MATH 590: Meshfree Methods

Chapter 37: RBF Hermite Interpolation in MATLAB

Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

Fall 2010

Outline

1. Clustered Lagrange Interpolation vs. Hermite Interpolation

Outline

1. Clustered Lagrange Interpolation vs. Hermite Interpolation

We now illustrate the symmetric approach to Hermite interpolation with a set of numerical experiments for first-order Hermite interpolation (i.e., to positional and gradient data) in 2D using the MATLAB program `RBFHermite_2D.m` listed below.

We now illustrate the symmetric approach to Hermite interpolation with a set of numerical experiments for first-order Hermite interpolation (i.e., to positional and gradient data) in 2D using the MATLAB program `RBFHermite_2D.m` listed below.

Since derivatives of both the RBFs and the test function need to be included in the program we use the function

$$f(x, y) = \frac{\tanh(9(y - x)) + 1}{\tanh(9) + 1}$$

which has fairly simple partial derivatives (see lines 9–10 of the program) to generate the data.

We now illustrate the symmetric approach to Hermite interpolation with a set of numerical experiments for first-order Hermite interpolation (i.e., to positional and gradient data) in 2D using the MATLAB program `RBFHermite_2D.m` listed below.

Since derivatives of both the RBFs and the test function need to be included in the program we use the function

$$f(x,y) = \frac{\tanh(9(y-x))+1}{\tanh(9)+1}$$

which has fairly simple partial derivatives (see lines 9–10 of the program) to generate the data.

The RBF used in this set of experiments is the multiquadric with shape parameter $\varepsilon = 6$.

We compare four different problems:

1. Lagrange interpolation, i.e., interpolation to function values only, at a set of N equally spaced points in the unit square.

We compare four different problems:

1. Lagrange interpolation, i.e., interpolation to function values only, at a set of N equally spaced points in the unit square.
2. Lagrange interpolation to function values at $3N$ clustered points with separation distance $q = 0.1h$, where h is the fill distance of the set of equally spaced points (see the left plot below).

We compare four different problems:

1. Lagrange interpolation, i.e., interpolation to function values only, at a set of N equally spaced points in the unit square.
2. Lagrange interpolation to function values at $3N$ clustered points with separation distance $q = 0.1h$, where h is the fill distance of the set of equally spaced points (see the left plot below).
3. The same as above, but with $q = 0.01h$ (see the right plot below).

We compare four different problems:

1. Lagrange interpolation, i.e., interpolation to function values only, at a set of N equally spaced points in the unit square.
2. Lagrange interpolation to function values at $3N$ clustered points with separation distance $q = 0.1h$, where h is the fill distance of the set of equally spaced points (see the left plot below).
3. The same as above, but with $q = 0.01h$ (see the right plot below).
4. Hermite interpolation to function value, and values of both first-order partial derivatives at the N equally spaced points used in the first experiment.

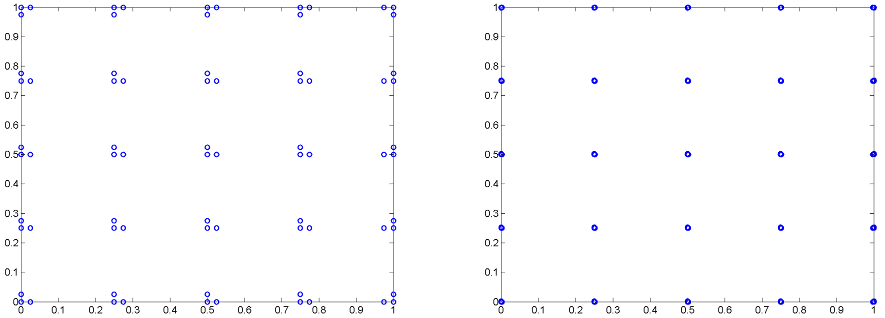

Figure: Clustered point sets with $N = 25$ basic data points. Cluster size $h/10$ (left) and cluster size $h/100$ (right).

The standard Lagrange interpolants were computed via a slightly modified `RBFInterpolation2D.m`.

The standard Lagrange interpolants were computed via a slightly modified `RBFInterpolation2D.m`.

Lagrange interpolation at clustered data sites was accomplished by the same program by adding the following lines to `RBFInterpolation2D.m` (see `RBFInterpolation2Dcluster.m`):

```
q = 0.1/(sqrt(N)-1);
grid = linspace(0,1,sqrt(N));
shifted = linspace(q,1+q,sqrt(N)); shifted(end) = 1-q;
[xc1,yc1] = meshgrid(shifted,grid);
[xc2,yc2] = meshgrid(grid,shifted);
dsites = [dsites; xc1(:) yc1(:); xc2(:) yc2(:)];
```

The program `RBFHermite_2D.m` maintains the same basic structure as earlier interpolation programs.

The program `RBFHermite_2D.m` maintains the same basic structure as earlier interpolation programs.

Now, however, we need to define derivatives of the RBF of up to twice the order of the data.

The program `RBFHermite_2D.m` maintains the same basic structure as earlier interpolation programs.

Now, however, we need to define derivatives of the RBF of up to twice the order of the data.

This is done for the MQ basic function on lines 1–6.

Program (RBFHermite_2D.m)

```
rbf = @(e,r) sqrt(1+(e*r).^2);  ep = 6;   % MQ RBF
dxrbf = @(e,r,dx) dx*e^2./sqrt(1+(e*r).^2);
dyrbf = @(e,r,dy) dy*e^2./sqrt(1+(e*r).^2);
dxxrbf = @(e,r,dx) e^2*(1+(e*r).^2-(e*dx).^2)./...
                       (1+(e*r).^2).^(3/2);
dxyrbf = @(e,r,dx,dy) -e^4*dx.*dy./(1+(e*r).^2).^(3/2);
dyyrbf = @(e,r,dy) e^2*(1+(e*r).^2-(e*dy).^2)./...
                       (1+(e*r).^2).^(3/2);
tf = @(x,y) (tanh(9*(y-x))+1)/(tanh(9)+1);
tfDx = @(x,y) 9*(tanh(9*(y-x)).^2-1)/(tanh(9)+1);
tfDy = @(x,y) 9*(1-tanh(9*(y-x)).^2)/(tanh(9)+1);
N = 289; dsites = CreatePoints(N,2,'u'); ctrs = dsites;
M = 1600;  epoints = CreatePoints(M,2,'u');
DM_eval = DistanceMatrix(epoints,ctrs);
dx_eval = DifferenceMatrix(epoints(:,1),ctrs(:,1));
dy_eval = Differencematrix(epoints(:,2),ctrs(:,2));
DM_data = DistanceMatrix(dsites,ctrs);
dx_data = DifferenceMatrix(dsites(:,1),ctrs(:,1));
dy_data = DifferenceMatrix(dsites(:,2),ctrs(:,2));
```

Program (RBFHermite_2D.m (cont.))

```
rhs = [tf(dsites(:,1),dsites(:,2)); ...
        tfDx(dsites(:,1),dsites(:,2)); ...
        tfDy(dsites(:,1),dsites(:,2))];
exact = tf(epoints(:,1),epoints(:,2));
IM = rbf(ep,DM_data);
DxIM = dxrbf(ep,DM_data,dx_data);
DyIM = dyrbf(ep,DM_data,dy_data);
DxxIM = dxxrbf(ep,DM_data,dx_data);
DxyIM = dxyrbf(ep,DM_data,dx_data,dy_data);
DyyIM = dyyrbf(ep,DM_data,dy_data);
IM = [IM -DxIM -DyIM;
      DxIM -DxxIM -DxyIM;
      DyIM -DxyIM -DyyIM];
EM = rbf(ep,DM_eval);
DxEM = dxrbf(ep,DM_eval,dx_eval);
DyEM = dyrbf(ep,DM_eval,dy_eval);
EM = [EM -DxEM -DyEM];
Pf = EM * (IM\rhs);
maxerr = norm(Pf-exact,inf)
```

Since the derivatives of the basic function now also contain difference terms we need another subroutine that computes matrices of differences of point coordinates.

Since the derivatives of the basic function now also contain difference terms we need another subroutine that computes matrices of differences of point coordinates.

Program (`DifferenceMatrix.m`)

```
function DM = DifferenceMatrix(datacoord,centercoord)
[dr,cc] = ndgrid(datacoord(:),centercoord(:));
DM = dr-cc;
```

Since the derivatives of the basic function now also contain difference terms we need another subroutine that computes matrices of differences of point coordinates.

Program (`DifferenceMatrix.m`)

```
function DM = DifferenceMatrix(datacoord,centercoord)
[dr,cc] = ndgrid(datacoord(:),centercoord(:));
DM = dr-cc;
```

Remark

This code is used in the block matrices `IM` *and* `EM` *in* `RBFHermite_2D.m`*. The minus signs used in columns 2 and 3 of the block matrices reflect differentiation of the basic function with respect to its second variable.*

Mesh	Lagrange		Clustered, $q = 0.1h$	
	RMS-error	cond(A)	RMS-error	cond(A)
3×3	1.620492e-001	6.078349e+001	8.471301e-002	9.052247e+003
5×5	6.148258e-002	9.464176e+002	2.733258e-002	3.073957e+005
9×9	8.521994e-003	6.523036e+004	2.678543e-003	8.811980e+007
17×17	2.246810e-004	9.017750e+007	3.138761e-005	3.555214e+012
33×33	2.017643e-006	4.799960e+013	2.925784e-007	6.474324e+020

Table: 2D interpolation with clustered data vs. Hermite interpolation (part 1).

Mesh	Clustered, $q = 0.01h$		Hermite	
	RMS-error	cond(A)	RMS-error	cond(A)
3×3	9.084939e-002	8.580483e+005	9.128193e-002	1.326346e+002
5×5	2.792157e-002	2.829762e+007	2.794943e-002	2.292450e+003
9×9	2.687753e-003	8.325283e+009	2.688346e-003	2.185224e+005
17×17	3.147808e-005	3.426489e+014	3.148843e-005	2.486624e+009
33×33	8.941613e-006	8.943758e+020	5.731027e-009	6.261336e+018

Table: 2D interpolation with clustered data vs. Hermite interpolation (part 2).

Remark

- *We can see that the limit of the clustered Lagrange interpolants as $q \to 0$ behaves like the Hermite interpolants.*

Remark

- *We can see that the limit of the clustered Lagrange interpolants as $q \to 0$ behaves like the Hermite interpolants.*
- *Interpolation to function and derivative data at a given point is more accurate than interpolation to function values alone.*

Remark

- *We can see that the limit of the clustered Lagrange interpolants as $q \to 0$ behaves like the Hermite interpolants.*
- *Interpolation to function and derivative data at a given point is more accurate than interpolation to function values alone.*
- *The advantage of the Hermite interpolation approach over the clustered Lagrange approach is clearly evident for the experiments with $N = 33 \times 33 = 1089$ basic data points (or $N = 3267$ clustered data points).*

Remark

- *We can see that the limit of the clustered Lagrange interpolants as $q \to 0$ behaves like the Hermite interpolants.*
- *Interpolation to function and derivative data at a given point is more accurate than interpolation to function values alone.*
- *The advantage of the Hermite interpolation approach over the clustered Lagrange approach is clearly evident for the experiments with $N = 33 \times 33 = 1089$ basic data points (or $N = 3267$ clustered data points).*
  - *The ℓ_2-condition number of A for the clustered interpolants is on the order of 10^{20}, while it is "only" 6.261336e+018 for the Hermite matrix.*

Remark

- *We can see that the limit of the clustered Lagrange interpolants as $q \to 0$ behaves like the Hermite interpolants.*
- *Interpolation to function and derivative data at a given point is more accurate than interpolation to function values alone.*
- *The advantage of the Hermite interpolation approach over the clustered Lagrange approach is clearly evident for the experiments with $N = 33 \times 33 = 1089$ basic data points (or $N = 3267$ clustered data points).*
    - *The ℓ_2-condition number of A for the clustered interpolants is on the order of 10^{20}, while it is "only" 6.261336e+018 for the Hermite matrix.*
    - *This difference has a significant impact on the numerical stability, and the resulting RMS-errors.*

Remark

- *We can see that the limit of the clustered Lagrange interpolants as $q \to 0$ behaves like the Hermite interpolants.*
- *Interpolation to function and derivative data at a given point is more accurate than interpolation to function values alone.*
- *The advantage of the Hermite interpolation approach over the clustered Lagrange approach is clearly evident for the experiments with $N = 33 \times 33 = 1089$ basic data points (or $N = 3267$ clustered data points).*
  - *The ℓ_2-condition number of A for the clustered interpolants is on the order of 10^{20}, while it is "only" 6.261336e+018 for the Hermite matrix.*
  - *This difference has a significant impact on the numerical stability, and the resulting RMS-errors.*
  - *The Hermite interpolant is more than three orders of magnitude more accurate than the Lagrange interpolant to clusters with $q = h/100$.*

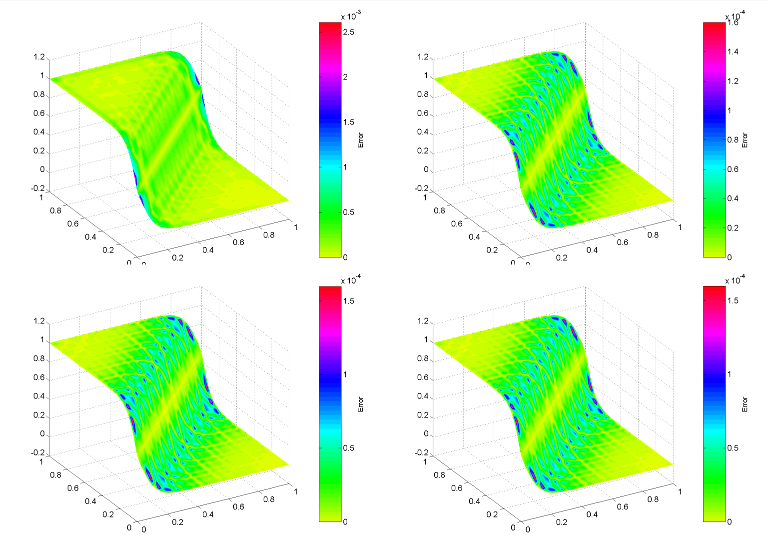

Figure: Fits for clustered interpolants with $N = 289$ basic data points. Top left to bottom right: Lagrange interpolant, interpolant with cluster size $h/10$, interpolant with cluster size $h/100$, Hermite interpolant.

References I

Buhmann, M. D. (2003).
Radial Basis Functions: Theory and Implementations.
Cambridge University Press.

Fasshauer, G. E. (2007).
Meshfree Approximation Methods with MATLAB.
World Scientific Publishers.

Higham, D. J. and Higham, N. J. (2005).
MATLAB *Guide*.
SIAM (2nd ed.), Philadelphia.

Iske, A. (2004).
Multiresolution Methods in Scattered Data Modelling.
Lecture Notes in Computational Science and Engineering 37, Springer Verlag (Berlin).

References II

G. Wahba (1990).
Spline Models for Observational Data.
CBMS-NSF Regional Conference Series in Applied Mathematics 59, SIAM (Philadelphia).

Wendland, H. (2005a).
Scattered Data Approximation.
Cambridge University Press (Cambridge).