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

procedure GHdyn_nonsing_kernel
  $j :=$ blockIdx.x; $i :=$ blockIdx.y
  $x :=$ threadIdx.x; $y :=$ threadIdx.y
  if $i \neq j$ **then** ▷ Singularity case will be computed by CPU
    $Hbuffer[x][y] \leftarrow$ GenerateMatrixHdyn(i, j, x, y)
    $Gbuffer[x][y] \leftarrow$ GenerateMatrixGdyn(i, j, x, y)
    if $x < 3$ and $y < 3$ **then**
      $Helem[x][y] \leftarrow$ `thrust::reduce`$(Hbuffer[x][y][0], Hbuffer[x][y][g^2])$
      $H[i][j] \leftarrow Helem$ ▷ $H[i][j]$ is a 3×3 matrix
    else if $g - 3 \leq x \leq g$ and $g - 3 \leq y \leq g$ **then** ▷ Make sure this runs in another warp
      $u := g - 3$; $v := g - 3$
      $Gelem[u][v] \leftarrow$ `thrust::reduce`$(Gbuffer[u][v][0], Gbuffer[u][v][g^2])$
      $G[i][j] \leftarrow Gelem$ ▷ $G[i][j]$ is a 3×3 matrix

procedure GHdyn_singular
  #pragma omp parallel for
  for $i := 1, m$ **do**
    $G_{\text{elem}} \leftarrow 0$
    $H_{\text{elem}} \leftarrow 0$
    for $y := 1, g$ **do**
      for $x := 1, g$ **do**
        $G_{\text{elem}} \leftarrow G_{\text{elem}} +$ GenerateMatrixGsing(i, x, y)
        $H_{\text{elem}} \leftarrow H_{\text{elem}} +$ GenerateMatrixHsing(i, x, y)
    $Gdiag[i] \leftarrow G_{\text{elem}}$
    $Hdiag[i] \leftarrow H_{\text{elem}}$

procedure GHdyn_assembly
  Allocate $Hdiag, Gdiag$ in Host memory
  Run GHdyn_singular$(Hdiag, Gdiag)$ in another CPU thread
  Move data to GPU memory
  Allocate H and $G \in \mathbb{C}^{(3m)\times(3n)}$ in GPU memory
  Allocate *Hbuffer*, *Gbuffer* in GPU shared memory, buffer of matrices 3×3 of size g^2
  Run `GHdyn_kernel` with $m \times n$ blocks and $g \times g$ threads.
  Retrieve H, G from GPU memory.
  Retrieve $Hdiag, Gdiag$ from `GHdyn_singular` when completed.
  for $i := 1, m$ **do**
    $G[i][i] \leftarrow Gdiag[i]$
    $H[i][i] \leftarrow Hdiag[i]$