging se danas koristi u raznim disciplinama, uključujući geoznanosti, poljoprivredu, hidrologiju, i ekologiju.

Osnovna formula Kriginga

Kriging se temelji na pretpostavci da prostorna varijabilnost može biti modelirana kao kombinacija deterministčkog trenda i prostorno-korelirane slučajne komponente. Osnovna formula za predikciju vrijednosti $Z(\mathbf{x_0})$ u tački $\mathbf{x_0}$ je:

$$Z(x_0) = \sum_{i=1}^{n} \lambda_i Z(x_i)$$

Gdje su:

- $Z(\mathbf{x_0})$**:** procijenjena vrijednost u tački $\mathbf{x_0}$
- $Z(\mathbf{x_i})$ poznata vrijednost u tački $\mathbf{x_i}$
- $\boldsymbol{\lambda_i}$**:** težina pridružena tački $\mathbf{x_i}$
- **n:** broj poznatih tačaka

Težine $\boldsymbol{\lambda_i}$ određene su na osnovu prostorne kovarijance (ili varijanse) između tačaka, s ciljem minimizacije varijance procjene.

$$\mathbf{C}\lambda = \mathbf{c}_0$$

Gdje su:

- $\mathbf{C}$ $n \times n$ kovarijacijska matrica između svih poznatih tačaka x_i i x_j, tj. $C_{ij} = \text{cov}(Z(x_i), Z(x_j))$.
- λ vektor težina $\lambda_1, \lambda_2, \ldots, \lambda_n$.
- $\mathbf{c}_0$ vektor kovarijacije između svake poznate tačke x_i i tačke predikcije x_0, tj. $c_{0i} = \text{cov}(Z(x_0), Z(x_i))$.

U eksplicitnom obliku, težine λ_i se dobijaju rješavanjem:

$$\lambda_i = \sum_{j=1}^{n} \left(C^{-1}\right)_{ij} c_{0j}$$

Gdje je C^{-1} inverzna kovarijacijska matrica.

Primjena Kriginga

Kriging se koristi u širokom spektru primjena, uključujući:

- **Geoznanosti**: Za procjenu koncentracija minerala, nafte, ili podzemnih voda u neistraženim područjima.
- **Ekologija**: Za mapiranje distribucije biljnih ili životinjskih vrsta.
- **Hidrologija**: Za interpolaciju podataka o padavinama ili vodenim tokovima.
- **Poljoprivreda**: Za precizno poljodjelstvo, kao što je procjena plodnosti tla ili distribucija hranjivih tvari.

Pseudokod Kriginga

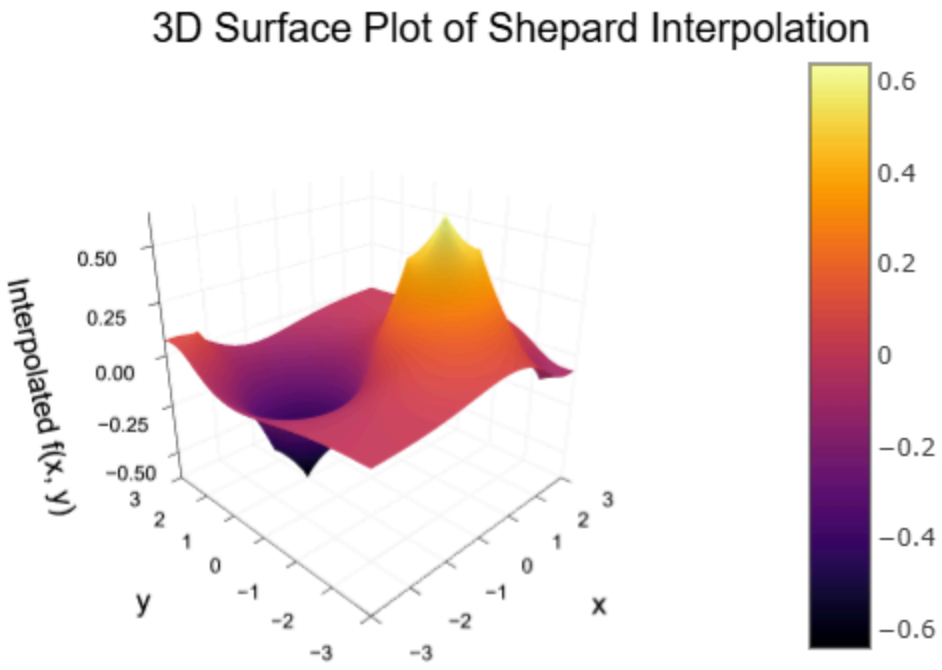
Pseudokod ili implementacija na osnovni Kriginga može izgledati ovako:

```
# Izračunava kovarijansu između dva para tačaka
function covariance(x1, y1, x2, y2, range=1.0)
    d = sqrt((x1 - x2)^2 + (y1 - y2)^2)
    return exp(-d / range)  # Exponential covariance function (Gaussian)
end

# Kreira matricu kovarijanse za skup tačaka
function covariance_matrix(x, y, range=1.0)
    n = length(x)
    C = zeros(n, n)
    for i in 1:n
        for j in 1:n
            C[i, j] = covariance(x[i], y[i], x[j], y[j], range)
        end
    end
    return C
end
```

```
# Kreira vektor kovarijanse između tačaka i novih tačaka
function covariance_vector(x, y, x_new, y_new, range=1.0)
    n = length(x)
    C_new = zeros(n)
    for i in 1:n
        C_new[i] = covariance(x[i], y[i], x_new, y_new, range)
    end
    return C_new
end
```

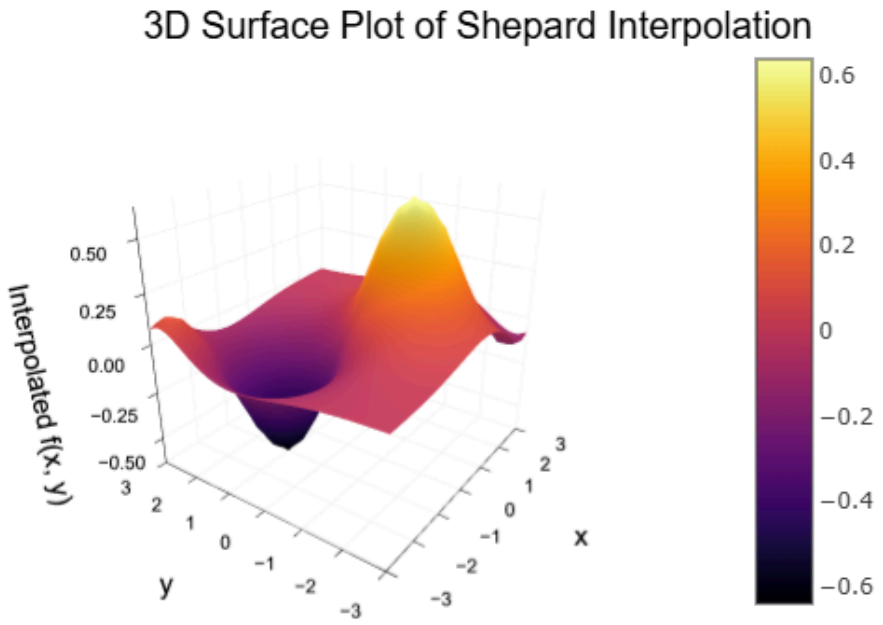
```
# Kriging interpolacija
function kriging_interpolation(x, y, z, x_new, y_new, range=1.0)
    n = length(x)

    # Matrica kovarijanse
    C = covariance_matrix(x, y, range)
    # Dodajemo malo dijagonalu za numeričku stabilnost
    C += 1e-10 * I
    # Vektor kovarijanse za nove tačke
    C_new = covariance_vector(x, y, x_new, y_new, range)
    # Računamo težine
    weights = C \ C_new

    # Izračunavamo vrednost
    z_new = dot(weights, z)
    return z_new
end
```

Uzimajući manji broj tačaka (oko 26), dobijamo sljedeću interpolaciju iste funkcije:



Srednja apsolutna greška: 0.10849686604021033
Srednja relativna greška: 0.996931647401401

Uzimajući veći broj tačaka (oko 56), dobijamo sljedeću interpolaciju iste funkcije:



Srednja apsolutna greška: 0.09976965539097032
Srednja relativna greška: 0.9467871819966818

```
Za tačku (-0.45, 0.45):
Interpolirana vrijednost: -0.398618739585887
Stvarna vrijednost: -0.41313057286996635
Razlika: 0.014511833284079345

Za tačku (-0.25, 0.25):
Interpolirana vrijednost: -0.21152738556558578
Stvarna vrijednost: -0.24354832880354732
Razlika: 0.032020943237961536

Za tačku (-0.1, 0.1):
Interpolirana vrijednost: -0.08315011963488259
Stvarna vrijednost: -0.09958393708102278
Razlika: 0.016433817446140198
```

Ove greške nastaju zbog prirode Kriging interpolacije, koja koristi statistički model za procjenu vrijednosti na nepoznatim tačkama na osnovu poznatih tačaka i njihove prostorne povezanosti. Evo nekoliko ključnih razloga za greške:

1. **Udaljenost od poznatih tačaka**: Kako su tačke koje interpoliramo relativno blizu poznatim tačkama, greške su male, ali postoje zbog razlika u udaljenostima i varijacijama u kovarijansi između tačaka.

2. **Model kovarijanse**: Korištenje eksponencijalne kovarijanse može dati dobre rezultate, ali može doći do greške ako tačke nisu savršeno prostorno povezane prema ovom modelu.
3. **Numerička aproksimacija**: U procesu računanja težina i interpolacije može doći do male greške zbog aproksimacija u matrici kovarijanse.

Iako su greške male, one pokazuju da interpolacija nije savršena, ali daje dovoljno precizne rezultate za bliske tačke.

Krigingova interpolacija je moćna metoda za statističku procjenu i interpolaciju podataka. Evo ključnih tačaka o prednostima, manama i mogućim poboljšanjima Krigingove interpolacije:

Prednosti

1. **Optimalna Procjena**: Kriging pruža optimalnu procjenu u smislu najmanje varijanse. Temelji se na statističkim modelima i koristi kovarijansu za najbolje predviđanje nepoznatih vrijednosti.
2. **Uzimanje u Obzir Neizvesnosti**: Kriging ne samo da predviđa vrijednosti, već i daje meru nesigurnosti kroz standardnu devijaciju interpolacije.
3. **Fleksibilnost u Modeliranju**: Moguće je koristiti različite kovarijanse i modele (npr., eksponencijalni, Gaussovski) što omogućava adaptaciju na različite vrste prostornih raspodjela.

Mane

1. **Računarska Složenost**: Kriging može biti računarski intenzivan, posebno za velike skupove podataka, jer uključuje invertovanje velike kovarijanse matrice.
2. **Zavisnost od Modela**: Tačnost Krigingove interpolacije zavisi od izbora kovarijance i parametara modela. Loš izbor može dovesti do netačnih rezultata.
3. **Preosetljivost na Pretpostavke**: Kriging je osetljiv na pretpostavke o kovarijansi i distribuciji podataka. Ako ove pretpostavke nisu tačne, rezultati mogu biti nepouzdani.

Moguća Poboljšanja

1. **Optimizacija Performansi**: Koristiti efikasnije algoritme za invertovanje matrice, kao što su aproksimacije ili metode učenja sa niskim redom.
2. **Automatsko Podešavanje Parametara**: Implementirati metode za automatsko podešavanje parametara kovarijanse (npr., korišćenje cross-validation) kako bi se poboljšala tačnost modela.
3. **Paralelno Računanje**: Primjena paralelnih metoda za obradu velikih skupova podataka kako bi se smanjilo vrijeme obrade.
4. **Prostorno Analiziranje**: Integrisati dodatne prostorne analize i modeliranje za bolje razumevanje i predstavljanje složenijih prostornih struktura.

Radial Basis Function (RBF)

Uvod

Višedimenzionalna interpolacija je tehnika koja omogućava procjenu vrijednosti funkcije u prostoru na temelju poznatih vrijednosti u više tačaka. Jedna od popularnih metoda za ovakvu interpolaciju je korištenje Radijalne Bazne Funkcije (RBF). Ova metoda ima široku primjenu u područjima kao što su računarski grafika, modeliranje podataka, i rješavanje parcijalnih diferencijalnih jednadžbi.

Radijalna Bazna Funkcija (RBF)

Radijalna Bazna Funkcija (RBF) je funkcija čija vrijednost ovisi samo o udaljenosti od središnje tačke. U kontekstu interpolacije, RBF se koristi za izgradnju glatke funkcije koja prolazi kroz dane tačke u prostoru.

Za n-dimenzionalni prostor, RBF interpolacija se može definirati kao:

$$s(x) = \sum_{i=1}^{N} \lambda_i \cdot \phi(\|x - x_i\|)$$

gdje je $\phi(\,//\, x - x_i \,//\,)$ radijalna bazna funkcija, $//\, x - x_i \,//$ euklidska udaljenost između tačke x i centra x_i, a λ_i koeficijenti koji se određuju rešavanjem sistema linearnih jednačina. Sistem se postavlja tako da interpolaciona funkcija s(x) prolazi kroz sve tačke podataka, što se matematički može zapisati kao:

$$s(\mathbf{x}_j) = y_j, \quad j = 1, \ldots, N$$

Ovo dovodi do sistema N linearnih jednačina sa N nepoznatih λ_i, koji se može rešiti koristeći numeričke metode.

Vrste RBF

Najčešće korištene funkcije ϕ(r) uključuju:

- **Gaussian**: $\phi(r) = e^{-(\epsilon r)^2}$
- **Multiquadric**: $\phi(r) = \sqrt{1 + (\epsilon r)^2}$
- **Inverse Multiquadric**: $\phi(r) = \frac{1}{\sqrt{1+(\epsilon r)^2}}$
- **Linear**: $\phi(r) = r$
- **Cubic**: $\phi(r) = r^3$

Gdje je **r= // x−xi //** **i ϵ** je parametar skaliranja.

Primjena

RBF interpolacija se koristi u mnogim područjima, uključujući:

1. **Modeliranje i predikcija**: u strojnim učenjima za aproksimaciju nepoznatih funkcija.
2. **Numerička analiza:** Za rešavanje parcijalnih diferencijalnih jednačina na nepravilnim mrežama.
3. **Geostatistika**: RBF se koristi za prostornu interpolaciju geoloških podataka.
4. **Kompjuterska grafika:** Za modeliranje glatkih površina i deformacija u 3D prostoru.
5. **Mašinsko učenje:** RBF se koristi kao aktivaciona funkcija u RBF neuronskim mrežama za klasifikaciju i regresiju.

Pseudokod za RBF Interpolaciju

Evo jednostavnog pseudokod i Implementacija u Julia jeziku za RBF interpolaciju:

```julia
using LinearAlgebra
using Plots
# Funkcija za RBF interpolaciju
function rbf_interpolation(X, Y, x_nova, phi, lambda=nothing)
    N = length(X)
    A = zeros(N, N)
    for i in 1:N
        for j in 1:N
            A[i, j] = phi(norm(X[i] - X[j]))
        end
    end
    if lambda != nothing
        A += lambda * I(N)
    end
    w = A \ Y
    y_nova = 0.0
    for i in 1:N
        y_nova += w[i] * phi(norm(x_nova - X[i]))
    end
    return y_nova
end
```

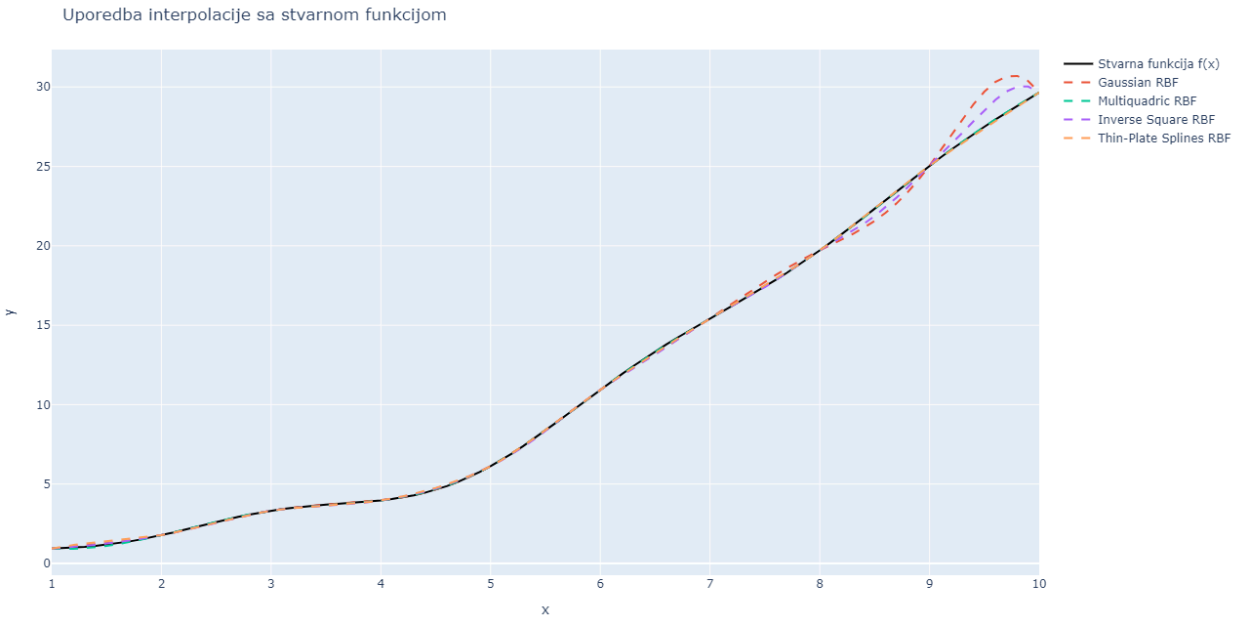
```julia
# Definišemo složeniju funkciju f(x) za generisanje Y vrednosti
function f(x)
    return sin(x) + 0.5 * cos(2 * x) + 0.3 * x^2  # Složena funkcija
end
# Definisanje RBF kernel funkcija
function gaussian_rbf(r, epsilon=1.0)
    return exp(-(epsilon * r)^2)
end

function multiquadric_rbf(r, c=1.0)
  return sqrt(r^2 + c^2)
end

function inverse_square_rbf(r, c=1.0)
  return 1.0 / (r^2 + c^2)
end

function thin_plate_splines_rbf(r)
  return r^2 * log(r + 1e-10)  # Dodajemo malu vrijednost da izbjegnemo log(0)
end
# Ulazni podaci
X = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0]  # Više tačaka
Y = [f(x) for x in X]  # Generisanje Y vrednosti koristeći funkciju f
x_nova = 5.5  # Nova tačka za interpolaciju
```

Ps. **Thin-plate splines $\phi(r)= r^2 \log(r)$** Ova funkcija je često korištena za glatku interpolaciju u 2D i 3D prostoru.

Rezultati:

```
Tačka za interpolaciju: 5.5
Stvarna vrijednost: 8.371672523423634
Interpolirana vrijednost sa Gaussian RBF funkcijom: 8.411014646367237,
Greška: 0.039342122943603286
Interpolirana vrijednost sa Multiquadric RBF funkcijom: 8.37256880313647,
Greška: 0.0008962797128369004
Interpolirana vrijednost sa Inverznim kvadratnim RBF funkcijom:
8.334338505669075, Greška: 0.037334017754558246
Interpolirana vrijednost sa Thin-plate splines RBF funkcijom:
8.387381181697904, Greška: 0.015708658274270704
```

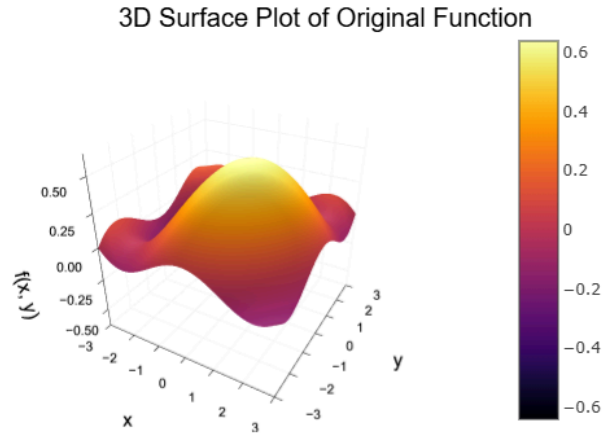
- **Multiquadric RBF** funkcija je pružila najtačnije rezultate u našem slučaju zbog svoje sposobnosti da uravnoteži lokalne i globalne informacije.
- **Gaussian RBF** može biti previše "smirena" za složene funkcije.
- **Inverse Square RBF** previše naglašava bliske tačke.
- **Thin-Plate Splines RBF** može biti bolja za glatke funkcije, ali lošija za složene promjene.

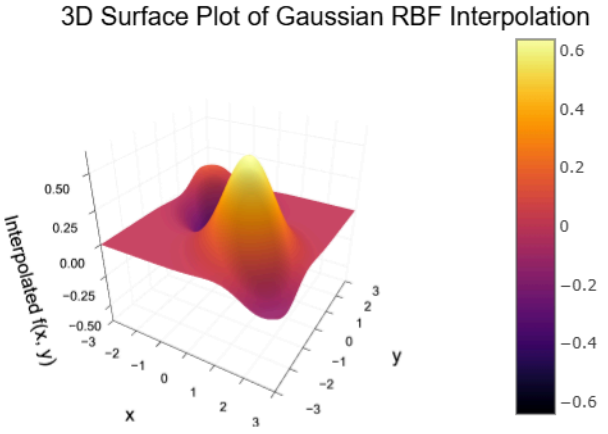


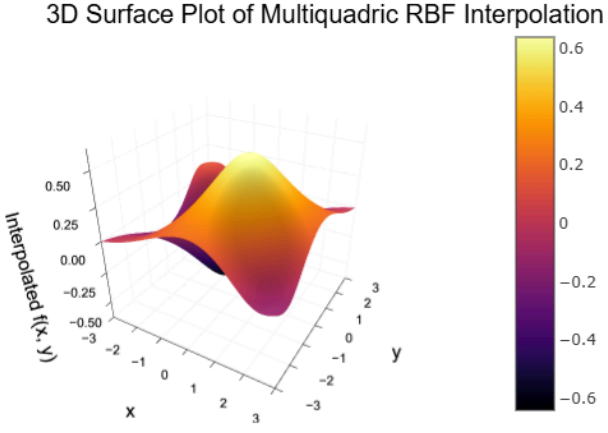
Sada uz neke sitne izmjene unutar funkcije rbf_interpolation mozemo interpolirati u 3D-u:

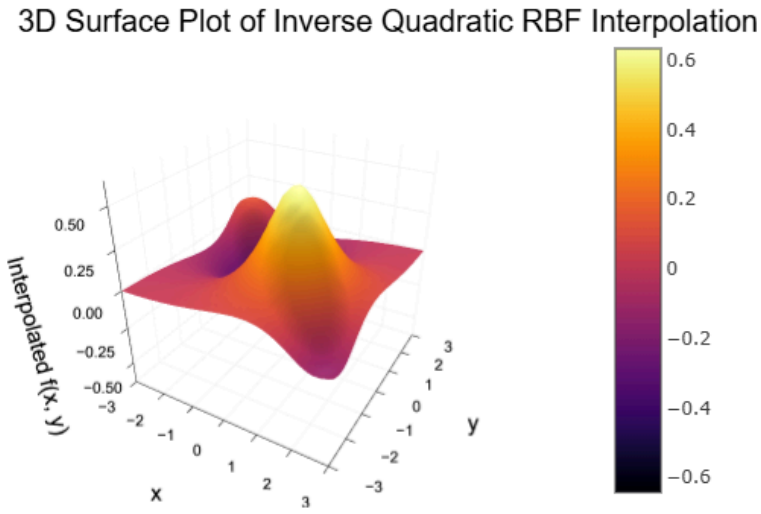
(Ako znamo da nam je funkcija nad kojom interpoliramo glasi)

```
function f(x, y)
    return sin((x - y) / 2) * cos(sqrt(x^2 + y^2) / 2)
end
```


3D Surface Plot of Original Function
f(x, y)
0.50
0.25
0.00
−0.25
−0.50
x
y
0.6
0.4
0.2
0
−0.2
−0.4
−0.6


3D Surface Plot of Gaussian RBF Interpolation
Interpolated f(x, y)
0.50
0.25
0.00
−0.25
−0.50
x
y
0.6
0.4
0.2
0
−0.2
−0.4
−0.6


3D Surface Plot of Multiquadric RBF Interpolation
Interpolated f(x, y)
0.50
0.25
0.00
−0.25
−0.50
x
y
0.6
0.4
0.2
0
−0.2
−0.4
−0.6



Srednja apsolutna greška za Gaussian RBF: 0.11501791139384517
Srednja relativna greška za Gaussian RBF: 1.885539839808223e7

Srednja apsolutna greška za Multiquadric RBF: 0.04787499884241526
Srednja relativna greška za Multiquadric RBF: 7.848361751157399e6

Srednja apsolutna greška za Inverse Quadratic RBF: 0.08970447850824839
Srednja relativna greška za Inverse Quadratic RBF: 1.4705654622552458e7

Zaključak:

- Multiquadric RBF se pokazao kao najprecizniji među ovim RBF funkcijama, sa najnižom srednjom apsolutnom greškom i relativnom greškom.
- Gaussian RBF i Inverse Quadratic RBF imaju višu relativnu grešku, što može značiti da interpolacija nije bila tako precizna, posebno u odnosu na veličinu stvarnih vrijednosti.

Najveća Prednost RBF Interpolacije

Jedna od najvećih prednosti RBF interpolacije je njena sposobnost da se nosi sa nepravilno raspoređenim podacima u višedimenzionalnim prostorima. Za razliku od mnogih drugih interpolacionih metoda, RBF ne zahtjeva redovnu mrežu tačaka, što omogućava upotrebu u raznim praktičnim situacijama gde podaci mogu biti raspoređeni na nepravilan način. Takođe, RBF može proizvesti glatke i diferencijabilne površine, što je korisno u aplikacijama kao što su modelovanje površina i analize u mašinskom učenju.

Najveća Mana RBF Interpolacije

Najveća mana RBF interpolacije je njena računska složenost, posebno kada se radi sa velikim skupovima podataka. Sistemi linearnih jednačina koje je potrebno rješiti za određivanje koeficijenata λi imaju složenost $\mathbf{O(N^3)}$ za direktno rešavanje, gde je $\boldsymbol{N}$ broj tačaka podataka. Ovo može postati vrlo zahtjevno za računare kada broj tačaka postane veliki. Takođe, izbor parametra $\boldsymbol{\epsilon}$ (koji određuje širinu RBF funkcije) može biti izazovan jer zavisi od specifične aplikacije i može značajno uticati na rezultate interpolacije.

Zaključak

Radijalna Bazna Funkcija (RBF) interpolacija je moćna tehnika koja omogućava glatku interpolaciju podataka u višedimenzionalnim prostorima. Njena fleksibilnost i primjenjivost u raznim domenama čini je ključnim alatom u numeričkoj analizi i računarskim znanostima. Korištenjem programskog jezika Julia, možemo lako implementirati i vizualizirati RBF interpolaciju, što dodatno potvrđuje njenu praktičnost u modernom računanju.

Popis literature

1. Shepardova Interpolacija

- Samra Salihić: Algoritmi za višedimenzionalnu interpolaciju [4] (str. 16)

2. Kriging (Geostatistička interpolacija)

- Samra Salihić: Algoritmi za višedimenzionalnu interpolaciju [4] (str. 17..19)
- https://juliaearth.github.io/geospatial-data-science-with-julia/ [1]

3. Radijalna Bazna Funkcija (RBF)

- Samra Salihić: Algoritmi za višedimenzionalnu interpolaciju [4] (str. 14, 15)
- Radial Basis Function Interpolation by Wilna du Toit [4]