ALGORITHMS FOR LOCAL REFINEMENT IN HIERARCHICAL SPLINE SPACES. UPDATE

EDUARDO M. GARAU*† AND RAFAEL VÁZQUEZ‡

$\mathtt{ME} \leftarrow$ **compute_cells_to_refine** $(\{\mathtt{E}_\ell^\mathtt{A}\}, \{\mathtt{MARKED}_\ell\})$
$\mathtt{ME} \leftarrow$ **Compute_Cells_To_Refine** $(\{\mathtt{E}_\ell^\mathtt{A}\}, \{\mathtt{MARKED}_\ell\})$
$\mathtt{ME} \leftarrow$ **ComputeCellsToRefine** $(\{\mathtt{E}_\ell^\mathtt{A}\}, \{\mathtt{MARKED}_\ell\})$
$\mathtt{ME} \leftarrow$ ComputeCellsToRefine $(\{\mathtt{E}_\ell^\mathtt{A}\}, \{\mathtt{MARKED}_\ell\})$

Me gusta más la primera opción y luego la última.

1. Introduction. [8, 1], [6]?

2. 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 $\mathbf{p} := (p_1, p_2, \dots, p_d)$ denotes the vector of polynomial degrees of the splines with respect to each coordinate direction.

2.1. 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

$$\mathcal{S}_0 \subset \mathcal{S}_1 \subset \mathcal{S}_2 \subset \mathcal{S}_3 \subset \dots, \tag{2.1}$$

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

$$\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \dots, \tag{2.2}$$

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 [4, 7]:

- *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.,

$$\sum_{\beta\in\mathcal{B}_\ell} \beta \equiv 1, \qquad \text{on } \Omega. \tag{2.3}$$

- *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,

$$\beta_\ell = \sum_{\beta_{\ell+1}\in\mathcal{C}(\beta_\ell)} c_{\beta_{\ell+1}}(\beta_\ell)\beta_{\ell+1}, \qquad \forall\, \beta_\ell \in \mathcal{B}_\ell, \tag{2.4}$$

  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 [2].

*Instituto de Matemática Aplicada del Litoral (CONICET-UNL), Argentina
†Facultad de Ingeniería Química (UNL), Argentina
‡Istituto di Matematica Applicata e Tecnologie Informatiche 'E. Magenes' (CNR), Italy

REMARK 2.1. *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* (2.4) *can be written as*

$$\beta_\ell = \sum_{\beta_{\ell+1}\in\mathcal{B}_{\ell+1}} c_{\beta_{\ell+1}}(\beta_\ell)\beta_{\ell+1}, \qquad \forall\, \beta_\ell \in \mathcal{B}_\ell. \tag{2.5}$$

In particular, we remark that

$$\mathcal{C}(\beta_\ell) = \{\beta_{\ell+1} \in \mathcal{B}_{\ell+1} \,|\, c_{\beta_{\ell+1}}(\beta_\ell) > 0\} \subset \{\beta_{\ell+1} \in \mathcal{B}_{\ell+1} \,|\, \operatorname{supp}\beta_{\ell+1} \subset \operatorname{supp}\beta_\ell\}. \tag{2.6}$$

2.2. Hierarchical B-spline basis and hierarchical spline space. DEFINITION 2.2. *If* $n \in \mathbb{N}$*, we say that* $\mathbf{\Omega}_n := \{\Omega_0, \Omega_1, \ldots, \Omega_n\}$ *is a* hierarchy of subdomains of Ω of depth n *if*

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

(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

$$\mathcal{H} = \bigcup_{\ell=0}^{n-1} \{\beta \in \mathcal{B}_\ell \mid \operatorname{supp}\beta \subset \Omega_\ell \wedge \operatorname{supp}\beta \not\subset \Omega_{\ell+1}\}. \tag{2.7}$$

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

$$\mathcal{Q} := \bigcup_{\ell=0}^{n-1} \{Q \in \mathcal{Q}_\ell \;\mid\; Q \subset \Omega_\ell \;\wedge\; Q \not\subset \Omega_{\ell+1}\}, \tag{2.8}$$

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

$$\sum_{\beta\in\mathcal{H}} a_\beta \beta \equiv 1, \qquad \text{on } \Omega. \tag{2.9}$$

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 $\mathbf{\Omega}_n$.

3. 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 3.1. *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

$$\mathcal{H}^* := \bigcup_{\ell=0}^{n} \{\beta \in \mathcal{B}_\ell \mid \operatorname{supp}\beta \subset \Omega^*_\ell \wedge \operatorname{supp}\beta \not\subset \Omega^*_{\ell+1}\}, \tag{3.1}$$

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 [5] 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

$$\begin{cases} \Omega^*_0 & := \Omega_0, \\ \Omega^*_\ell & := \Omega_\ell \cup \displaystyle\bigcup_{\beta\in\mathcal{M}_{\ell-1}} \operatorname{supp}\beta, \qquad \ell = 1, 2, \ldots, n, \\ \Omega^*_{n+1} & := \emptyset. \end{cases} \tag{3.2}$$

  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

$$\begin{cases} \Omega^*_0 & := \Omega_0, \\ \Omega^*_\ell & := \Omega_\ell \cup \displaystyle\bigcup_{Q\in\mathcal{M}_{\ell-1}} Q, \qquad \ell = 1, 2, \ldots, n, \\ \Omega^*_{n+1} & := \emptyset. \end{cases} \tag{3.3}$$

  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.

4. 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 $\mathcal{S}_\ell$:
- $I = \texttt{get\_cells}(i_\beta, \ell)$, where i_β is the global index of a function $\beta \in \mathcal{B}_\ell$, and I contains the global indices of the cells in $\mathcal{Q}_\ell$ which are subsets of $\operatorname{supp}\beta$.
- $I = \texttt{get\_neighbors}(i_\beta, \ell)$, where i_β is the global index of a function $\beta \in \mathcal{B}_\ell$, and I contains the global indices of functions in $\mathcal{B}_\ell$ whose supports have at least one cell of level ℓ within $\operatorname{supp}\beta$.
- $I = \texttt{get\_basis\_functions}(i_Q, \ell)$, 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 $\mathcal{S}_{\ell+1}$):
- $I = \texttt{split\_cell}(i_Q, \ell)$, 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.
- $[I, c] = \texttt{split\_fun}(i_\beta, \ell)$, 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 (2.4).

REMARK 4.1. *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* (2.4) *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.* (2.9)*). We remark that some a posteriori error estimators may require this information [3].*

Finally, we remark that if we do not need the explicit knowledge of the coefficients in (2.9)*, we can consider a simple version of the function* **split_fun** *given by*

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

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 (2.9) corresponding to the active B-splines β of level ℓ, i.e., to the functions in F_ℓ^A.

4.1. Getting the new active basis functions from the current ones. Let `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. (3.2) and (3.3).

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

Algorithm 1 Refine

Input: `MESH`, `SPACE`, `MARKED`
if (Marking functions) **then**
  `ME` $\leftarrow$ **compute_cells_to_refine** ($\{\texttt{E}_\ell^\texttt{A}\}$, $\{\texttt{MARKED}_\ell\}$)
else if (Marking elements) **then**
  `ME` $\leftarrow$ `MARKED`
end if
`MESH` $\leftarrow$ **refine_hierarchical_mesh** (`ME`) ▷ Think about these two lines
`NE` $\leftarrow$ **get_children** (`ME`)
`SPACE` $\leftarrow$ **refine_hierarchical_space** (`MESH`, `SPACE`, `MARKED`, `NE`)
Output: `MESH`, `SPACE`

4.1.1. Routines for the mesh.

Algorithm 2 `compute_cells_to_refine`

Input: $\{\texttt{E}_\ell^\texttt{A}\}$, $\{\texttt{MF}_\ell\}$
for $\ell = 0, \ldots, n-1$ **do**
  $\texttt{ME}_\ell \leftarrow$ **get_cells** ($\texttt{MF}_\ell$)
  $\texttt{ME}_\ell \leftarrow \texttt{ME}_\ell \cap \texttt{E}_\ell^\texttt{A}$
end for
Output: $\{\texttt{ME}_\ell\}$

4.1.2. 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 $\mathcal{Q}^*$.

Algorithm 3 Refine_hierarchical_mesh

Input: MESH, {ME}
if $\texttt{ME}_{n-1} \neq \emptyset$ **then**
  MESH ← **initialize_empty_level (n)**
end if
for $\ell = 0, \dots, n-1$ **do**
  $\texttt{E}^{\texttt{A}}_{\ell} \leftarrow \texttt{E}^{\texttt{A}}_{\ell} \cap \texttt{ME}_{\ell}$
  $\texttt{E}^{\texttt{D}}_{\ell} \leftarrow \texttt{E}^{\texttt{D}}_{\ell} \cup \texttt{ME}_{\ell}$
  $\texttt{NE}_{\ell} \leftarrow$ **get_children** $(\texttt{ME}_{\ell})$ ▷ Should we move this out?
  $\texttt{E}^{\texttt{A}}_{\ell+1} \leftarrow \texttt{E}^{\texttt{A}}_{\ell+1} \cup \texttt{NE}_{\ell+1}$
end for
Output: MESH, $\{\texttt{NE}_{\ell}\}$

Algorithm 4 refine_hierarchical_space

Input: MESH, SPACE, $\{\texttt{MARKED}_{\ell}\}$, $\{\texttt{NE}_{\ell}\}$
MF ← **compute_functions_to_deactivate** (MESH, SPACE, MARKED)
if $\texttt{MARKED}_{n-1} \neq \emptyset$ **then**
  SPACE ← **initialize_empty_level (n)**
end if
REFINED_SPACE ← **Update_Active_Functions** (MESH, SPACE, MF, MARKED) ▷ Split in two (elements and functions)?
K ← **Compute_Refinement_Matrix** (SPACE, REFINED_SPACE)
Output: REFINED_SPACE, K ▷ The refined space

4.2. Refinement matrix between hierarchical spaces (knot insertion). Since the spaces are nested, $\operatorname{span}\mathcal{H} \subset \operatorname{span}\mathcal{H}^*$, we can write a function in the first space as a linear combination of basis functions in the second one, and the operation can be done through a matrix. It is possible to use the information of active and deactivated functions to obtain this.

MORE NOTATION TO BE FIXED

- N_ℓ: number of basis functions for the tensor-product space of level ℓ.
- N_ℓ^A: number of active functions of level ℓ in the hierarchical space.
- $N_\ell^{A\cup D}$: number of active and deactivated functions of level ℓ.

The analogous notation with $*$ is used for the refined space.

Since any (active) function in $\mathcal{H}$ is either in $\mathcal{H}^*$ or in the set of deactivated functions (DEFINE), we can define the rectangular matrix $\tilde{I}_\ell \in \mathcal{M}_{N_\ell^{*A\cup D}, N_\ell^A}$ such that

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

We define the matrix for refinement $K = K_n$ with the following recursive algorithm:

1. $K_0 = \tilde{I}_0$,
2. $K_{\ell+1} = \begin{bmatrix} K_\ell^{A^*} & 0 \\ K_\ell^{\ell+1} K_\ell^{D^*} & \tilde{I}_{\ell+1} \end{bmatrix}$,

where $K_\ell^{A^*}$ and $K_\ell^{D^*}$ are submatrices of K_ℓ restricted to the rows corresponding to active and deactivated functions in $\mathcal{H}^*$. The matrix $K_\ell^{\ell+1}$ is the submatrix of $C_\ell^{\ell+1}$

Algorithm 5 Compute_Functions_To_Deactivate

Input: MESH, SPACE, MARKED
for $\ell = 0, \ldots, n-1$ **do**
  if (Marking functions) **then**
    $\mathtt{MF}_\ell \leftarrow$ **get_neighbors** $(\mathtt{MARKED}_\ell)$
    $\mathtt{MF}_\ell \leftarrow (\mathtt{MF}_\ell \cap \mathtt{F}^\mathtt{A}_\ell) \setminus \mathtt{MARKED}_\ell$ ▷ Why not $\mathtt{MF}_\ell \leftarrow \mathtt{MF}_\ell \cap \mathtt{F}^\mathtt{A}_\ell$?
  else if (Marking elements) **then**
    $\mathtt{MF}_\ell \leftarrow$ **get_basis_functions** $(\mathtt{MARKED}_\ell)$
    $\mathtt{MF}_\ell \leftarrow \mathtt{MF}_\ell \cap \mathtt{F}^\mathtt{A}_\ell$
  end if
  Update $\mathtt{MF}_\ell$ by removing the functions that have al least one active cell of level ℓ within their supports. Use `get_cells`
  if (Marking functions) **then**
    $\mathtt{MF}_\ell \leftarrow \mathtt{MF}_\ell \cup \mathtt{MARKED}_\ell$
  end if
end for
Output: $\{\mathtt{MF}\}$

Algorithm 6 Update_active_functions: simplified hierarchical space

Input: MESH, SPACE, $\{\mathtt{MF}\}$
for $\ell = 0, \ldots, n-1$ **do**
  $\mathtt{F}^\mathtt{A}_\ell \leftarrow \mathtt{F}^\mathtt{A}_\ell \setminus \mathtt{MF}_\ell$
  $\mathtt{F}^\mathtt{D}_\ell \leftarrow \mathtt{F}^\mathtt{D}_\ell \cup \mathtt{MF}_\ell$
  $\mathtt{F}^\mathtt{C} \leftarrow$ **get_children** $(\mathtt{MF}_\ell)$
  $\mathtt{F}^\mathtt{C} \leftarrow \mathtt{F}^\mathtt{C} \setminus \mathtt{F}^\mathtt{A}_{\ell+1} \cup \mathtt{F}^\mathtt{D}_{\ell+1}$
  $\mathtt{F}^\mathtt{A}_{\ell+1} \leftarrow (\mathtt{F}^\mathtt{A}_{\ell+1} \cup \mathtt{F}^\mathtt{C})$
  Use `get_cells` to update $\mathtt{MF}_{\ell+1}$ by adding the functions in $\mathtt{F}^\mathtt{C}$ that have no active cell of level $\ell+1$ within its support.
end for
Output: $\mathtt{REFINED\_SPACE}(\{\mathtt{F}^\mathtt{A}\}, \{\mathtt{F}^\mathtt{D}\})$

restricted to the rows of active and deactivated functions of level $\ell+1$ in $\mathcal{H}^*$, and the columns of deactivated functions of level ℓ in $\mathcal{H}^*$.

We remark that this recursive algorithm is independent of the chosen hierarchical space (either standard or simplified). To compute the same matrix for truncated hierachical B-splines, one only needs to replace the matrix $C_\ell^{\ell+1}$ (and its submatrix) by its truncated version. (EXPLAIN IN PREVIOUS SECTIONS)

The refinement matrix K may become useful in several occasions. For instance, it can be used for applying knot insertion; in time dependent problems, it can serve to pass the solution from the previous time step to the current mesh; in multigrid methods, it can be used to pass from a grid of n levels to a grid of $n+1$ levels...

4.3. 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 $\{E_\ell^A\}_{\ell=0}^{n-1}$.

REMARK 4.2. *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*

Algorithm 7 Update_active_functions: standard hierarchical space

Input: $\mathtt{MESH}, \mathtt{SPACE}, \{\mathtt{MF}\}\{\mathtt{NE}\}$

for $\ell = 0, \ldots, n-1$ **do**
    $\mathtt{F}^{\mathtt{A}}_{\ell} \leftarrow \mathtt{F}^{\mathtt{A}}_{\ell} \setminus \mathtt{MF}_{\ell}$
    $\mathtt{F}^{\mathtt{D}}_{\ell} \leftarrow \mathtt{F}^{\mathtt{D}}_{\ell} \cup \mathtt{MF}_{\ell}$
    $\mathtt{F}^{\mathtt{C}} \leftarrow$ **get_basis_functions** $(\mathtt{NE}_{\ell+1})$
    $\mathtt{F}^{\mathtt{C}} \leftarrow \mathtt{F}^{\mathtt{C}} \setminus \mathtt{F}^{\mathtt{A}}_{\ell+1}$
    Use `get_cells` to update $\mathtt{F}^{\mathtt{C}}$ by removing the functions such that have at least one cell of level $\ell+1$ in their support that does not belong to $\mathtt{E}^{\mathtt{A}}_{\ell+1} \cup \mathtt{E}^{\mathtt{D}}_{\ell+1}$
    $\mathtt{F}^{\mathtt{A}}_{\ell+1} \leftarrow \mathtt{F}^{\mathtt{A}}_{\ell+1} \cup \mathtt{F}^{\mathtt{C}}$
end for

Output: $\mathtt{REFINED\_SPACE}(\{\mathtt{F}^{\mathtt{A}}\}, \{\mathtt{F}^{\mathtt{D}}\})$

Algorithm 8 build_hierarchical_space

Input: $\{\mathtt{E}^{\mathtt{A}}\}$

$\mathtt{MESH} \leftarrow$ **mesh_cartesian** ▷ Initialize as a Cartesian grid of level 0
$\mathtt{SPACE} \leftarrow$ **spline_space** ▷ Initialize as a tensor product space of level 0
for $\ell = 0, n-2$ **do**
    $\{\mathtt{MARKED}\}_0^{\ell-1} \leftarrow \emptyset$
    $\mathtt{MARKED}_{\ell} \leftarrow \hat{\mathtt{E}}^{\mathtt{A}}_{\ell} \setminus \mathtt{E}^{\mathtt{A}}_{\ell}$
    $[\mathtt{MESH}, \mathtt{SPACE}] \leftarrow$ **refine** $(\mathtt{MESH}, \mathtt{SPACE}, \mathtt{MARKED})$
end for

Output:

`refine` *has cells only of one level.*

5. Assembly of the matrix. SOME NOTATION, to be changed after discussion:

- N_ℓ: number of basis functions in $\mathcal{B}_\ell$.
- N_ℓ^A: number of active functions of level ℓ. Give a name to the sets in (2.7).
- $\tilde{N}_\ell^A := \sum_{k=0}^{\ell} N_k^A$, number of active functions up to level ℓ.
- $\beta_{j,\ell}^A$ 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 (2.4): 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

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

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

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

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

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

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

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

1. $C_0 = \tilde{I}_0$.
2. $C_\ell = [C_{\ell-1}^\ell C_{\ell-1}, \tilde{I}_\ell]$.

By doing so, the global mass matrix is written as

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

REFERENCES

[1] P.B. Bornemann and F. Cirak, *A subdivision-based implementation of the hierarchical B-spline finite element method*, Comput. Methods Appl. Mech. Engrg., 253 (2013), pp. 584 – 598.

[2] Annalisa Buffa and Eduardo M. Garau, *New refinable spaces and local approximation estimates for hierarchical splines*, tech. report, 2015.

[3] ———, *A posteriori error estimators for hierarchical B-spline discretizations*, tech. report, 2015.

[4] Carl de Boor, *A practical guide to splines*, vol. 27 of Applied Mathematical Sciences, Springer-Verlag, New York, revised ed., 2001.

[5] Carlotta Giannelli, Bert Jüttler, and Hendrik Speleers, *Strongly stable bases for adaptively refined multilevel spline spaces*, Adv. Comput. Math., 40 (2014), pp. 459–490.

[6] Gábor Kiss, Carlotta Giannelli, and Bert Jüttler, *Algorithms and data structures for truncated hierarchical B-splines*, in Mathematical Methods for Curves and Surfaces, Michael Floater, Tom Lyche, Marie-Laurence Mazure, Knut Mrken, and Larry L. Schumaker, eds., vol. 8177 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2014, pp. 304–323.

[7] Larry L. Schumaker, *Spline functions: basic theory*, Cambridge Mathematical Library, Cambridge University Press, Cambridge, third ed., 2007.

[8] M.A. Scott, D.C. Thomas, and E.J. Evans, *Isogeometric spline forests*, Computer Methods in Applied Mechanics and Engineering, 269 (2014), pp. 222 – 264.