"Rotation" is one of the main kernels in our computation. It involves a series of matrix-matrix multiplications. We're interested in both single and double precision computations.

The input matrices are $A^{i,0}(i = 0, \ldots, p)$, and B. The output matrix is C. Here, p is a "mesh-size" per red blood cell (RBC), an N is the number of RBCs. The matrices $A^{i,0}$ are precomputed and known, while the matrix B stores the position of points on RBCs and changes at every time step.

Let $M := 2p(p+1)$. Then each $A^{i,0} \in \mathbb{R}^{M \times M}$, and $B, C \in \mathbb{R}^{M \times N}$.

The pseudocode below summarizes the algorithm:

---

for $i = 0$ to p **do**
  for $j = 0$ to $2p - 1$ **do**
    $A^{i,j} \leftarrow$ permute $A^{i,0}$
    $C \leftarrow A^{i,j} B$
    Process C
  end for
end for

In this double loop, there is a permutation step, in which $A^{i,0}$ is permuted to $A^{i,j}$ and then it is applied to B. This permutation step involves exchanging columns of A^i. For any fixed j $(j = 0, \ldots, 2p - 1)$ it is defined as

$$A^{i,j}(n, m) = A^{i,0}\left(n, \left\lfloor \frac{m}{2p} \right\rfloor + (m \mod 2p - j)\right), \forall n, m. \tag{1}$$

Here we're using Matlab's index notation. Also, $m \mod 2p - j$ must be positive between 0 and $2p - 1$. When it is negative it is shifted by $2p$.

In our current implementation, we loop sequentially over i and j, we form $A^{i,j}$ and then we multiply with B calling GEMM.

Possible optimizations include exploring parallelism in the j-index and using equation (1) in a matrix-free matrix-matrix multiplication instead of explicitly forming $A^{i,j}$. In addition, there is parallelism in i but the memory costs are extensive. Typical values of N and p are 10–1000 and 6–20 respectively.