Apply RBF normalized MQ to reconstruct Surfaces using exterior auxiliary points

Anil Kumar Katwal
Department of Physics and Astronomy, The University of Southern Mississippi, Hattiesburg, Mississippi, 39406 USA[*]

I. INTRODUCTION

The normalized Multiquadric (MQ) Radial Basis Function (RBF) has significantly advanced surface reconstruction methodologies. Defined as

$$\phi(r) = \sqrt{1 + (cr)^2}$$

, where r represents the radial distance from the origin and c denotes a shape parameter, this function has found widespread application in approximating scattered data points and generating continuous surfaces. The normalization of the MQ RBF enhances its numerical stability, particularly in scenarios involving noisy or irregularly sampled data points. This property makes it invaluable in accurately reconstructing surfaces, even in challenging conditions. Moreover, the normalized MQ RBF excels in handling data points with varying densities, ensuring robust reconstruction across diverse datasets. Its adaptability to different shapes and the ability to control surface smoothness through the shape parameter render it a versatile tool in surface reconstruction tasks. Additionally, the MQ RBF's capability for interpolation and approximation, along with its reduced sensitivity to outliers, further solidify its position as a crucial component in various domains, including computer-aided design, medical imaging, and geological modeling.

II. PROBLEMS

Apply RBF normalized MQ to reconstruct the surfaces of the following cloud data points using exterior auxiliary points: for hand and forbunny

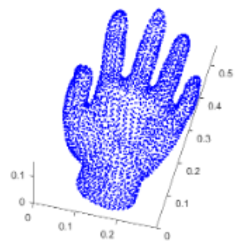
III. METHOD

This MATLAB script implemented a 3D Radial Basis Function (RBF) interpolation. It began by loading a dataset from a file, separating it into x, y, and z coordinates. The script defined a custom RBF function. Key constants and dataset size were set, and a set of interpolation centers was established. A vector was prepared for solving, and distances between centers were computed. The Influence Matrix (IM) was calculated based on these distances. Coefficients for interpolation were solved for. The script determined a spatial bounding box and created an evaluation grid. Distances between evaluation points and centers were calculated. The Evaluator Matrix (EM) was generated using the RBF function. The resulting scalar field was obtained by multiplying EM by the coefficients. An isosurface plot was generated, and its appearance, including color and lighting, was configured. The final plot displaying the isosurface in three dimensions was rendered. Some lines in the code were commented out, likely for testing or debugging purposes.

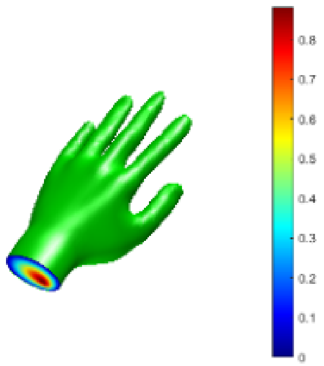
Listing 1: MATLAB Code for 3D hand Plot

```
clear;
data = load('hand2.txt');
x_data = data(:,1);
y_data = data(:,2);
z_data = data(:,3);

rbf = @(e,r) sqrt((e*r).^2+1);
c = 700;
N = size(data, 1);
neval = 50;
ctrs = [x_data y_data z_data];
ctrs = [ctrs;
    [0.133894,0.280227,0.111215];
    [0.226034,0.305576,0.16492];
    [0.404979,0.31825,0.119331];
    [0.58219,0.0774393,0.134727];
    [0.501432,-0.0239546,0.0346656];
    [0.114395,0.252269,0.112196];
    [0.318766,0.296531,0.16621];
    [0.538522,0.0752226,0.146095]];
rhs = [zeros(N,1); ones(8,1)];
DM_data = pdist2(ctrs, ctrs);
IM = sqrt((c * DM_data).^2 + 1);
coe = IM \ rhs;
bmin = min(ctrs, [], 1);
bmax = max(ctrs, [], 1);
xgrid = linspace(bmin(1)+0.02, bmax(1)+0.05, neval);
ygrid = linspace(bmin(2)-0.1, bmax(2)+0.1, neval);
zgrid = linspace(bmin(3)-0.05, bmax(3)+0.05, neval);
[xe, ye, ze] = meshgrid(xgrid, ygrid, zgrid);
epoints = [xe(:) ye(:) ze(:)];
block_size = 500;
% Determine the number of blocks
num_blocks = ceil(size(epoints, 1) / block_size);
pf = zeros(neval^3, 1);
for block = 1:num_blocks
    start_idx = (block - 1) * block_size + 1;
    end_idx = min(block * block_size, size(epoints, 1));
    current_epoints = epoints(start_idx:end_idx, :);
    DM_eval = pdist2(current_epoints, ctrs);
    EM = rbf(c, DM_eval);
    pf_block = EM * coe;
    pf(start_idx:end_idx) = pf(start_idx:end_idx) + pf_block;
end

pf = reshape(pf, neval, neval, neval);
% Generate isosurface
f = figure;
p = patch(isosurface(xe, ye, ze, pf, 0));
% p.FaceColor = 'green';
% p.EdgeColor = 'none';
set(p,'FaceLighting','gouraud','FaceColor','g','EdgeColor','nc
```

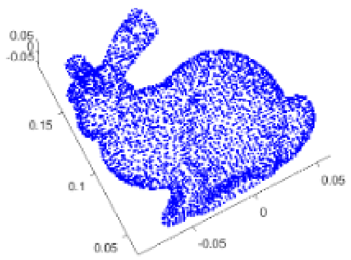
[*] Anil.Katwal@usm.edu



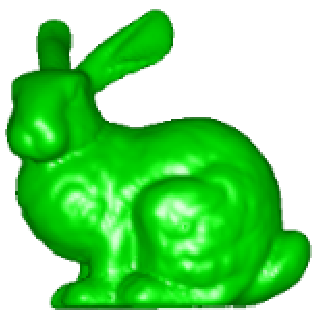
(a) Scatter plot of dataset of hand of given problem



(b) Surface Reconstruction using RBF of given problem



(c) Scatter Plot for Bunny using data cloud of given problem.

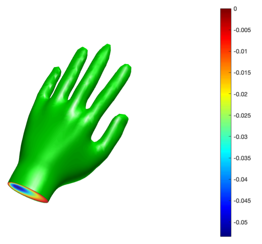


(d) Surface Reconstruction of Bunny using RBF of given problem.

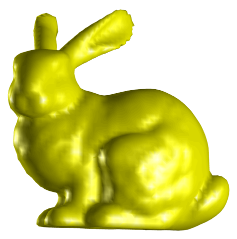
```

% Compute normals
isonormals(xe, ye, ze, pf, p);
isocaps(xe,ye,ze,pf,0,'below')
light('Position',[1 0 1],'Style','infinite');
light('Position',[-1 0 -1],'Style','infinite');

axis equal; colormap jet; colorbar;
view(3);
axis off;
```

Listing 2: MATLAB Code for 3D Bunny Plot

```
% Clear workspace
clear;
data = load('Data3D_Bunny2.mat');
dsites = data.dsites;
x_data = dsites(:,1);
y_data = dsites(:,2);
z_data = dsites(:,3);
rbf = @(e,r) sqrt((e*r).^2+1);
c = 700;
N = size(dsites, 1);
neval = 50;
% Exterior Point
data1 = load('external_buny.txt');
x0 = data1(:,1);
y0 = data1(:,2);
z0 = data1(:,3);
ctrs = [x_data y_data z_data];
ctrs = [ctrs;[x0,y0,z0]];
rhs = [zeros(N,1); ones(18,1)];
DM_data = pdist2(ctrs,ctrs);
IM = sqrt((c*DM_data).^2+1);
coe = IM \ rhs; %RBF coefficient
bmin = min(ctrs,[],1);
bmax = max(ctrs,[],1);
xgrid = linspace(bmin(1)-0.001, bmax(1)+0.001, neval);
ygrid = linspace(bmin(2)-0.001, bmax(2)+0.001, neval);
zgrid = linspace(bmin(3)-0.005, bmax(3)+0.005, neval);
[xe, ye, ze] = meshgrid(xgrid, ygrid, zgrid);
epoints = [xe(:) ye(:) ze(:)];
number = 100;
% Define block size
block_size = 500;
% Determine the number of blocks
num_blocks = ceil(size(epoints, 1) / block_size);
pf = zeros(neval^3, 1);
for block = 1:num_blocks
    start_idx = (block - 1) * block_size + 1;
    end_idx = min(block * block_size, size(epoints, 1));
    current_epoints = epoints(start_idx:end_idx, :);
    DM_eval = pdist2(current_epoints, ctrs);
    EM = rbf(c, DM_eval);
    pf_block = EM * coe;
    pf(start_idx:end_idx) = pf(start_idx:end_idx) + pf_block;
end
figure(1);
hold on;
plot3(ctrs(1:N,1), ctrs(1:N,2), ctrs(1:N,3), 'bo', 'Marker', '
plot3(data1(:,1), data1(:,2), data1(:,3), 'ro', 'Marker', 'o',
view(45, 30);
axis off;
pf = reshape(pf, neval, neval, neval);
%disp(pf)
% Generate isosurface
f = figure;
p = patch(isosurface(xe, ye, ze, pf, 0));
p.FaceColor = 'yellow';
p.EdgeColor = 'none';
```



(a) Surface Reconstruction using RBF of given dataset hand2.txt.

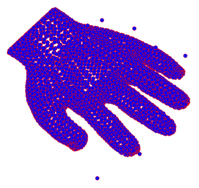


(b) Surface Reconstruction of Bunny using RBF given dataset.

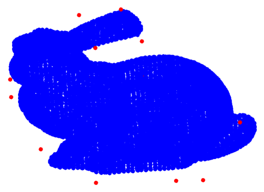
```
% Compute normals
isonormals(xe, ye, ze, pf, p);
set(p,'FaceLighting','gouraud','EdgeColor','none');
light('Position',[1 0 1],'Style','infinite');
light('Position',[-1 0 -1],'Style','infinite');
%axis equal; colormap jet; colorbar;
view(3);
axis equal;
```

.

IV. RESULTS



(a) Scatter plot of dataset of hand and dot points indicates the points taken for surface reconstructions



(b) Scatter plot and red points indicates the external points taken for surface reconstructions.