Algorithms for local refinement in hierarchical spline spaces

Eduardo M. Garau[1,2,3] and Rafael Vázquez[1]

[1]Istituto di Matematica Applicata e Tecnologie Informatiche 'E. Magenes' (CNR), Italy
[2]Instituto de Matemática Aplicada del Litoral (CONICET-UNL), Argentina
[3]Facultad de Ingeniería Química (UNL), Argentina

October 23, 2015

1 Setting

Let $d \geq 1$. We are going to consider tensor-product d-variate spline function spaces on $\Omega := [0,1]^d \subset \mathbb{R}^d$, where $\mathbf{p} := (p_1, p_2, \ldots, p_d)$ denotes the vector of polynomial degrees of the splines with respect to each coordinate direction.

1.1 Underlying sequence of tensor-product spline spaces

We consider a given sequence $\{\mathcal{S}_\ell\}_{n\in\mathbb{N}_0}$ of tensor-product d-variate spline spaces such that

$$\mathcal{S}_0 \subset \mathcal{S}_1 \subset \mathcal{S}_2 \subset \mathcal{S}_3 \subset \ldots, \tag{1}$$

with the corresponding tensor-product B-spline bases denoted by

$$\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \ldots, \tag{2}$$

respectively. Furthermore, for $\ell \in \mathbb{N}_0$, we denote by $\mathcal{Q}_\ell$ the tensor-product mesh associated to $\mathcal{B}_\ell$ and we say that $Q \in \mathcal{Q}_\ell$ is a *cell of level* ℓ. We now state some well-known properties of the B-spline basis functions [dB01, S07]:

- *Local linear independence.* For any nonempty open set $O \subset \Omega$, the functions in $\mathcal{B}_\ell$ that do not vanish identically on O, are linearly independent on O.

- *Positive partition of unity.* The B-spline basis functions of level ℓ form a partition of the unity on Ω, i.e.,
$$\sum_{\beta\in\mathcal{B}_\ell} \beta \equiv 1, \qquad \text{on } \Omega. \tag{3}$$

- *Two-scale relation between consecutive levels.* The B-splines of level ℓ can be written as a linear combination of B-splines of level $\ell+1$. More precisely,

$$\beta_\ell = \sum_{\beta_{\ell+1} \in \mathcal{C}(\beta_\ell)} c_{\beta_{\ell+1}}(\beta_\ell)\beta_{\ell+1}, \qquad \forall\, \beta_\ell \in \mathcal{B}_\ell, \tag{4}$$

  where the coefficients $c_{\beta_{\ell+1}}(\beta_\ell)$ are strictly positive, and $\mathcal{C}(\beta_\ell) \subset \mathcal{B}_{\ell+1}$ is the set of children of β_ℓ as defined in [BG15a].

Remark 1.1. Notice that if we define $c_{\beta_{\ell+1}}(\beta_\ell) := 0$ when $\beta_{\ell+1}$ is not a child of β_ℓ, then equation (4) can be written as

$$\beta_\ell = \sum_{\beta_{\ell+1} \in \mathcal{B}_{\ell+1}} c_{\beta_{\ell+1}}(\beta_\ell)\beta_{\ell+1}, \qquad \forall\, \beta_\ell \in \mathcal{B}_\ell. \tag{5}$$

In particular, we remark that

$$\mathcal{C}(\beta_\ell) = \{\beta_{\ell+1} \in \mathcal{B}_{\ell+1} \,|\, c_{\beta_{\ell+1}}(\beta_\ell) > 0\} \subset \{\beta_{\ell+1} \in \mathcal{B}_{\ell+1} \,|\, \operatorname{supp}\beta_{\ell+1} \subset \operatorname{supp}\beta_\ell\}. \tag{6}$$

1.2 Hierarchical B-spline basis and hierarchical spline space

Definition 1.2. If $n \in \mathbb{N}$, we say that $\mathbf{\Omega}_n := \{\Omega_0, \Omega_1, \dots, \Omega_n\}$ is a *hierarchy of subdomains of* Ω *of depth* n if

(i) Ω_ℓ is the union of cells of level $\ell - 1$, for $\ell = 1, 2, \dots, n$.

(ii) $\Omega = \Omega_0 \supset \Omega_1 \supset \cdots \supset \Omega_{n-1} \supset \Omega_n = \emptyset$.

We now define the hierarchical B-spline basis $\mathcal{H} = \mathcal{H}(\mathbf{\Omega}_n)$ by

$$\mathcal{H} = \bigcup_{\ell=0}^{n-1} \{\beta \in \mathcal{B}_\ell \mid \operatorname{supp}\beta \subset \Omega_\ell \wedge \operatorname{supp}\beta \not\subset \Omega_{\ell+1}\}. \tag{7}$$

We say that β is *active* if $\beta \in \mathcal{H}$. The corresponding underlying mesh $\mathcal{Q} = \mathcal{Q}(\mathbf{\Omega}_n)$ is given by

$$\mathcal{Q} := \bigcup_{\ell=0}^{n-1} \{Q \in \mathcal{Q}_\ell \;|\; Q \subset \Omega_\ell \,\wedge\, Q \not\subset \Omega_{\ell+1}\}, \tag{8}$$

and we say that Q is an *active cell* is $Q \in \mathcal{Q}$, or that Q is an *active cell of level* ℓ if $Q \in \mathcal{Q} \cap \mathcal{Q}_\ell$.

Unlike the B-spline bases $\mathcal{B}_\ell$ for tensor-product spline spaces, the hierarchical B-spline basis $\mathcal{H}$ does not constitute a partition of the unity. Instead, in view of the linear independence of functions in $\mathcal{H}$ and taking into account that $\mathcal{S}_0 = \operatorname{span}\mathcal{B}_0 \subset \operatorname{span}\mathcal{H}$, we have that there exists a set $\{a_\beta\}_{\beta\in\mathcal{H}} \subset \mathbb{R}$, uniquely determined, such that

$$\sum_{\beta \in \mathcal{H}} a_\beta \beta \equiv 1, \qquad \text{on } \Omega. \tag{9}$$

It can be proved that $a_\beta \geq 0$, for all $\beta \in \mathcal{H}$. On the other hand, we remark that these coefficients depend on the hierarchy of subdomains $\mathbf{\Omega}_n$.

2 Refinement of hierarchical spline spaces

In this section, we present a precise technique to refine locally a given hierarchical spline space $\operatorname{span}\mathcal{H}$.

Definition 2.1. Let $\mathbf{\Omega}_n := \{\Omega_0, \Omega_1, \ldots, \Omega_n\}$ and $\mathbf{\Omega}^*_{n+1} := \{\Omega^*_0, \Omega^*_1, \ldots, \Omega^*_n, \Omega^*_{n+1}\}$ be hierarchies of subdomains of Ω of depth (at most) n and $n+1$, respectively. We say that $\mathbf{\Omega}^*_{n+1}$ is an *enlargement* of $\mathbf{\Omega}_n$ if

$$\Omega_\ell \subset \Omega^*_\ell, \qquad \ell = 1, 2, \ldots, n.$$

Let $\mathcal{H}$ and $\mathcal{Q}$ be the hierarchical B-spline basis and the hierarchical mesh associated to the hierarchy of subdomains of depth n, $\mathbf{\Omega}_n := \{\Omega_0, \Omega_1, \ldots, \Omega_n\}$.

Let $\mathbf{\Omega}^*_{n+1}$ be an enlargement of $\mathbf{\Omega}_n$. Now, the corresponding hierarchical B-spline basis $\mathcal{H}^*$ and *refined* mesh $\mathcal{Q}^*$ are given by

$$\mathcal{H}^* := \bigcup_{\ell=0}^{n} \{\beta \in \mathcal{B}_\ell \mid \operatorname{supp}\beta \subset \Omega^*_\ell \wedge \operatorname{supp}\beta \not\subset \Omega^*_{\ell+1}\}, \tag{10}$$

and

$$\mathcal{Q}^* := \bigcup_{\ell=0}^{n} \{Q \in \mathcal{Q}_\ell \mid Q \subset \Omega^*_\ell \wedge Q \not\subset \Omega^*_{\ell+1}\}.$$

Let $\{a^*_\beta\}_{\beta\in\mathcal{H}^*}$ denote the sequence of coefficients (with respect to the hierarchy $\mathbf{\Omega}^*_{n+1}$) such that

$$\sum_{\beta\in\mathcal{H}^*} a^*_\beta \beta \equiv 1, \qquad \text{on } \Omega.$$

In [GJS14] has been proved that any enlargement of $\mathbf{\Omega}_n$ gives rise to a new enriched hierarchical B-spline basis $\mathcal{H}^*$, in the sense that

$$\operatorname{span}\mathcal{H} \subset \operatorname{span}\mathcal{H}^*.$$

In order to enlarge the given subdomains $\mathbf{\Omega}_n = \{\Omega_0, \Omega_1, \ldots, \Omega_n\}$ we have to select the areas in Ω where more ability of approximation is required. Such a choice can be done by selecting to *refine* some active basis functions or some active cells. More precisely, we consider the two following ways of enlarging the hierarchy $\mathbf{\Omega}_n$:

- **Marking basis functions:** We consider a subset $\mathcal{M}$ of active B-spline basis functions, i.e., $\mathcal{M} \subset \mathcal{H}$. Let $\mathcal{M}_\ell := \mathcal{M} \cap \mathcal{B}_\ell$, for $\ell = 0, 1, \ldots, n-1$. Now, we define the hierarchy of domains $\mathbf{\Omega}^*_{n+1} := \{\Omega^*_0, \Omega^*_1, \ldots, \Omega^*_n, \Omega^*_{n+1}\}$ of depth (at most) $n+1$, by

$$\begin{cases} \Omega^*_0 & := \Omega_0, \\ \Omega^*_\ell & := \Omega_\ell \cup \displaystyle\bigcup_{\beta\in\mathcal{M}_{\ell-1}} \operatorname{supp}\beta, \qquad \ell = 1, 2, \ldots, n, \\ \Omega^*_{n+1} & := \emptyset. \end{cases} \tag{11}$$

  Let $\mathcal{H}^*$ be the hierarchical B-spline basis associated to $\mathbf{\Omega}^*_{n+1}$. Notice that $\mathcal{M} \subset \mathcal{H} \setminus \mathcal{H}^*$, i.e., at least the functions in $\mathcal{M}$ have been removed (or deactivated) from the hierarchical basis $\mathcal{H}$.

- **Marking active cells:** We consider a subset $\mathcal{M}$ of active cells, i.e., $\mathcal{M} \subset \mathcal{Q}$. Let $\mathcal{M}_\ell := \mathcal{M} \cap \mathcal{Q}_\ell$, for $\ell = 0, 1, \ldots, n-1$. Now, we define the hierarchy of domains $\mathbf{\Omega}^*_{n+1} := \{\Omega^*_0, \Omega^*_1, \ldots, \Omega^*_n, \Omega^*_{n+1}\}$ of depth (at most) $n+1$, by

$$\begin{cases} \Omega^*_0 & := \Omega_0, \\ \Omega^*_\ell & := \Omega_\ell \cup \bigcup\limits_{Q \in \mathcal{M}_{\ell-1}} Q, \qquad \ell = 1, 2, \ldots, n, \\ \Omega^*_{n+1} & := \emptyset. \end{cases} \tag{12}$$

  Let $\mathcal{H}^*$ be the hierarchical B-spline basis associated to $\mathbf{\Omega}^*_{n+1}$ and $\mathcal{Q}^*$ be the corresponding hierarchical mesh. In this case, $\mathcal{M} \subset \mathcal{Q} \setminus \mathcal{Q}^*$, i.e., all cells in $\mathcal{M}$ have been refined.

3 Algorithms for initialization and refinement of a hierarchical B-spline basis

Here we describe an algorithm to compute the active B-splines in the finer basis taking advantage of the knowledge of the active B-splines in the current coarse basis.

We consider a global numbering for all the basis functions in $\mathcal{B}_\ell$, for each $\ell \in \mathbb{N}_0$, and assume that we have available the following basic routines related with the underlying tensor-product spline spaces:

(1) Basic routines in each tensor-product space $\mathcal{S}_\ell$:

- $I =$ `get_cells`(i_β, ℓ), where i_β is the global index of a function $\beta \in \mathcal{B}_\ell$, and I contains the global indices of the cells in $\mathcal{Q}_\ell$ which are subsets of $\operatorname{supp} \beta$.
- $I =$ `get_neighbors`(i_β, ℓ), where i_β is the global index of a function $\beta \in \mathcal{B}_\ell$, and I contains the global indices of functions in $\mathcal{B}_\ell$ whose supports have at least one cell of level ℓ within $\operatorname{supp} \beta$.
- $I =$ `get_basis_functions`(i_Q, ℓ), where i_Q is the global index of a cell $Q \in \mathcal{Q}_\ell$ and I contains the global indices of functions in $\mathcal{B}_\ell$ that do not vanish on Q.

(2) Basic routines linking two consecutive levels of the tensor-product spaces ($\mathcal{S}_\ell$ and $\mathcal{S}_{\ell+1}$):

- $I =$ `split_cell`(i_Q, ℓ), where i_Q is the global index of a cell $Q \in \mathcal{Q}_\ell$ and I contains the global indices of the cells in $\mathcal{Q}_{\ell+1}$ which are inside of Q.
- $[I, c] =$ `split_fun`(i_β, ℓ), where i_β is the global index of a function $\beta \in \mathcal{B}_\ell$, I contains the global indices of all children of β, and c is an array with the corresponding coefficients given by (4).

Remark 3.1. The routines listed above are in fact elementary. The *hardest* thing in the previous routines is the computation of the coefficients for the two-scale relation (4) in the function `split_fun`. Nevertheless, by virtue of the tensor-product structure of B-splines, this task can be done by computing the corresponding coefficients in the two-scale relation for univariate B-splines and Kronecker products. On the other hand, it is important to remark that the coefficients in the univariate case can be computed using knot insertion formulae. It is important to mention that these coefficients will be used in our algorithms below to compute the coefficients of the hierarchical basis functions for the partition of the unity (cf. (9)). We remark that some a posteriori error estimators may require this information [BG15b].

Finally, we remark that if we do not need the explicit knowledge of the coefficients in (9), we can consider a simple version of the function `split_fun` given by

$$I = \texttt{split\_fun}(i_\beta, \ell),$$

where i_β is the global index of a function $\beta \in \mathcal{B}_\ell$ and I contains the global indices of the children of β. In this case, the algorithms described below can also be considerably simplified.

The hierarchical mesh $\mathcal{Q}$ can be defined through the variable $\texttt{MESH} = \{E_\ell^A, E_\ell^D\}_{\ell=0}^{n-1}$, where

- E_ℓ^A is the array containing the global indices of active cells of level ℓ, i.e., cells in $\mathcal{Q} \cap \mathcal{Q}_\ell$.
- E_ℓ^D is the array containing the global indices of deactivated cells of level ℓ, i.e., cells $Q \in \mathcal{Q}_\ell$ such that $Q \subset \Omega_{\ell+1}$. Notice that $E_{n-1}^D = \emptyset$.

On the other hand, the hierarchical B-spline basis $\mathcal{H}$ associated to $\mathcal{Q}$ can be described through the variable $\texttt{SPACE} = \{F_\ell^A, F_\ell^D, W_\ell\}_{\ell=0}^{n-1}$, where

- F_ℓ^A is the array containing the global indices of active B-splines of level ℓ, i.e., functions in $\mathcal{H} \cap \mathcal{B}_\ell$.
- F_ℓ^D is an array containing the global indices of B-splines in $\mathcal{B}_\ell$ whose supports are subsets of $\Omega_{\ell+1}$, i.e.,

  $$F_\ell^D := \{i_\beta \,|\, \beta \in \mathcal{B}_\ell \quad \wedge \quad \operatorname{supp} \beta \subset \Omega_{\ell+1}\}.$$

  Notice that $F_{n-1}^D = \emptyset$.
- W_ℓ is an array containing the values of the coefficients a_β for the partition of the unity (9) corresponding to the active B-splines β of level ℓ, i.e., to the functions in F_ℓ^A.

3.1 Getting the new active basis functions from the current ones

Let $\texttt{MARKED} = \{M_\ell\}_{\ell=0}^{n-1}$, where $M_\ell \subset F_\ell^A$ (or $M_\ell \subset E_\ell^A$) is the set of global indices of *marked* functions (or elements) of level ℓ, i.e., functions (or elements) in $\mathcal{M}_\ell$. Now, we present an algorithm for updating the information in MESH and SPACE when enlarging the hierarchy of subdomains $\mathbf{\Omega}_n$ with the marked functions or elements as explained in the previous section, cf. (11) and (12).

`function` $[\mathtt{MESH}, \mathtt{SPACE}] = \mathtt{refine}(\mathtt{MESH}, \mathtt{SPACE}, \mathtt{MARKED})$

% This function updates `MESH` and `SPACE` when enlarging the current subdomains with the marked functions (or elements) given in `MARKED`

Data: (Data for $\mathcal{Q}$, $\mathcal{H}$ and $\mathcal{M}$): $\begin{cases} \mathtt{MESH} = \{E_\ell^A, E_\ell^D\}_\ell \\ \mathtt{SPACE} = \{F_\ell^A, F_\ell^D, W_\ell\}_\ell \\ \mathtt{MARKED} = \{M_\ell\}_\ell \end{cases}$ $(\ell = 0, 1, \dots, n-1)$.

Result: (Data for $\mathcal{Q}^*$ and $\mathcal{H}^*$): $\begin{cases} \mathtt{MESH} = \{E_\ell^A, E_\ell^D\}_\ell \\ \mathtt{SPACE} = \{F_\ell^A, F_\ell^D, W_\ell\}_\ell \end{cases}$ $(\ell = 0, 1, \dots, n)$.

% REFINE MESH;

switch *marked functions or elements* **do**

1. **case** *functions,* $\mathtt{ME} = \mathtt{compute\_cells\_to\_refine}(\{E_\ell^A\}_{\ell=0}^{n-1}, \mathtt{MARKED})$;
2. **case** *elements,* $\mathtt{ME} = \mathtt{MARKED}$;

endsw

% `ME` contains the cells that have to be refined;

3. $[\mathtt{MESH}, \mathtt{NE}] = \mathtt{refine\_hierarchical\_mesh}(\mathtt{MESH}, \mathtt{ME})$;

% `NE` contains the new cells;

% REFINE SPACE;

4. $\mathtt{SPACE} = \mathtt{refine\_hierarchical\_space}(\mathtt{MESH}, \mathtt{SPACE}, \mathtt{MARKED}, \mathtt{NE})$;

Algorithm 1: `refine` (REFINE THE HIERARCHICAL MESH AND SPACE)

3.1.1 Routines for the mesh

function $\mathtt{ME} = \mathtt{compute\_cells\_to\_refine}(\{E^A_\ell\}_{\ell=0}^{n-1}, \mathtt{MARKED})$

% This function computes the indices of cells that have to be splitted when marking for refinement the functions in MARKED

Data: $\begin{cases} \{E^A_\ell\}_\ell \text{ (indices of active cells)} \\ \mathtt{MARKED} = \{M_\ell\}_\ell \end{cases} \quad (\ell = 0, 1, \ldots, n-1).$

Result: $\mathtt{ME} = \{\mathtt{ME}_\ell\}_{\ell=0}^{n-1}$ (Indices of cells that have to be refined)

foreach $\ell = 0, 1, \ldots, n-1$ **do**

1. Use get_cells to compute the set of indices $\mathtt{ME}_\ell$ of the cells of level ℓ that are included in the support of functions in M_ℓ;
2. $\mathtt{ME}_\ell \leftarrow \mathtt{ME}_\ell \cap E^A_\ell$ % Remove the nonactive cells from $\mathtt{ME}_\ell$;

end

Algorithm 2: compute_cells_to_refine

function $[\mathtt{MESH}, \mathtt{NE}] = \mathtt{refine\_hierarchical\_mesh}(\mathtt{MESH}, \mathtt{ME})$

% This function updates the active and deactive cells in each level when refining the cells in ME

Data: (Data for $\mathcal{Q}$ and ME): $\begin{cases} \mathtt{MESH} = \{E^A_\ell, E^D_\ell\}_\ell \\ \mathtt{ME} = \{\mathtt{ME}_\ell\}_\ell \end{cases} \quad (\ell = 0, 1, \ldots, n-1).$

Result: (Data for $\mathcal{Q}^*$ and NE): $\begin{cases} \mathtt{MESH} = \{E^A_\ell, E^D_\ell\}_\ell \\ \mathtt{NE} = \{\mathtt{NE}_\ell\}_\ell \end{cases} \quad (\ell = 0, 1, \ldots, n).$

1. **if** $\mathtt{ME}_{n-1} \neq \emptyset$, **then** $E^A_n = E^D_n = \emptyset$;

   foreach $\ell = 0, 1, \ldots, n-1$ **do**

   2. $E^A_\ell \leftarrow E^A_\ell \setminus \mathtt{ME}_\ell$ % Update E^A_ℓ by removing the cells to be refined;
   3. $E^D_\ell \leftarrow E^D_\ell \cup \mathtt{ME}_\ell$ % Update E^D_ℓ by adding the cells that were deactivated;
   4. Use split_cell to get the set of indices $\mathtt{NE}_{\ell+1}$ of cells of level $\ell+1$ which are inside of the cells in $\mathtt{ME}_\ell$;
   5. $E^A_{\ell+1} \leftarrow E^A_{\ell+1} \cup \mathtt{NE}_{\ell+1}$ % Update $E^A_{\ell+1}$ by adding the new active cells of level $\ell+1$;

   end

Algorithm 3: update_active_cells

3.1.2 Routines for the space

For the routines in this section notice that the variable MESH is already updated, i.e., it contains the information about $\mathcal{Q}^*$.

function $\texttt{SPACE} = \texttt{refine\_hierarchical\_space}(\texttt{MESH}, \texttt{SPACE}, \texttt{MARKED}, \texttt{NE})$

Data: (Data for $\mathcal{Q}^*$, $\mathcal{H}$, $\mathcal{M}$, and NE): $\begin{cases} \texttt{MESH} = \{E_\ell^A, E_\ell^D\}_{\ell=0}^n \\ \texttt{SPACE} = \{F_\ell^A, F_\ell^D, W_\ell\}_\ell \\ \texttt{MARKED} = \{M_\ell\}_\ell \\ \texttt{NE} = \{\texttt{NE}_\ell\}_\ell \end{cases}$ $(\ell = 0, \dots, n-1)$.

Result: (Data for $\mathcal{H}^*$): $\texttt{SPACE} = \{F_\ell^A, F_\ell^D, W_\ell\}_\ell \quad (\ell = 0, 1, \dots, n)$.

1. $\texttt{MF} = \texttt{compute\_functions\_to\_deactivate}(\texttt{MESH}, \texttt{SPACE}, \texttt{MARKED})$;

   % MF contains all B-splines that have to be deactivated;
2. $\texttt{SPACE} = \texttt{update\_active\_functions}(\texttt{MESH}, \texttt{SPACE}, \texttt{MF}, \texttt{NE})$;

Algorithm 4: refine_hierarchical_space

function $\texttt{MF} = \texttt{compute\_functions\_to\_deactivate}(\{E_\ell^A\}_{\ell=0}^n, \{F_\ell^A\}_{\ell=0}^{n-1}, \texttt{MARKED})$

% This function computes the indices of the functions that have to be deactivated when marking for refinement the functions (or elements) in MARKED

Data: $\begin{cases} \{E_\ell^A\}_{\ell=0}^n \text{ (indices of active cells)} \\ \{F_\ell^A\}_\ell \text{ (indices of active basis functions)} \\ \texttt{MARKED} = \{M_\ell\}_\ell \end{cases}$ $(\ell = 0, 1, \dots, n-1)$.

Result: $\texttt{MF} = \{\texttt{MF}_\ell\}_{\ell=0}^{n-1}$ (Indices of basis functions that have to be deactivated)

foreach $\ell = 0, 1, \dots, n-1$ **do**

% Computation of functions which *possibly* have to be removed;

switch *marked functions or elements* **do**

case *functions,*

1. Use get_neighbors to compute the set of indices $\texttt{MF}_\ell$ of functions of level ℓ whose supports intersect the support of a function in M_ℓ;
2. $\texttt{MF}_\ell \leftarrow (\texttt{MF}_\ell \cap F_\ell^A) \setminus M_\ell$ % Remove from $\texttt{MF}_\ell$ the nonactive functions and the active functions already selected for deactivation;

end

case *elements,*

3. Use get_basis_functions to compute the set of indices $\texttt{MF}_\ell$ of functions of level ℓ whose supports intersect at least one cell in M_ℓ;
4. $\texttt{MF}_\ell \leftarrow (\texttt{MF}_\ell \cap F_\ell^A)$ % Remove from $\texttt{MF}_\ell$ the nonactive functions;

end

endsw

% Computation of functions which *in fact* have to be removed;

5. Update $\texttt{MF}_\ell$ by removing the functions that have al least one active cell of level ℓ within their supports. Use get_cells;
6. **if** *marking functions,* **then** $\texttt{MF}_\ell = \texttt{MF}_\ell \cup M_\ell$;

end

Algorithm 5: compute_functions_to_deactivate

`function` $\quad$ $\texttt{SPACE} = \texttt{update\_active\_functions}(\texttt{MESH}, \texttt{SPACE}, \texttt{MF}, \texttt{NE})$;

% This function updates the active and deactive B-spline basis functions. The inputs `MESH` and `NE` are already updated (see `refine_hierarchical_mesh`) and `MF` contains the indices of *all B-splines that have to be deactivated* (see `compute_functions_to_deactivate`). Finally, the variable `SPACE` in the input contains the information before updating the space

Data: (Data for $\mathcal{Q}^*$, $\mathcal{H}$, `MF` and `NE`): $\begin{cases} \texttt{MESH} = \{E_\ell^A, E_\ell^D\}_{\ell=0}^n \\ \texttt{SPACE} = \{F_\ell^A, F_\ell^D, W_\ell\}_{\ell=0}^{n-1} \\ \texttt{MF} = \{\texttt{MF}_\ell\}_{\ell=0}^{n-1} \\ \texttt{NE} = \{\texttt{NE}_\ell\}_{\ell=0}^{n} \end{cases}$

Result: (Data for $\mathcal{H}^*$): $\texttt{SPACE} = \{F_\ell^A, F_\ell^D, W_\ell\}_{\ell=0}^n$

1. **if** $\texttt{MF}_{n-1} \neq \emptyset$, **then** $F_n^A = F_n^D = W_n = \emptyset$;

% Update of $\{F_\ell^A, F_\ell^D, W_\ell\}_\ell$;

foreach $\ell = 0, 1, \ldots, n-1$ **do**

2. $\texttt{MF}_\ell \leftarrow \texttt{MF}_\ell \cup (F_\ell^A \cap F_\ell^D)$ % This line is important for $\ell \geq 1$;
3. $F_\ell^A \leftarrow F_\ell^A \setminus \texttt{MF}_\ell$ % Remove $\texttt{MF}_\ell$ from the active functions of level ℓ;
4. Save in `W` the values of W_ℓ corresponding to functions in $\texttt{MF}_\ell$ and update W_ℓ by removing these values;
5. $F_\ell^D \leftarrow F_\ell^D \cup \texttt{MF}_\ell$ % Update F_ℓ^D by adding the functions to be deactivated;

foreach β *in* $\texttt{MF}_\ell$ **do**

6. Compute the set of indices I_β of the B-splines of level $\ell+1$ and the corresponding coefficients c of the two-scale relation (4) when writing β as a linear combination of functions in $\mathcal{B}_{\ell+1}$
7. Use `get_cells` to update $F_{\ell+1}^D$ by adding the functions in $(I_\beta \setminus F_{\ell+1}^A) \setminus F_{\ell+1}^D$ that have no active cell of level $\ell+1$ within its support. REMOVED
8. $F_{\ell+1}^A \leftarrow F_{\ell+1}^A \cup I_\beta$ % Enlarge $F_{\ell+1}^A$ by adding the *possible* new active functions;
9. Enlarge $W_{\ell+1}$ in order to match with the new $F_{\ell+1}^A$, setting equal to zero the coefficients corresponding to the new functions in $F_{\ell+1}^A$.

   % Now, we update the values in $W_{\ell+1}$
10. **foreach** $\beta_{\ell+1}$ *in* I_β **do** $a_{\beta_{\ell+1}} \leftarrow a_{\beta_{\ell+1}} + a_\beta * c_{\beta_{\ell+1}}$, where a_β is the value in `W` corresponding to β, and $a_{\beta_{\ell+1}}$ and $c_{\beta_{\ell+1}}$ are the values in $W_{\ell+1}$ and c, respectively, corresponding to $\beta_{\ell+1}$;

end

% Now, we activate B-splines of level $\ell+1$ that are not children of any deactivated B-spline of level ℓ;

11. Use `get_basis_functions` to compute the indices I of the B-splines of level $\ell+1$ that do not vanish in some cell of $\texttt{NE}_{\ell+1}$;
12. $I \leftarrow I \setminus F_{\ell+1}^A$ % Remove from I the functions which were already active;
13. Use `get_cells` to update I by removing the functions such that have at least one cell of level $\ell+1$ in their support that does not belong to $E_{\ell+1}^A \cup E_{\ell+1}^D$;
14. $F_{\ell+1}^A \leftarrow F_{\ell+1}^A \cup I$;
15. Enlarge $W_{\ell+1}$ in order to match with the new $F_{\ell+1}^A$, setting equal to zero the coefficients corresponding to the new functions in $F_{\ell+1}^A$;

end

Algorithm 6: `update_active_functions`

3.2 Initialization of a hierarchical spline space

The function `refine` detailed in the previous section can be also used to select the active functions of a hierarchical B-spline basis $\mathcal{H}$, if we know the active cells of each level $\{E_\ell^A\}_{\ell=0}^{n-1}$.

`function` $[\texttt{MESH}, \texttt{SPACE}] = \texttt{build\_hierarchical\_space}(\{E_\ell^A\}_{\ell=0}^{n-1})$

Data: $\{E_\ell^A\}_{\ell=0}^{n-1}$ (Active elements in each level)

Result: (Data for $\mathcal{Q}$ and $\mathcal{H}$): $\begin{cases} \texttt{MESH} = \{E_\ell^A, E_\ell^D\}_\ell \\ \texttt{SPACE} = \{F_\ell^A, F_\ell^D, W_\ell\}_\ell \end{cases}$ $(\ell = 0, 1, \ldots, n-1)$.

1. Define $\hat{E}_0^A$ as the set of indices of all the cells in $\mathcal{Q}_0$;
2. Define F_0^A as the set of indices of all the basis functions in $\mathcal{B}_0$;
3. Define the corresponding array of weights W_0 setting all the values equal to 1;
4. $\hat{E}_0^D = \emptyset$;
5. $F_0^D = \emptyset$;
6. $\texttt{MESH} = \{\hat{E}_0^A, \hat{E}_0^D\}$;
7. $\texttt{SPACE} = \{F_0^A, F_0^D, W_0\}$;

foreach $k = 0, \ldots, n-2$ **do**

8. Let $M_0 = \cdots = M_{k-1} = \emptyset$ and $M_k = \hat{E}_k^A \setminus E_k^A$. Then, $\texttt{MARKED} = \{M_\ell\}_{\ell=0}^k$;
9. $[\texttt{MESH}, \texttt{SPACE}] = \texttt{refine}(\texttt{MESH}, \texttt{SPACE}, \texttt{MARKED})$;

end

Algorithm 7: `build_hierarchical_space`

Remark 3.2. In order to make the last algorithm more understandable we have used the function `refine`. This algorithm can be improved by adapting the code of `refine` instead of call it. Notice that the set of marked cells in each of the calls to `refine` has cells only of one level.

4 Assembly of the matrix

SOME NOTATION, to be changed after discussion:

- N_ℓ: number of basis functions in $\mathcal{B}_\ell$.
- A_ℓ: number of active functions of level ℓ. Give a name to the sets in (7).

- $\tilde{A}_\ell := \sum_{k=0}^{\ell} A_\ell$, number of active functions up to level ℓ.
- $\beta^A_{j,\ell}$ active functions. Probably we can avoid these.

One of the issues when implementing IGA with hierarchical splines is the assembly of the matrices. In order to compute the matrices of the discrete problem it is necessary to evaluate integrals that involve basis functions from different levels, and possibly in a level that does not correspond to the level of the functions. For instance, to compute one entry of the mass matrix, one must compute

$$\int_\Omega \beta_{i,\ell_i}\beta_{j,\ell_j}\, d\mathbf{x} = \sum_{Q\in\mathcal{Q}} \int_Q \beta_{i,\ell_i}\beta_{j,\ell_j}\, d\mathbf{x} = \sum_{\ell=\max\{\ell_i,\ell_j\}}^{n} \sum_{Q_\ell\in\mathcal{Q}_\ell} \int_{Q_\ell} \beta_{i,\ell_i}\beta_{j,\ell_j}\, d\mathbf{x}.$$

In order to compute the integrals, we take advantage of the two-scale relation (4): I AM CHANGING THE NOTATION

$$\beta_{i,\ell} = \sum_{k=1}^{N_{\ell+1}} c_{k,\ell+1}(\beta_{i,\ell})\beta_{k,\ell+1}, \qquad \forall\, \beta_{i,\ell}\in\mathcal{B}_\ell,$$

and computing the coefficients for each basis function, we obtain an operator $C_\ell^{\ell+1} : \mathcal{S}_\ell \longrightarrow \mathcal{S}_{\ell+1}$, which can be written in matrix form.

Succesively applying the two-scale relation, we obtain the general version

$$\beta_{i,\ell} = \sum_{k=1}^{N_{\ell+m}} c_{k,\ell+m}(\beta_{i,\ell})\beta_{k,\ell+m}, \qquad \forall\, \beta_{i,\ell}\in\mathcal{B}_\ell,$$

and the matrix operator $C_\ell^{\ell+m} = C_{\ell+m-1}^{\ell+m}\dots C_{\ell+1}^{\ell+2}C_\ell^{\ell+1}$.

Plugging this expression into our integral, and after reordering, we obtain that

$$\int_\Omega \beta_{i,\ell_i}\beta_{j,\ell_j}\, d\mathbf{x} = \sum_{\ell=\max\{\ell_i,\ell_j\}}^{n} \sum_{Q_\ell\in\mathcal{Q}_\ell} \int_{Q_\ell} \left(\sum_{k_i=1}^{N_\ell} c_{k_i,\ell}(\beta_{i,\ell_i})\beta_{k_i,\ell}\right)\left(\sum_{k_j=1}^{N_\ell} c_{k_j,\ell}(\beta_{j,\ell_j})\beta_{k_j,\ell}\right) d\mathbf{x} =$$
$$\sum_{\ell=\max\{\ell_i,\ell_j\}}^{n} \sum_{k_i=1}^{N_\ell}\sum_{k_j=1}^{N_\ell} c_{k_i,\ell}(\beta_{i,\ell_i})c_{k_j,\ell}(\beta_{j,\ell_j}) \sum_{Q_\ell\in\mathcal{Q}_\ell} \int_{Q_\ell} \beta_{k_i,\ell}\beta_{k_j,\ell}\, d\mathbf{x}.$$

After the arrangements, we see that it is only needed to compute, in the active elements of level ℓ, the integrals involving the tensor product functions of that level, both active and inactive. The computation of these integrals is easily available in any software for IGA.

The only missing part is the computation of the coefficients in an efficient way. This can be done rewriting the equations in matrix form. Let us denote by M_ℓ the matrix obtained from function of level ℓ, that is

$$(M_\ell)_{k_ik_j} = \sum_{Q_\ell\in\mathcal{Q}_\ell} \int_{Q_\ell} \beta_{k_i,\ell}\beta_{k_j,\ell}\, d\mathbf{x},$$

and let us also denote by $\tilde{I}_\ell \in \mathcal{M}_{N_\ell, A_\ell}$ the rectangular matrix such that

$$(\tilde{I}_\ell)_{ij} = \begin{cases} 1, & \text{if } \beta_{i,\ell} = \beta^A_{j,\ell} (active) \\ 0, & \text{otherwise.} \end{cases}$$

We define the matrices for basis change $C_\ell \in \mathcal{M}_{N_\ell, \tilde{A}_\ell}$ by the recursive algorithm

1. $C^0 = \tilde{I}_0$.
2. $C_\ell = [C^\ell_{\ell-1} C_{\ell-1}, \tilde{I}_\ell]$.

By doing so, the global mass matrix is written as

$$M = \sum_{\ell=0}^{n} C_\ell M_\ell C_\ell^T.$$

References

[BG15a] A. Buffa and E.M. Garau, *New refinable spaces and local approximation estimates for hierarchical splines*, submitted, 2015.

[BG15b] A. Buffa and E.M. Garau, *A posteriori error estimators for hierarchical B-spline discretizations*, in preparation, 2015.

[dB01] C. de Boor, *A practical guide to splines*, Revised edition. Applied Mathematical Sciences, 27. Springer-Verlag, New York, 2001.

[GJS14] C. Giannelli, B. Jüttler, and H. Speleers, *Strongly stable bases for adaptively refined multilevel spline spaces*, Adv. Comput. Math. 40 (2014), no. 2, 459–490.

[K98] R. Kraft, *Adaptive und linear unabhängige multilevel B-Splines und ihre Anwendungen*, Ph.D. thesis, Universität Stuttgart, 1998.

[S07] L.L. Schumaker, *Spline functions: basic theory*, Third edition, Cambridge Mathematical Library, Cambridge University Press, Cambridge, 2007.

[VGJS11] A.-V. Vuong, C. Giannelli, B. Jüttler and B. Simeon, *A hierarchical approach to adaptive local refinement in isogeometric analysis*, Comput. Methods Appl. Mech. Engrg. 200 (2011), no. 49-52, 3554–3567.