Marko Petek

Univerza v Mariboru
Fakulteta za naravoslovje in matematiko

Maribor,
13. VII. 2019

Block Structured Mesh

BlockMesh.cs

Two 2nd rank tensors serve as storage for node variable values (node vars):

$$\mathbf{u}_\triangleright \qquad u_\triangleright^{jl} \qquad \textsf{Tensor Uf} \qquad \texttt{free node vars} \, ,$$
$$\mathbf{u}_\triangleleft \qquad u_\triangleleft^{jl} \qquad \textsf{Tensor Uc} \qquad \texttt{constrained node vars} \, .$$

where the slots represent:

$$j \;\rightarrow\; N \;\texttt{nodes},$$
$$l \;\rightarrow\; m \;\texttt{variables}.$$

The two tensors hold mutually exclusive information - if the component $u^{5,4}$ appears in $\mathbf{u}_\triangleright$, it cannot appear in $\mathbf{u}_\triangleleft$ because a variable is either constrained or it isn't. The sum of them thus produces a tensor which holds all values:

$$\mathbf{u}_\bowtie = \mathbf{u}_\triangleright + \mathbf{u}_\triangleleft \qquad \textsf{double U} \qquad \texttt{all = free + constrained} \, .$$

Here U is a method that can access values from both Uf and Uc - given an index, it retrieves the value from the correct source. A third 2nd rank tensor stores all forcing vars (right-hand side od `PDE`):

$$\mathbf{f}_\bowtie \qquad f_\bowtie^{jl} \qquad \textsf{Tensor F} \qquad \texttt{forcing vars} \, .$$

The dynamic parameters (determining the system's evolution in the next step) are stored in a 4th rank tensor $\mathbf{A}$, also known as the stiffness matrix:

$$\mathbf{A} \qquad A^{iphl} \qquad \textsf{Tensor A} \qquad \texttt{stiffness matrix} \, ,$$

where the slots represent:

$$i \;\rightarrow\; N \;\texttt{nodes},$$
$$p \;\rightarrow\; 3 \;\texttt{partials},$$
$$h \;\rightarrow\; m \;\texttt{equations},$$
$$l \;\rightarrow\; m \;\texttt{variables} \, .$$

Dynamics must not depend on element shapes. This is properly accounted for with node-to-node influence weights in the form of overlap integrals. Triple overlap integrals reside in a 7th rank tensor **S**, while double overlap integrals reside in a 5th rank tensor **T**:

S	$S_{\varepsilon\beta\alpha p\gamma\delta q}$	Tensor S	`tripple overlap integrals ,`
T	$T_{\varepsilon\beta\alpha p\gamma}$	Tensor T	`double overlap integrals ,`

where the slots represent:

$$\begin{aligned}
\varepsilon &\rightarrow n \texttt{ elements,} \\
\beta &\rightarrow 12 \texttt{ basis funcs of 1st } \mathbf{A}\,, \\
\alpha &\rightarrow 12 \texttt{ basis funcs of } \mathbf{v}\,, \\
p &\rightarrow 3 \texttt{ partials of } \mathbf{v}\,, \\
\gamma &\rightarrow 12 \texttt{ basis funcs of } \mathbf{u}_{\triangleleft} \texttt{ or } \mathbf{f}\,, \\
\delta &\rightarrow 12 \texttt{ basis funcs of } \mathbf{u}_{\triangleright}\,, \\
q &\rightarrow 3 \texttt{ partials of } \mathbf{u}_{\triangleright}\,.
\end{aligned}$$

Assembly process turns this:

$$\forall\, v_{\triangleright}^{ik} : \quad \sum_{i,j}^{N} v_{\triangleright}^{ik} \sum_{\varepsilon}^{n} \sum_{\substack{\alpha,\delta\,\ni: \\ (\varepsilon,\alpha)=i \\ (\varepsilon,\delta)=j}}^{12} \left(S_{\varepsilon\beta\alpha p\gamma\delta q} A^{(\varepsilon,\beta)p}{}_{hk} A^{(\varepsilon,\gamma)qhl} u_{\triangleright}^{(\varepsilon,\delta)l} \right) = \tag{1}$$

$$\sum_{i}^{N} v_{\triangleright}^{ik} \sum_{\varepsilon}^{n} \sum_{\substack{\alpha\,\ni: \\ (\varepsilon,\alpha)=i}}^{12} \left(T_{\varepsilon\beta\alpha p\gamma} A^{(\varepsilon,\beta)p}{}_{hk} f_{\bowtie}^{(\varepsilon,\gamma)h} - S_{\varepsilon\beta\alpha p\gamma\delta q} A^{(\varepsilon,\beta)p}{}_{hk} A^{(\varepsilon,\gamma)qhl} u_{\triangleleft}^{(\varepsilon,\delta)l} \right)$$

into this:

$$K_{ikjl} u_{\triangleright}^{jl} = F_{ik} \ .$$

The tensor K_{ikjl} is symmetric over indices i and j because the integral between nodes (i,j) must be identical to the integral between nodes (j,i). Therefore, to avoid redundant work, we iterate on each element over node indices α and δ in the following way:

i	j										
1											0
2										1	0
3									2	1	0
⋮						⋮					
10		9	8	7	6	5	4	3	2	1	0
11	10	9	8	7	6	5	4	3	2	1	0

and add the same value to the pair (i,j) and its symmetric pair (j,i). Then iterate over all repeated indices: (0,0), (1,1), ..., (10,10), (11,11) and add each value only once.