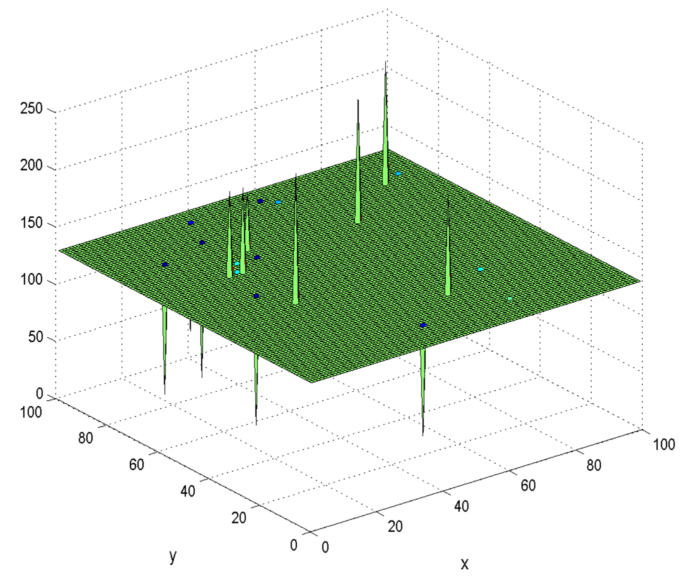
# Project: Interpolation for Scattered Data

MA-4313/6313 Numerical Analysis I (2014 Fall)

**Due: Thu 11/6**

Interpolation techniques are often required in various tasks, particularly in image processing, computer vision, and finite element simulation. It is the first of the two basic resampling steps and transforms a discrete data into a continuous function. (The second step of the resampling is to produce discrete data by evaluating the continuous function on a new set of points.) Most of interpolation methods introduce artifacts such as ringing and image blur. In order to reduce artifacts, many methods have been studied in the literature. However, more effective methods are yet to be developed, particularly for scattered data (point cloud data).

Here we will consider an example. Let a set of point cloud data be given as follows.

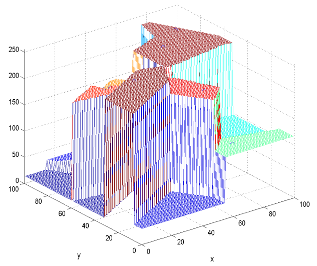


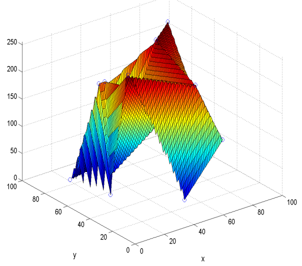
Matlab has a build-in function for the interpolation of such a data set.

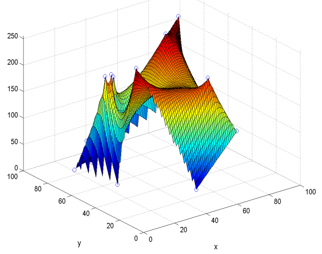
```
F=TriScatteredInterp(x,y,z);

%F.Method='nearest'; % nearest neighbor interpol: discontinuous.
%F.Method='linear';  % linear interpol: C0 continuous (default)
F.Method='natural'; % natural neighbor interpol: C1 continuous except
                    % at the scattered data locations.
```

The results are:







None of above images are satisfactory.

In this project, we will consider interpolation methods for scattered data. In various engineering experiments, data values are acquired from nonuniformly spaced points and a continuous surface must be effectively constructed involving those data values. One of most popular surface construction methods is the so-called **inverse-distance weighting** (IDW) algorithm:

$$u(x) = \frac{\Sigma_k\left(w(x_k)\,u(x_k)\right)}{\Sigma_k w(x_k)} \qquad \textbf{(1)}$$

where $u(x)$ is the estimated value at $x$, $u(x_k)$ is the value at a neighboring point $x_k$, and $w(x_k)$ denotes the weight for $u(x_k)$ defined as

$$w(x_k) = \frac{1}{\|x - x_k\|^p}.$$

Here $\|\cdot\|$ is the Euclidean norm and $p$ is an exponential number greater than or equal to 2.

## <u>Objectives</u>

- Make your own group, if you want.
- Adjust "InterpolIDW" (a modelcode) to incorporate variable window size (InterpolIDW2).
- Analyze your new code with various *rwin*: 2,3,4,5, and 6.
- If you work in a group, you need to analyze the algorithms with more details. For example, you may analyze your code with various *samplerate*: 20, 30, and 50.
- Report your implementation and results, along with **your final thought** on the experience.

```
## ------------------------------
## Modelcode for project
## ------------------------------

restart
with(LinearAlgebra) : with(CurveFitting) :

RandomPositioning :=proc(n, a, b, c, d, R, IR, samplerate)
   local roll, Delx, Dely, count, samplenumber, ix, iy, px, py;
   roll := rand(1 ..n + 1);
   Delx := (b − a) / n; Dely := (d − c) / n;
   samplenumber := MTM[int32]( (n + 1)^2·samplerate / 100);
   print( `samplerate=`, samplerate, `samplenumber=`, samplenumber);

  count := 0;
  while count < samplenumber do
     ix := roll( ) :  iy := roll( ) :
     if (IR[ix, iy] = 0) then
        count := count + 1;
        IR[ix, iy] := 1;
        px := (ix − 1) ·Delx : py := (iy − 1) ·Dely :
        R[ix, iy] := evalf[14]( eval(f, [x = px, y = py]) ) :
      end if;
    end do;
    print( `count=`, count);
end proc:

InterpolIDW :=proc(n, rwin, mink, p, R, IR)
   local ix, iy, i, j, i0, i1, j0, j1;
   local count, wgt, top, bot, hp;
   print( `n=`, n, `rwin=`, rwin, `mink=`, mink, `p=`, p);
   hp := p / 2;

   for iy from 1 to n do
   for ix from 1 to n do
      if (IR(ix, iy) = 0) then
         i0 := max(1, ix − rwin); i1 := min(n, ix + rwin);
         j0 := max(1, iy − rwin); j1 := min(n, iy + rwin);
         top := 0; bot := 0;
         for j from j0 to j1 do
         for i from i0 to i1 do
            if (IR(i, j) ≠ 0) then
               wgt := 1 / evalf_14( ( (ix − i)^2 + (iy − j)^2 )^hp );
               top := top + wgt·R(i, j);
               bot := bot + wgt;
            end if;
         end do;
         end do;
```

```
            R(ix, iy) := top/bot;
        end if;
    end do;
    end do;

end proc:

InterpolIDW2 :=proc(n, rwin, mink, p, R, IR)
    local ix, iy, i, j, i0, i1, j0, j1;
    local count, wgt, top, bot, hp;
    print(`n=`, n, `rwin=`, rwin, `mink=`, mink, `p=`, p);
    hp := p/2;

    for iy from 1 to n do
    for ix from 1 to n do
        if (IR(ix, iy) = 0) then
            i0 := max(1, ix − rwin);  i1 := min(n, ix + rwin);
            j0 := max(1, iy − rwin);  j1 := min(n, iy + rwin);

            ## -------------------------------------------------------
            ## ``Add lines here for  "InterpolIDW2"
            ## -------------------------------------------------------

            top := 0;  bot := 0;
            for j from j0 to j1 do
            for i from i0 to i1 do
                if (IR(i, j) ≠ 0) then
                    wgt := 1/evalf_14(((ix − i)^2 + (iy − j)^2)^hp);
                    top := top + wgt·R(i, j);
                    bot := bot + wgt;
                end if;
            end do;
            end do;
            R(ix, iy) := top/bot;
        end if;
    end do;
    end do;

end proc:
```

**InterpolIDW2_Kim**(*n*, *rwin*, *mink*, *p*, *R*, *IR*)
### Deleted

```
ErrCheck :=proc(n, a, b, c, d, R)
   local Delx, Dely, ix, iy, px, py, ftrue, l2err;
   Delx := (b - a) / n : Dely := (d - c) / n :
   l2err := 0 :
  for iy from 1 to n + 1 do
  for ix from 1 to n + 1 do
     px := (ix - 1) ·Delx : py := (iy - 1) ·Dely :
     ftrue := evalf[14]( eval(f, [x = px, y = py]) ) ;
     l2err := l2err + (ftrue - R(ix, iy))^2;
  end do;
  end do;
  l2err := sqrt(l2err / (n + 1)^2);
  printf(" L2_Error=%g\n", l2err);
 end proc:
```

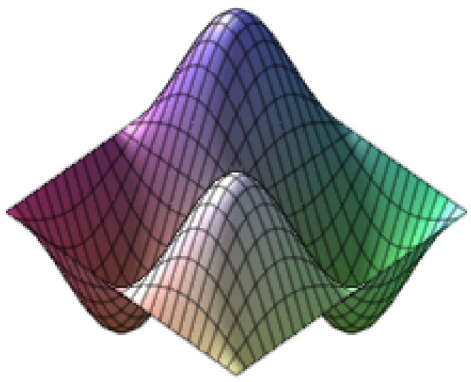
```
> n := 100
                                n := 100                                    (2)
> a := 0; b := 1; c := 0; d := 1
                                 a := 0
                                 b := 1
                                 c := 0
                                 d := 1                                     (3)
> samplerate := 20
                            samplerate := 20                                (4)
> f := sin(2 π x) ·sin(2 π y)
                        f := sin(2 π x) sin(2 π y)                          (5)
R := Matrix(n + 1, n + 1) :
IR := Matrix(n + 1, n + 1) :
RandomPositioning(n, a, b, c, d, R, IR, samplerate) :
                 samplerate=, 20, samplenumber=, 2040
                              count=, 2040                                  (6)
IR[n/2 - 1 ..n/2 + 3, n/2 - 1 ..n/2 + 3]
```

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad (7)$$

```
R[n/2 - 1 ..n/2 + 3, n/2 - 1 ..n/2 + 3]
```

$$\begin{bmatrix} 0.015708419435683 & 0 & 0 & -0.0078697388497913 & 0 \\ 0 & 0.0039426493427612 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0. & 0. \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -0.0078697388497913 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(8)}$$

$plot3d(f, x = 0..1, y = 0..1)$



### Let's begin analysis
###---------------------------------------------------------------------
$p := 2$ :
$rwin := 3$ :
$InterpolIDW(n+1, rwin, mink, p, R, IR)$ :

$n=, 101, rwin=, 3, mink=, mink, p=, 2$

Error, (in InterpolIDW) numeric exception: division by zero

$rwin := 5$ :
$InterpolIDW(n+1, rwin, mink, p, R, IR)$ :

$n=, 101, rwin=, 5, mink=, mink, p=, 2$ **(9)**

$ErrCheck(n, a, b, c, d, R)$

```
L2_Error=0.0247746
```

$rwin := 7$ :
$InterpolIDW(n+1, rwin, mink, p, R, IR)$ :

$n=, 101, rwin=, 7, mink=, mink, p=, 2$ **(10)**

$ErrCheck(n, a, b, c, d, R)$

```
L2_Error=0.0292606
```

$$\begin{bmatrix} \textit{101 x 101 Matrix} \\ \textit{Data Type: anything} \\ \textit{Storage: rectangular} \\ \textit{Order: Fortran\_order} \end{bmatrix} \quad \mathbf{(11)}$$

### **You should call your own "IterpolIDW2", below**
###-------------------------------------------------------------------------
$rwin := 2$ :
$mink := 4$ :
$InterpolIDW2Kim(n+1, rwin, mink, p, R, IR)$ :

$n=, 101, rwin=, 2, mink=, 4, p=, 2$ **(12)**

$ErrCheck(n, a, b, c, d, R)$

```
L2_Error=0.0244889
```