All important interface functions are listed in “main_cpp.cpp”.

vega_FEM_initiate_cpp(void)

// Purpose: Initiate the Vega FEM for solid Finite element simulation.

initConfigurations(); // parse the “mesh.veg”, “mesh.bou”, “vega.config”, set parameters.

initSimulation(); // initiate the FEM simulation.

readFlowGridXYZ(); // read flow filed grid for pressure & velocity interpolation Index&Ratio.

readSurfaceTriangleMesh(); // read the triangle mesh information of the membrane.

vega_interpolateIndexRatio_cpp(int *markerInterpolateIndex_,double *markerInterpolateRatio_)

obtainBodyNormalVectorCW(); // the unit normal vector at each vertices on the membrane surface.

obtainMarkerInterpolateIndexRatio(markerInterpolateIndex_,markerInterpolateRatio_); // calculate the positions of points that has prescribed distance with the vertices of membrane and also are located on the normal vector line respectively. The positions are compared with flow field grids and return the x,y,z index and ratio, saved in the **int markerInterpolateIndex[] & double markerInterpolateRatio[].**

vega_deformation_cpp(double *markerPressure_,double *markerInterpolateVelocity_,double *bodyMotion_)

obtainMarkerForceFromPressureAndVelocity(markerPressure_,markerInterpolateVelocity_,bodyMotion_); // calculate the pressure induced force and viscos shear flow induced force according to the pressure and velocity of the pointers that are vertical to the membrane surface, and the velocity of the vertices on membrane.

integratorBase->SetqState(u_base,uvel_base,uaccel_base); // Set the FEM solver with the deformation displacement, velocity & accelerate velocity of the Nth time step.

integratorBase->SetInternalForces(f_int_base); // Set the FEM solver with the internal force of the Nth time step.

……. // copy the marker force to the array f_ext[] with multiplying forceRatio.

filterForce(bodyType,f_ext); // filter the f_ext[], remove the huge peak values for algorithm stability.

integratorBase->SetExternalForces(f_ext); // Set the FEM solver with the external force f_ext[].

integratorBase->DoTimestep(); // FEM solver do time step from Nth to (N+1)th.

integratorBase->GetqState(u,uvel,uaccel); // get the deformation displacement, velocity & accelerate velocity of the (N+1)th time step.

…… // calculate the deformation velocity from Nth time to (N+1)th time step, and return to IBM solver.

vega_reNewBodyPosition_cpp(void) // after body velocity converged in the whole mixed solver

integratorBase->GetqState(u_base,uvel_base,uaccel_base); // get the deformation information of N+1)th time, save them into u_base[], uvel_base[], uaccel_base[].

integratorBase->GetInternalForces(f_int_base); // the similar purpose as the upper function.

timestepCounter++; //