Interpolacija ploskev

3. domača naloga pri Matematičnem modeliranju

Anže Mur

2. maj 2018

1 Opis problema

Imamo matriko V, ki predstavlja vrednosti funkcije $f(i, j) = V_{ij}$ v (celoštevilskih) točkah (i, j). Radi bi definirali gladko (zvezno odvedljivo) funkcijo $f(x, y)$, ki interpolira vrednosti f za vmesne realne argumente (x, y). Z drugimi besedami, iz podatkov kakršni so prikazani na Figure 1, bi radi znali dobiti gladko ploskev kot je prikazana na Figure 2.

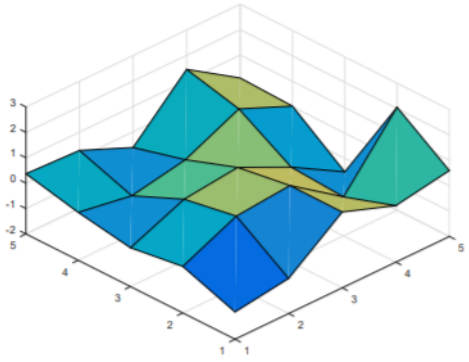

Figure 1: Naključna 5 x 5 matrika podatkov kot jo izriše ukaz $surf$, ki vmesne vrednosti samodejno interpolira z linearnimi funkcijami.

Ideja je, da funkcijo na celotnem območju [1, n] x [1, m] sestavimo kot zlepek funkcij, ki so definirane na posameznih kvadratih enotske velikosti. Pri tem pa je seveda potrebno poskrbeti, da se bodo vrednosti funkcije f in tudi vrednosti njenih odvodov na robu posameznega kvadrata povsod ujemale z vrednostmi in odvodi na robovih sosednjih kvadratov. To je možno storiti na več načinov, ideja rešitve, ki jo morate v nalogi implementirati, pa temelji na pojmu t.i. razčlenitve enote.

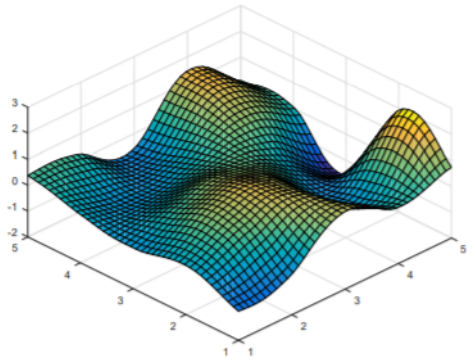

Figure 2: Gladka interpolacijska funkcija $f(x, y)$, ki je definirana povsod na [1, 5] x [1, 5] in ki se za celoštevilske argumente ujema z vrednostmi, ki so prikazane na Figure 1.

2 Naloge

1. Naloga: Poiščite polinom 3. stopnje $p(x)$, ki zadošča pogojem:

$$p(0) = 1, p'(0) = 0, p(1) = 0, p'(1) = 0$$

Rešitev:
Enačba polinoma tretje stopnje:

$$p(x) = ax^3 + bx^2 + cx + d$$

a) p(0) = 1

$$p(0) = d$$
$$d = 1$$

b) p'(0) = 0

$$p'(x) = 3ax^2 + 2bx + c$$
$$p'(0) = c$$
$$c = 0$$

c) p(1) = 0

$$p(1) = a + b + 1$$
$$0 = a + b + 1$$
$$a = -1 - b$$

d) p'(1) = 0

$$p'(1) = 3a + 2b$$

$$0 = 3a + 2b$$
$$0 = -3 - 3a + 2b$$
$$b = -3$$

Končna oblika polinoma $p(x)$:

$$p(x) = 2x^3 - 3x^2 + 1$$

2. Naloga: Napišite učinkovito funkcijo $Z = interpolationFunction(data, len)$, ki na len x len mreži enakomerno razporejenih točk na enotskem kvadratu K izračuna vrednosti funkcije, kjer spremenljivka data vsebuje vseh 12 potrebnih podatkov, in izračunane vrednosti vrne v len x len matriki Z. Kako podatke razporedite v strukturo data je vaša izbira (lahko je stolpec, vrstica, matrika ali . . .).

Rešitev v octave-u:

```
function Z = interpolationFunction(data, len)
%Z = interploatioinFunction(data, len) je funkcija, ki na enakomerno
%razporejeni mrezi tock dimenzije len x len izracuna vrednosti
%interpolacijske funkcije f.
%Vhodni parametri:
%data -> vsebuje 12 potrebnih podatkov za izracun interpolacijske
%       funkcije f.
%data = [a, b, c, d, a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y]
%len -> stevilo, ki pove na koliko manjsih kvadratov razdelimo
%       enotski kvadrat ter kako natancna/zglajena naj bo interpolacija



%funkcija za izracun polinoma ki ustreza pogojem:
% p(0) = 1
% p'(0) = 0
% p(1) = 1
% p'(1) = 0
p = @(x) 2*x^3 - 3*x^2 + 1;

%preberemo podatke iz data
a = data(1);
b = data(2);
c = data(3);
d = data(4);

a_x = data(5);
b_x = data(6);
c_x = data(7);
d_x = data(8);

a_y = data(9);
b_y = data(10);
```

```
c_y = data(11);
d_y = data(12);

%funckije za izracun utezi
W_A = @(x, y) p(x)*p(y);
W_B = @(x, y) (1 - p(x))*p(y);
W_C = @(x, y) p(x)*(1 - p(y));
W_D = @(x, y) (1 - p(x))*(1-p(y));

%funckije za izracun pomoznih funckij
f_A = @(x, y) a + a_x*x + a_y*y;
f_B = @(x, y) b + b_x*(x - 1) + b_y*y;
f_C = @(x, y) c + c_x*x + c_y*(y - 1);
f_D = @(x, y) d + d_x*(x - 1) + d_y*(y - 1);

%interpolacijska funkcija f
f = @(x, y) f_A(x, y)*W_A(x, y) + f_B(x, y)*W_B(x, y) + f_C(x, y)*W_C(x,
    y) + f_D(x, y)*W_D(x, y);

x = 0;
y = 0;

iter = 1/(len-1);

for idV = 1 : len
   x = 0;

   for idS = 1 : len
      Z(idV, idS) = f(x, y);

      x = x + iter;
   endfor

   y = y + iter;

endfor

%!demo
%! data = [1, 5, 5, 1, 0.4, 0.2, 0.4, 0.2, 0.2 ,0.4 ,0.2, 0.4];
%! len = 20;

%! figure(1)
%! surf([1, 5; 5, 1,])

%! Z = interpolationFunction(data, len);
%! figure(2)
%! surf(Z);

endfunction
```

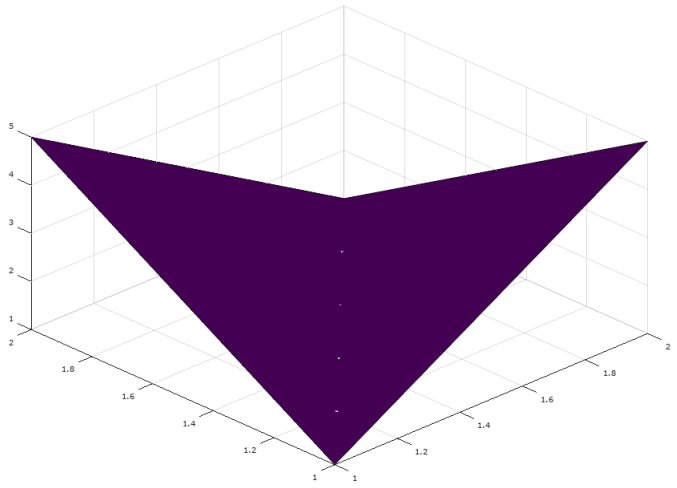

Figure 3: Prikaz delovanja funckije interpolationFunction pred klicem.

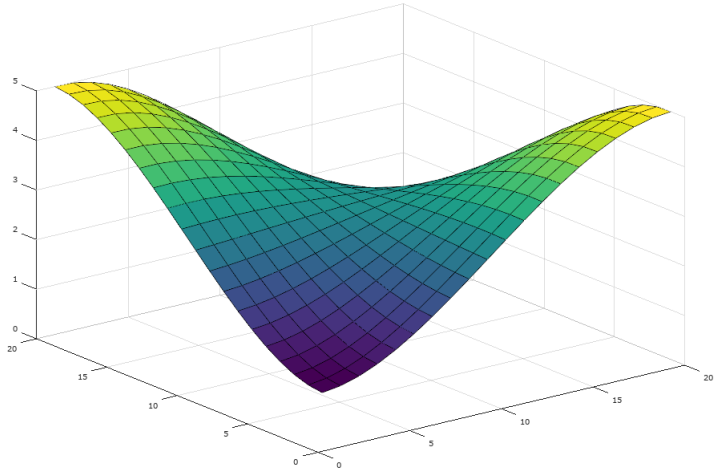

Figure 4: Prikaz delovanja funckije interpolationFunction po klicu.

3. Naloga :
a) Pokažite, na čimbolj enostaven način, da za funkcije povsod na K velja:

$$w_A + w_B + w_C + w_D = 1$$

(od tu ime razčlenitev enote). Ta enakost zagotavlja, da bo v primeru, da so podatki konstantni (ali pa linearna funkcija (i, j)) tudi f povsod konstantna (oz. linearna), kar je tudi najbolj smiselno. Utemeljite zadnjo trditev!

b) Izračunajte vrednosti funkcije f in njenih parcialnih odvodov v ogliščih. Dovolj je pokazati, kaj dobimo za eno oglišče.

c) Pokažite tudi, da so vrednosti funkcije f in njenih odvodov na dani stranici kvadrata K odvisne samo od podatkov, ki se nanašajo na oglišči te stranice. Tudi tu je dovolj to pokazati za eno stranico.

Rešitev:
a)

$$w_A + w_B + w_C + w_D = 1$$
$$p(x)p(y) + (1 - p(x))p(y) + p(x)(1 - p(y)) + (1 - p(x))(1 - p(y)) = 1$$
$$p(x)p(y) + p(y) - p(x)p(y) + p(x) - p(x)(y) + 1 - p(y) - p(x) + p(x)(y) = 1$$
$$1 = 1$$

Zgornja enakost zagotavlja, da je vpliv uteži na posamezno točko (x, y) sorazmeren z razdaljo te točke od ogljišč enotksega trikotnika ter to zagotavlja pravilno razmerje med funkcijami. Enakost mora veljati, saj mora glede na koordinate vsaka utež prinesti uteženo vrednost glede na oddaljenost od točke (bližje kot smo uteži večji vpliv ima). Če to nebi veljalo, bi bile uteži nesmiselne, saj nebi dodale pravilne vrednosti in bi se razmerje med funkcijami pokvarilo.

b)
$A(0, 0)$
$x = 0$
$y = 0$
$p(0) = 1$

Uteži:

$$w_A(x, y) = p(x)p(y)$$
$$w_B(x, y) = (1 - p(x))p(y)$$
$$w_C(x, y) = p(x)(1 - p(y))$$
$$w_D(x, y) = (1 - p(x))(1 - p(y))$$

Enačbe:

$$f_A(x, y) = a + a_x x + a_y y$$
$$f_B(x, y) = b + b_x(x - 1) + b_y y$$

$$f_C(x,y) = c + c_x x + c_y(y-1)$$
$$f_D(x,y) = d + d_x(x-1) + d_y(y-1)$$

$$f = f_A w_A + f_B w_B + f_C w_C + f_D w_D$$

Izpeljava:

$$w_A(0,0) = 1 \cdot 1 = 1$$
$$w_B(0,0) = 0 \cdot 1 = 0$$
$$w_C(0,0) = 1 \cdot 0 = 0$$
$$w_D(0,0) = 0 \cdot 0 = 0$$

$$f = f_A \cdot 1 + f_B \cdot 0 + f_C \cdot 0 + f_D \cdot 0$$
$$f = a + a_x \cdot 0 + a_y \cdot 0$$
$$f = a$$

c)
Stranica: A(0,0) B(1,0)
x = x
y = 0

Izpeljava:

$$w_A(x,0) = p(x) \cdot 1 = p(x)$$
$$w_B(x,0) = (1 - p(x)) \cdot 1 = 1 - p(x)$$
$$w_C(x,0) = p(x) \cdot 0 = 0$$
$$w_D(x,0) = (1 - p(x)) \cdot 0 = 0$$

$$f = f_A \cdot 1 + f_B \cdot (1 - p(x)) + f_C \cdot 0 + f_D \cdot 0$$
$$f = (a + a_x x + a_y y) \cdot p(x) + (b + b_x(x-1) + b_y y) \cdot (1 - p(x))$$
$$f = (a + a_x x) \cdot p(x) + (b + b_x(x-1)) \cdot (1 - p(x))$$

4. Naloga :
Napišite funkcijo $interpolation(V, len)$, ki kot na slikah 1 in 2 izriše podatke iz matrike V (z ukazom surf ali mesh) in nato še sliko interpolacijske ploskve, ki jo dobimo z lepljenjem funkcij za posamezne kvadrate mreže. Tu je len enak kot pri funkciji $interpolationFunction$, torej število delilinih točk v x in y smeri pri izračunu za posamezen kvadrat (na sliki 2 je npr. len=10). Pri tem je pri klicih funkcije $interpolationFunction$ potrebno smiselno izbrati vrednosti odvodov v oglišcih kvadratov. Bistveno je, da se argumenti pri klicu funkcije, ki nanašajo na oglišči stranice posameznega kvadrata, ujemajo z argumenti za to isto stranico pri klicu funkcije za sosednji kvadrat. Naravna izbira za parcialni odvod funkcije v danem oglišču v določeni smeri je odvod linearne funkcije skozi sosednji oglišči v tisti smeri.

Rešitev v octave-u:

```
function interpolation(V,len)
%interpolation(V, len) je funkcija, ki sprejme dolzino delitve kvadratov
%len ter matriko V v kateri je predstavljena neka funkcija, ki jo bomo
%interopolirali.
%Funkcija najprej izrise vhodne podatke nato pa izrise se interpolirano
%matriko vhodne matrike V.

%izris matrike V
figure(1);
surf(V);
title("Vhodna matrika V");

%matrika za shranjevanje interpoliranih vrednosti
out = zeros(length(V)*len - 1 - len);
data = [];

%funkcije za racunanje parcialnih odvodov x in y
df_x = @(x,y) (x-y)/2;
df_y = @(x,y) (x-y)/2;


idRow = 1;
idCol = 1;

iter = 1/len;

for j = 1:length(V) - 1
   idCol = 1;

   for i = 1:length(V) - 1

      %v data shranimo a, b, c, d
      data = [V(i,j),V(i+1,j),V(i,j+1),V(i+1,j+1)];


      %d_x = [d_ax, d_bx, d_cx, d_dx]
      d_x = [0 0 0 0];
```

```
%d_y = [d_ay, d_by, d_cy, d_dy]
d_y = [0 0 0 0];



%zgornji levi kot
if(i == 1 && j == 1)

   d_x = [V(i+1,j) - V(i,j), df_x(V(i,j),V(i+2,j)), V(i+1,j+1) -
       V(i,j+1), df_x(V(i,j+1),V(i+2,j+1))];
   d_y = [V(i,j+1)-V(i,j), V(i+1,j+1)-V(i+1,j),
       df_y(V(i,j),V(i,j+2)), df_y(V(i+1,j),V(i+1,j+2))];

%spodnji levi kot
elseif((i == 1 && j == length(V) - 1))

   d_x = [V(i+1,j) - V(i,j), df_x(V(i,j),V(i+2,j)), V(i+1,j+1) -
       V(i,j+1), df_x(V(i,j+1),V(i+2,j+1)) ];
   d_y = [df_y(V(i,j-1),V(i,j+1)), df_y(V(i+1,j-1),V(i+1,j+1)),
       V(i,j+1)-V(i,j), V(i+1,j+1)-V(i+1,j) ];

%zgornji desni kot
elseif((i == length(V) - 1 && j == 1))

   d_x = [df_x(V(i-1,j),V(i+1,j)), V(i+1,j) - V(i,j) ,
       df_x(V(i-1,j+1),V(i+1,j+1)), V(i+1,j+1)-V(i,j+1)];
   d_y = [V(i,j+1)-V(i,j), V(i+1,j+1)-V(i+1,j),
       df_y(V(i,j),V(i,j+2)), df_y(V(i+1,j),V(i+1,j+2))];

%spodnji desni kot
elseif((i == length(V) - 1 && j == length(V) - 1))

   d_x = [df_x(V(i-1,j),V(i+1,j)), V(i+1,j) - V(i,j),
       df_x(V(i-1,j+1),V(i+1,j+1)), V(i+1,j+1)-V(i,j+1)];
   d_y = [df_y(V(i,j-1),V(i,j+1)), df_y(V(i+1,j-1),V(i+1,j+1)),
       V(i,j+1)-V(i,j), V(i+1,j+1)-V(i+1,j)];

%levi rob
elseif(i == 1)

   d_x = [V(i+1,j)-V(i,j), df_x(V(i,j),V(i+2,j)),
       V(i+1,j+1)-V(i,j+1), df_x(V(i,j+1),V(i+2,j+1))];
   d_y = [df_y(V(i,j-1),V(i,j+1)), df_y(V(i+1,j-1),V(i+1,j+1)),
       df_y(V(i,j),V(i,j+2)), df_y(V(i+1,j),V(i+1,j+2))];

%desni rob
elseif(i == length(V) - 1)

   d_x = [df_x(V(i-1,j),V(i+1,j)), V(i+1,j)-V(i,j),
       df_x(V(i-1,j+1),V(i+1,j+1)), V(i+1,j+1)-V(i,j+1)];
   d_y = [df_y(V(i,j-1),V(i,j+1)), df_y(V(i+1,j-1),V(i+1,j+1)),
       df_y(V(i,j),V(i,j+2)), df_y(V(i+1,j),V(i+1,j+2))];

%zgornji rob
```

```
      elseif(j == 1)

         d_x = [df_x(V(i-1,j),V(i+1,j)), df_x(V(i,j),V(i+2,j)),
              df_x(V(i-1,j+1),V(i+1,j+1)), df_x(V(i,j+1),V(i+2,j+1))];
         d_y = [V(i,j+1)-V(i,j), V(i+1,j+1)-V(i+1,j),
              df_y(V(i,j),V(i,j+2)), df_y(V(i+1,j),V(i+1,j+2))];

      %spodnji rob
      elseif(j == length(V) - 1)

         d_x = [df_x(V(i-1,j),V(i+1,j)), df_x(V(i,j),V(i+2,j)),
              df_x(V(i-1,j+1),V(i+1,j+1)), df_x(V(i,j+1),V(i+2,j+1))];
         d_y = [df_y(V(i,j-1),V(i,j+1)), df_y(V(i+1,j-1),V(i+1,j+1)),
              V(i,j+1)-V(i,j), V(i+1,j+1)-V(i+1,j)];

      else

         d_x = [df_x(V(i-1,j),V(i+1,j)), df_x(V(i,j),V(i+2,j)),
              df_x(V(i-1,j+1),V(i+1,j+1)), df_x(V(i,j+1),V(i+2,j+1))];
         d_y = [df_y(V(i,j-1),V(i,j+1)), df_y(V(i+1,j-1),V(i+1,j+1)),
              df_y(V(i,j),V(i,j+2)), df_y(V(i+1,j),V(i+1,j+2))];

      endif

      %v data shranimo vrednosti parcialnih odvodov
      data = [data, d_x, d_y];
      %interpoliramo trenutni enotski kvadrat
      out(idRow:idRow+len-1,idCol:idCol+len-1) =
          interpolationFunction(data,len);

      idCol += len-1;
   endfor
   idRow += len-1;
endfor

figure(2);
surf(out);
title(["Interpolirana matrika V s parametrom len=" num2str(len)]);

%!demo
%! len = 20;
%! V = [1 1 1 1 1 1 1 1 1;
%!      1 6 3 2 5 2 3 6 1;
%!      1 8 2 1 7 1 2 8 1;
%!      1 4 1 2 9 2 1 4 1;
%!      1 7 3 6 12 6 3 7 1;
%!      1 4 1 2 9 2 1 4 1;
%!      1 8 2 1 7 1 2 8 1;
%!      1 6 3 2 5 2 3 6 1;
%!      1 1 1 1 1 1 1 1 1];
%! interpolation(V, len);

endfunction
```

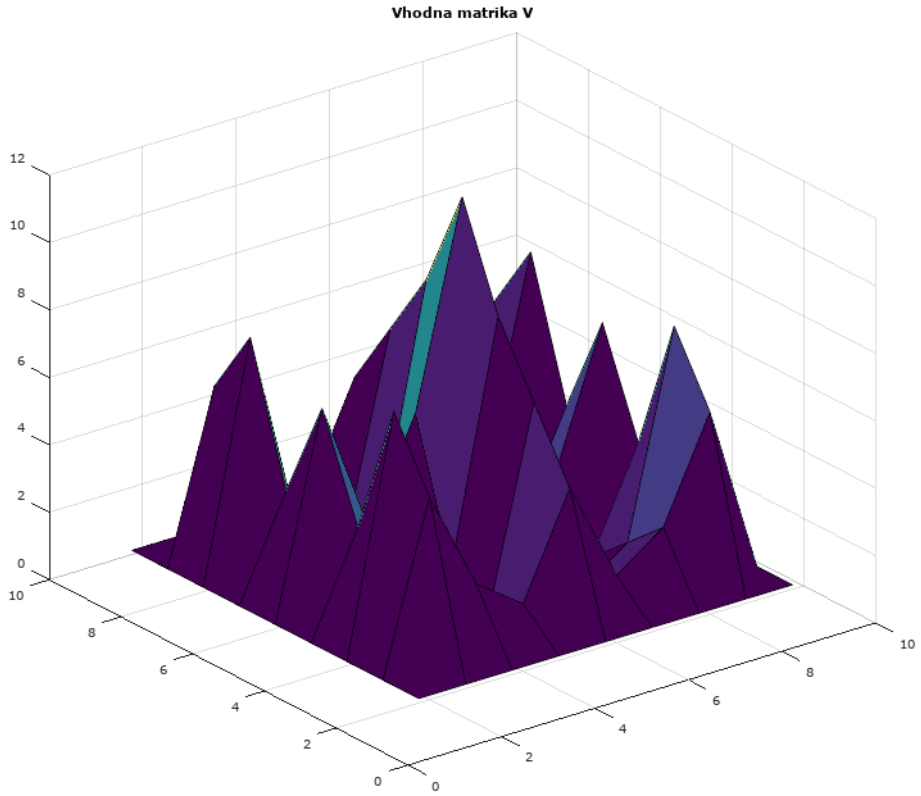

Figure 5: Matrika vhodnih podatkov V, ki jo dobimo z zagonom *test interpolation*, ki vmesne vrednosti samodejno interpolira z linearnimi funkcijami pri dolzini len = 20.

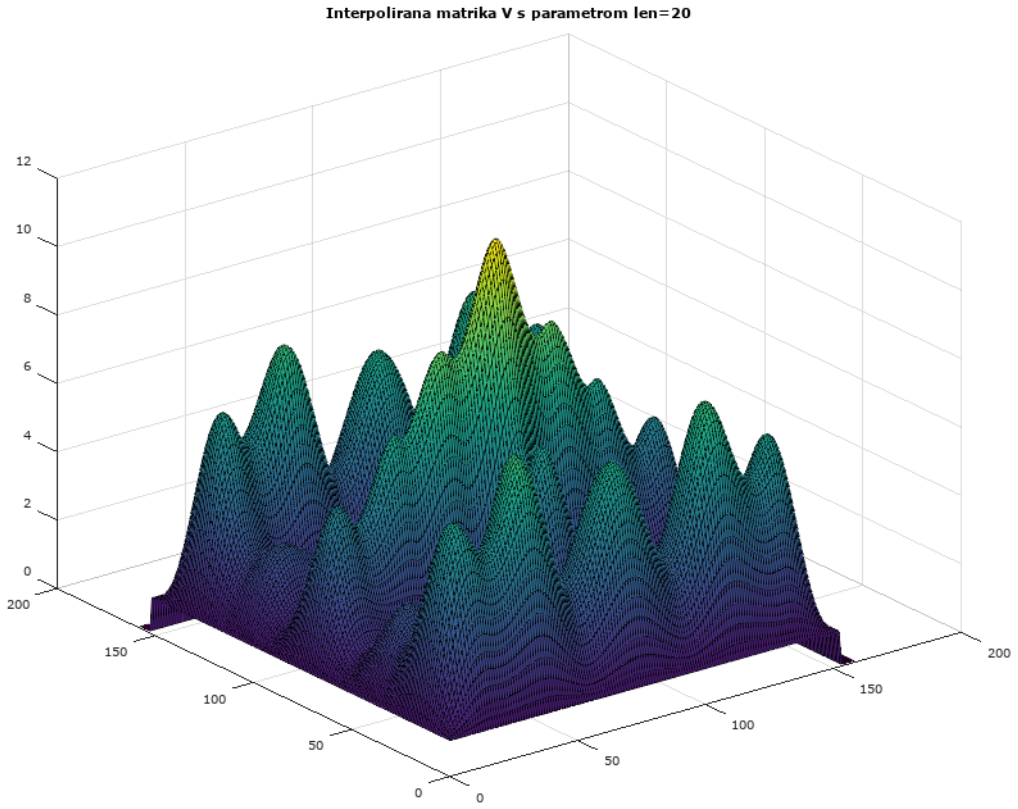

Figure 6: Matrika vhodnih podatkov V, ki jo dobimo z zagonom *test interpolation*, ki je gladko interpolirana s funkcijo $f(x, y)$ ter se za celoštevilske argumente ujema z vrednostmi, ki so prikazane na Figure 4.