Ryan Skinner

Improving Boundary Condition Stability in PHASTA

1 Introduction

2 Initial Outline of PHASTA

PHASTA begins execution at `main`, located in `phSolver/[in]compressible`, depending on which branch is desired. This function initializes MPI, and then calls `phasta`, located in `/phSolver/common`. Here, inputs are read and computed in `input`, and then the solver is run by calling `proces`, a Fortran routine. Within `proces`, `gendat` generates geometry and BC data.

Routines followed by an asterisk (*) are outlined in further detail separately.

INCOMPRESSIBLE ONLY, and we ignore cardiovascular impedance and RCR boundary stuff.

- `main`
  - initialize MPI
  - `phasta`
    - initialize PETSc
    - set input data paths
    - `input` — populate data structures with problem set-up and solver parameters
      - `readnblk` — read and blocks data
        - read `numstart.dat` and finds appropriate `restart.dat` files
        - read geometry from Posix or SyncIO files using `phio_readheader`
        - calculate maximum number of boundary element nodes
        - initialize constants like `ndof`, `ndofBC`, `ndiBCB`, and `ndBCB`
        - `genblk` — read and block connectivity
        - read BC mapping array into `nBC`
        - read temporary boundary condition code into `iBCtmp`
        - read BC data into `BCinp`
        - read periodic BC data into `iperread`
        - `genbkb` — generate boundary element blocks and traces for gather/scatter operations
        - read restart data into diffusive flux vector `qold`, primitive variables `uold`, and accelerations `acold`
      - echo global information
      - assert valid input constants (e.g. `icoord`, `navier`, `iexec`) defined in `common.h`
      - echo solver and integration information
      - `genint` — generate integration information
      - estimate number of nonzero globals
      - compute fluid thermodynamic properties
    - `proces` — generate problem data and calls the solution driver

        - `gendat` — generate geometry and BC data
            - `getshp` — generate the interior nodal mapping
            - `geniBC` — generate boundary condition codes
            - `genBC` — generate the essential boundary conditions
            - work with Dirichlet-to-Neumann BCs (?)
            - `genshpb` — generate boundary element shape functions
            - `genini` — read initial values in primitive ($\underline{U}$) form, satisfies BCs, and converts to $\underline{Y}$ form, filling the `y` vector
        - `setper` and `perprep` — store inverse of sum of one and number of slaves in `rcount`
        - LES-specific routines `keeplhsG` and `setrls` called as needed
        - `initStats` — allocate arrays to store flow statistics
        - RANS-specific routine `initTurb`
        - cardiovascular-specific routine `initSponge`
        - adjust BCs to interpolate from file `inlet.dat`, if it exists
        - set up eddy-viscosity ramp specific to NGC/Duct case
        - `itrdrv`*— iterate the discrete solution using the predictor multi-corrector algorithm
    - finalize PETSc
- finalize MPI

Numerical solution of the time-integrated unsteady Navier-Stokes equations occurs within `itrdrv`. Working arrays are listed in

- `itrdrv`
    - `initTimeSeries` — initialize time series collection to `varts.*.dat` files using `xyzts.dat` input
    - initialize `istep` and `ifuncs(:)` to zero, and set `yold = y` and `acold = ac`
    - `initEQS` — initialize equation solver (look into this later *?)
    - `do itsq = 1, ntseq` — main loop over time sequences
        - set `itseq = itsq`
        - set iteration-specific variables for `nstep`, `niter`, `loctim`, and `deltol`
        - `itrsetup` — set up time integration parameters
            - calculate α_m, α_f, and γ as functions of ρ_∞
            - set Jacobian type (appears to be for a fringe case only?)
            - mess with `ipred` and `y` for the same-delta predictor if we're on the first sequence
            - set global time increment inverse `Dtgl` and CFL data `CFLfl`
        - calculate number of flow solves per step, store in `nitr`
        - `do istp = 1, nstp` — main loop over time steps
            - `asbwmod` — set traction BCs if turbulence wall model is set (`itwmod`)
            - `itrPredict`*— predict solution at time $n+1$
            - `itrBC`*— satisfy BCs on the $\underline{Y}$-variables; returns a modified `y`
            - `itrBCSclr`*— satisfy BCs on the scalar `isclr`; returns a modified `y`
            - ...
            - ...
            - ...
        - deallocate variables and close files
    - deallocate variables and close files

Symbol	Dimension	Description
`nshg`		# global shape functions
`ndof`		# number of degrees of freedom
`npro`		# elements, indexed by e
`nshl`		# nodes per element, indexed by a
`ntseq`		# time sequences (?)
`nstep`		# time steps requested for current run
`lstep`		current time step
`lstep0`		first time step solved by current run, initialized to `lstep+1`
`istep`		step number relative to start of run
`iter`		iteration number
`niter`	`(MAXTS)`	# multi-corrector iterations per time step
`loctim`	`(MAXTS)`	local time stepping flag (?)
`deltol`	`(MAXTS, 2)`	velocity and pressure delta ratios
`impl`	`(MAXTS)`	heat, flow, and scalar solver flags (1's, 10's and 100's places)
`iturb`		indicates which turbulence model to use
`ifunc`		function evaluation counter, `niter*(lstep-lstep0)+iter`
`ifuncs`	`(6)`	function evaluation counter (?)
`y`	`(nshg, ndof)`	$\underline{Y}$ variables
`x`	`(nshg, nsd)`	node coordinates
`iBC`	`(nshg)`	BC codes
`BC`	`(nshg, ndofBC)`	BC constraint parameters
`shp`	`(nshape, ngauss)`	element shape functions at Gauss points (interior)
`shb`	`(nshapeb, ngaussb)`	element shape functions at Gauss points (boundary)
`shgl`	`(nsd, nshape, ngauss)`	local shape function gradients at Gauss points (interior)
`shglb`	`(nsd, nshapeb, nguassb)`	local shape function gradients at Gauss points (boundary)
`iper`	`(nshg)`	periodicity table

3 Life's Persistent PHASTA Questions

- Is `qold`, allocated in `readnblk.f` ever deallocated? Can't find it.
- Why do most of the time step parameters have dimension `MAXTS`?
  - It also seems that some parameters are indexed by `itseq`, but don't change from step to step.