One-Dimensional Blood Flow Solver

self

in preparation

Array management in the simulation

The primary data arrays that are being updated dynamically in the simulation have been named as follows:

- `solA, solQ` - main solution arrays that store `nsteps` rows and `nodes` columns of solutions for A and Q for total duration of one cardiac cycle
- `solA_m, solQ_m` - arrays that store `nsteps` rows of data for the solution at the end point X(`nnodes`) in 2 columns (column 1 is correspond to the solutions from the previous cardiac poeriod, and column 2 from the current cardiac period).
- `solA_mm1, solQ_mm1` - same as above, except that they store the solutions at the last interior node i.e. X(`nnodes` $-$ 1)
- `solA_mphalf, solQ_mphalf` - same as above, except that they store the solutions at the node X(`nnodes`+ 1/2).

The total simulation time is taken to be an integer multiple of cardiac period `numPeriods` $\times$ `Tcard`. Thereafter, each cardiac period is assumed to be divided into `nsteps` divisions. For iterative solution of the outflow boundary condition terms, the solution arrays need to be correctly updated so that appropriate components of solutions from previous and current time-steps are always tracked. The implementation of this in form of a pseudocode is presented below:

Algorithm 1 Array management in loops

for (*number of periods*) **do**
  `solA_m(:,1)` ← `solA_m(:,2)`
  `solQ_m(:,1)` ← `solQ_m(:,2)`
  `solA_mm1(:,1)` ← `solA_mm1(:,2)`
  `solQ_mm1(:,1)` ← `solQ_mm1(:,2)`
  `solA_mphalf(:,1)` ← `solA_mphalf(:,2)`
  `solQ_mphalf(:,1)` ← `solQ_mphalf(:,2)`
  for (*number of time-steps*) **do**
    apply boundary condition to inflow node
    update `Avec(1)`, `Qvec(1)`
    for (*number of interior nodes*) **do**
      update all interior node solutions
      `Avec(node)` ← updated solution for A
      `Qvec(node)` ← updated solution for Q
    end for
    apply boundary condition to outflow node
    update `Avec(end)`, `Qvec(end)`
    `solA_m(time-step,2)` ← A_m^{n+1}
    `solQ_m(time-step,2)` ← Q_m^{n+1}
    `solA_mm1(time-step,2)` ← `Avec(end-1)`
    `solQ_mm1(time-step,2)` ← `Qvec(end-1)`
    `solA_mphalf(time-step,2)` ← $A_{m+1/2}^{n+1/2}$ from converged iteration solutions
    `solQ_mphalf(time-step,2)` ← $Q_{m+1/2}^{n+1/2}$ from converged iteration solutions
  end for
  `solA(time-step,:)` ← `Avec`
  `solQ(time-step,:)` ← `Qvec`
end for