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):

$$
\begin{array}{llll}
\mathbf{u}_{\triangleright} & u_{\triangleright}^{jl} & \textsf{Tensor Uf} & \texttt{free node vars}\ , \\
\mathbf{u}_{\triangleleft} & u_{\triangleleft}^{jl} & \textsf{Tensor Uc} & \texttt{constrained node vars}\ .
\end{array}
$$

where the slots represent:

$$
\begin{array}{lll}
j & \rightarrow & N\ \texttt{nodes}, \\
l & \rightarrow & m\ \texttt{variables}.
\end{array}
$$

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`):

$$
\begin{array}{llll}
\mathbf{f}_{\bowtie} & f_{\bowtie}^{jl} & \textsf{Tensor F} & \texttt{forcing vars}\ .
\end{array}
$$

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:

$$
\begin{array}{llll}
\mathbf{A} & A^{iphl} & \textsf{Tensor A} & \texttt{stiffness matrix}\ ,
\end{array}
$$

where the slots represent:

$$
\begin{array}{lll}
i & \rightarrow & N\ \texttt{nodes}, \\
p & \rightarrow & 3\ \texttt{partials}, \\
h & \rightarrow & m\ \texttt{equations}, \\
l & \rightarrow & m\ \texttt{variables}\ .
\end{array}
$$

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:

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

In the assembly process we go over each element ε:

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

$$F_{ik} = \sum_{\varepsilon}^{n} \sum_{\substack{\alpha \ni: \\ (\varepsilon,\alpha)=i}}^{12} \left(T_{\varepsilon\beta\alpha p\gamma} A^{\overset{\varepsilon}{\beta}p}{}_{hk} f_{\bowtie}^{\overset{\varepsilon}{\gamma}h} - S_{\varepsilon\beta\alpha p\gamma\delta q} A^{\overset{\varepsilon}{\beta}p}{}_{hk} A^{\overset{\varepsilon}{\gamma}qhl} u_{\triangleleft}^{\overset{\varepsilon}{\delta}l} \right) \tag{2}$$

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

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

α	δ	
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

while going through all 3×3 combinations of derivatives. We 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) on the diagonal and add each value for all 3×3 combinations only once.