FUNZIONI MATLAB© PER LE WAVELET INTERPOLANTI

SILVIA BERTOLUZZA

1. Framework

Le wavelet interpolanti simmetriche hanno l'equazione di dilatazione che può essere scritta nella forma seguente:

$$\theta(x) = \theta(2x) + \sum_{k=1}^{N} a_k(\theta(2x-(2k-1)) + \theta(2x+(2k-1))).$$

Sono costruite come autocorrelazione delle funzioni scala φ di Daubechies. I coefficienti a_k verificano la relazione:

2. Strutture dati

`base.` È una struttura che include tutti i dati che servono per descrivere e manipolare una base di ondine interpolanti:

- `base.a`: il filtro che compare nell'equazione di dilatazione per la funzione θ;
- `base.J`: il livello di raffinamento usato per costruire i vettori dei valori della funzione theta e delle sue derivate;
- `base.L`: $\mathrm{supp}(\theta) = [-L, L]$;
- `base.P`: grado dei polinomi riproducibili esattamente
- `base.J0`: $J_0 = \lceil \log_2(L) \rceil$ è il livello minimo richiesto dalle modifiche di bordo
- `base.THETA`: matrice le cui colonne contengono i valori nei punti diadici della funzione θ e delle sue derivate prima e seconda.

`grid.` È una struttura dati che descrive un'insieme di punti diadici. In dimensione due abbiamo

- `grid.j0`: livello coarse della griglia (scalare)
- `grid.npoints`: numero di punti
- `grid.j`: $(x, y) = 2^{-j}(k_x, k_y) \to j$
- `grid.kx`: $(x, y) = 2^{-j}(k_x, k_y) \to k_x$
- `grid.ky`: $(x, y) = 2^{-j}(k_x, k_y) \to k_y$
- `grid.x`: $(x, y) = 2^{-j}(k_x, k_y) \to x$
- `grid.y`: $(x, y) = 2^{-j}(k_x, k_y) \to y$
- `grid.b`: 1 if the point is on the boundary, 0 otherwise. NB. It could be used to distinguish different boundary portions.

Remark 2.1. (Nota × me) non c'è per ora il tipo, perché ho usato solo le funzioni di tipo 2)

Remark 2.2. a meno che $j = j_0$, la coppia (k_x, k_y) non può essere formata da due numeri pari.

Remark 2.3. Grazie alla corrispondenza tra punti e funzioni, la struttura `grid` viene usata anche per rappresentare lo spazio discreto generato dalle funzioni corrispondenti ai punti.

3. Funzioni

`start.` Dato `nw` dispari (indicativo della wavelet di Daubechies di partenza) ed un livello massimo `J` costruisce una struttura `base`. Uso:

```
>> base = start(nw,J)
```

Remark 3.1. `nw` $\geq 3 \Rightarrow \theta \in C^2$, `nw` $\geq 1 \Rightarrow \theta \in C^0$

`uniform2d.` Costruisce una struttura di tipo `grid` corrispondente ad una griglia uniforme di passo $2^{-j_{max}}$. La griglia è organizzata in scale multiple, con livello coarse $= 2^{-J_0}$. Uso:

```
>> grid = uniform2d(j0,jmax)
```

Remark 3.2. di solito si usa $j_0 =$ `base.J0`.

`operator2d.` Costruisce la matrice di rigidita' ottenuta tramite metodo di collocazione usando lo spazio generato dalle funzioni di `space` (struttura di tipo grid) e collocando nei punti della griglia `grid` per un operatore di tipo

$$Au = \sum_{i=0}^{2} \sum_{j=0}^{2-i} c_{i,j} \partial_x^i \partial_y^j u$$

I coefficienti $c_{i,j}$ sono passati tramite una matrice 3×3 `coefs`: `coefs(i+1,j+1)`$= c_{i,j}$. Uso:

```
>> S = operator2d(space,grid,coefs,base)
```

Remark 3.3. Esempi di matrici per operatori particolari:

$$u \leftrightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$-\triangle u \leftrightarrow \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

$$\beta_x u_x + \beta_y u_y \leftrightarrow \begin{pmatrix} 0 & \beta_x & 0 \\ \beta_y & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

`appendbc2d.` Data una matrice di rigidità `S` e un vettore right hand side `f` costruisce una matrice di rigidità `Sb` ed un vettore `fb` modificati in modo che la soluzione del sistema lineare `Sb y = fb` verifich condizioni al contorno di Dirichlet omogenee. Uso:

```
>> [stiffb,fb] = appendbc2d(stiff,mass,f,grid,truesol)
```

`l2_error`. Dati i coefficienti `u` di una soluzione approssimata e una stringa che individui una function che calcola la soluzione esatta, valuta la norma L^2 della differenza.

```
>> [errl2] = l2_error(u,space,base,truesol);
```

Remark 3.4. Esempio di una funzione soluzione esatta (che deve poter ammettere input multipli)

```
function z=soluzione(x,y);
z = zeros(size(x));
% z=sin(x)+cos(y);
```

`adapt`. Data una griglia `grid` e i coefficienti `u` di una funzione definita sulla griglia, e due tolleranze `tolr` e `tolc`, costruisce una nuova griglia `newgrid` e l'interpolata `newu` nello spazio corrispondente. Uso

```
>> [newgrid,newu] = adapt(grid,u,tolr,tolc)
```

`plot2d`. Dati i valori `z` di una funzione nei punti della griglia `grid`, plotta la funzione usando una griglia di plottaggio uniforme 10×10. Uso:

```
>> plot2d(z,grid)
```

Remark 3.5. Dati i coefficienti della funzione sulla griglia `grid`, i valori nei punti si ottengono moltiplicando per la matrice di massa (costruita con `operator2d`).

Funzioni ausiliarie. . Le funzioni `buildcol`, `convolution`, `extrapolate`, `filtro`, `refine`, `restriction`, `valori` e `wavelet` sono funzioni ausiliarie che non vengono mai chiamate direttamente.

Utilities. . Le funzioni `soluzione`, `rhs`, `uno`, e `zero` sono funzioni che prendono in input vettori `x` e `y` e ritornano `z`. Possono essere usate come right hand side, valore al bordo o soluzione vera. Ad esempio per costruire il right hand side si fa

```
>> f = rhs(grid.x,grid.y)
```

E-mail address: silvia.bertoluzza@imati.cnr.it