Algorithm 3 Creates $[H]_{\text{dyn}}, [G]_{\text{dyn}} \in \mathbb{C}^{(3m)\times(3n)}$

1: **procedure** GHDYN_KERNEL(Hstadiag, Gstadiag)
2: $j :=$ blockIdx.x; $i :=$ blockIdx.y; $x :=$ threadIdx.x; $y :=$ threadIdx.y
3: $l :=$ laneId; $w :=$ warpId; $numWarps = \lceil g^2/32 \rceil$
4: $Hlocal \leftarrow$ GenerateMatrixHdyn(i, j, x, y) ▷ Return 3×3 matrix
5: $Glocal \leftarrow$ GenerateMatrixGdyn(i, j, x, y)
6: **if** $i = j$ **then** ▷ Singularity
7: OvercomeSingularity(*Hbuffer*, *Gbuffer*)
8: `shfl_down`(*Hlocal*) ▷ Reduces matrices in the same warp
9: `shfl_down`(*Glocal*)
10: **if** $l = 0$ **then**
11: *Hbuffer*[warpId] = *Hlocal*
12: *Gbuffer*[warpId] = *Glocal*
13: **if** $w = 0$ and $l < 9$ **then**
14: $v = l\%3$; $u = l/3$;
15: $Helem[u][v] \leftarrow$ `thrust::reduce`$(Hbuffer[u][v][0], Hbuffer[u][v][numWarps])$
16: **if** $i = j$ **then**
17: $Helem[u][v] \leftarrow Helem[u][v] + Hstadiag[i][u][v]$
18: $H[i][j] \leftarrow Helem$ ▷ $H[i][j]$ is a 3×3 matrix
19: **else if** $w = 1$ and $l < 9$ **then**
20: Repeat code from lines 14-18, but for G, *Gbuffer*, *Gelem* and *Gstadiag*
21: **procedure** GHDYN_ASSEMBLY
22: Move data to GPU memory
23: Allocate H and $G \in \mathbb{C}^{(3m)\times(3n)}$ in GPU memory
24: Allocate *Hbuffer*, *Gbuffer* in GPU shared memory, buffer of matrices 3×3 of size g^2
25: Run `GHdyn_kernel` with $m \times n$ blocks and $g \times g$ threads. Await for return
26: Reorder matrices columns regarding the boundary conditions
27: Keep H, G in GPU memory, if enough memory. Else retrieve both matrices