Esercitazione

Teoria dell'Approssimazione e Applicazioni

7 maggio 2015

MATLAB codes free downloadable at:

http://hdl.handle.net/2318/158790

R. CAVORETTO, A. DE ROSSI, E. PERRACCHIONE, *Fast computation of partition of unity interpolants through block-based data structures*, submitted (2015).

`PUM_2D_CSRBF.m` `PUM_3D_CSRBF.m`	scripts performing the partition of unity using CSRBFs
`BlockBased2D_Structure.m` `BlockBased3D_Structure.m`	scripts that store points into the different neighbourhoods
`BlockBased2D_ContainingQuery.m` `BlockBased3D_ContainingQuery.m`	scripts performing the containing query procedure
`BlockBased2D_RangeSearch.m` `BlockBased3D_RangeSearch.m`	scripts that perform the range search procedure
`BlockBased2D_DistanceMatrix.m` `BlockBased3D_DistanceMatrix.m`	scripts that form the distance matrix of two sets of points for CSRBFs
`inhull.m`	script that tests if a point belongs to the convex hull
`countingsort.m`	script that performs a sorting routine for integers
`haltonseq.m`	script that generates Halton data

Table 1: The MATLAB codes for the block-based partition of unity algorithms.

Remark: `inhull.m`, `countingsort.m` and `haltonseq.m` are available online at:

`http://www.mathworks.com/matlabcentral/fileexchange/`.

PUM_2D_CSRBF

```
%-------------------------------------------------------------------------%
%
% File: PUM_2D_CSRBF(dsites,rbf,wf,ep,testfunction,neval,npu)
%
% Goal: script that performs partition of unity interpolation using
%       compactly supported RBFs
%
% Inputs:  dsites:       NX2 matrix representing a set of N data sites
%          rbf:          radial basis function. The CSRBF and the weight
%                        function used with this code must be given
%                        in shifted form rbf2(u) = rbf(r), u = 1-e*r.
%                        For example, the Wendland C2
%                        rbf = @(e,r) max(1-e*r,0).^4.*(4*e*r+1);
%                        becomes rbf2 @(e,r) r.^4.*(5*spones(r)-4*r)
%          wf:           weight function
%          ep:           shape parameter of RBF
%          testfunction: test function
%          neval:        number of evaluation points in one direction
%          npu:          number of PU subdomains in one direction
%
% Outputs:  epoints: the evaluation points
%           Pf:      the interpolant computed at the evaluation points
%
%-------------------------------------------------------------------------%
```

BlockBased2D_Structure

```
%-------------------------------------------------------------------------%
%
% File: BlockBased2D_Structure(dsites,q,puradius,min_dsites)
%
% Goal: find the data sites located in each of the q^2 cells and in the
%       neighbouring cells
%
% Inputs: dsites:     NX2 matrix representing a set of N data sites
%         q:          number of cells in one direction
%         puradius:   radius of PU subdomains
%         min_dsites: minimum of data sites among two directions
%
% Calls on: countingsort: by B. Moore, from MATLAB Central File Exchange
%
% Outputs: idx_dsites: multiarray containing the indices of the data points
%                      located in k-th block and in the neighbouring blocks
%
%-------------------------------------------------------------------------%
```

BlockBased2D_ContainingQuery

```
%-------------------------------------------------------------------------%
%
% File: BlockBased2D_ContainingQuery(puctr,q,puradius,min_dsites)
%
% Goal: script that given a subdomain centre returns the index of
%       the square cell containing the subdomain centre
%
% Inputs: puctr:       subdomain centre
%         q:           number of square cells in one direction
%         puradius:    radius of PU subdomains
%         min_dsites:  minumim of data sites among two directions
%
% Outputs: index: the index of the square cell containing the subdomain
%                 centre
%
%-------------------------------------------------------------------------%
```

BlockBased2D_RangeSearch

```
%-------------------------------------------------------------------------%
%
% File: BlockBased2D_RangeSearch(puctr,puradius,dsites,index)
%
% Goal: find the data sites located in a given subdomain and the distances
%       between the subdomain centre and data sites
%
% Inputs: puctr:     subdomain centre
%         puradius:  radius of PU subdomains
%         dsites:    NX2 matrix representing a set of N data sites
%         index:     vector containing the indices of the data points
%                    located in the k-th block (the cell containing the
%                    subdomain centre) and in the neighbouring cells
%
% Outputs: idx:  vector containing the indices of the data points located
%                in a given PU subdomain
%          dist: vector containing the distances between the data sites
%                and the subdomain centre
%
%-------------------------------------------------------------------------%
```

ESERCIZIO

Usando il software MATLAB implementato per il metodo di partizione dell'unità, interpolare un insieme di dati sparsi (Halton) contenuti all'interno di un dominio pentagonale, i cui vertici sono $(0.25, 0)$, $(0.75, 0)$, $(1, 0.5)$, $(0.5, 1)$ e $(0, 0.5)$. Come funzione peso e RBF locale utilizzare la funzione di Wendland C^2, mentre come funzione test usare la funzione di Franke.

Nota: il numero di sottodomini lungo una direzione è definito da $\texttt{npu} = \left\lfloor \frac{1}{2} l_{box} \left(\frac{N}{A_K} \right)^{1/2} \right\rfloor$, dove

- $l_{box} = \max_{dsites} - \min_{dsites}$;
- $A_K =$ area dell'inviluppo convesso.

Soluzione.

```
clear all

% Define the convex hull (not input)
X = [0.25 0; 0.75 0; 1 0.5; 0.5 1; 0 0.5];

% Define the number of Halton data in the unit square
N = 4000;

% Compute Halton data
dsites = haltonseq(N,2);
K = convhulln(X);

% Reduce data in taking only those contained in the convex hull
in_dsites = inhull(dsites,X,K);
dsites = dsites(in_dsites,:);

plot(dsites(:,1),dsites(:,2),'.');
axis square

N = size(dsites,1);

% Compute the ratio between areas to define the number of subdomains
[K,area] = convhulln(dsites); % Compute the convex hull and the area
max1 = max(dsites(:,1)); min1 = min(dsites(:,1));
max2 = max(dsites(:,2)); min2 = min(dsites(:,2));

max_dsites = max(max1,max2); min_dsites = min(min1,min2);

ratio = (max_dsites-min_dsites).^2./area;

% Inputs
ep = 0.5; % Shape parameter
neval = 40; % Parameter for neval-by-neval grid of evaluation points
wf = @(e,r) r.^4.*(5*spones(r)-4*r); % Define weight function
rbf = @(e,r) r.^4.*(5*spones(r)-4*r); % Define RBF function

% Define the testfunction
f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1)/10));
f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);

% Parameter for npu-by-npu grid of PU subdomains
npu = floor((N./(4))^(1/2)*sqrt(ratio));
PUM_2D_CSRBF(dsites,rbf,wf,ep,testfunction,neval,npu);
```