Boundaries and subroutines [2020-06-02]

1. Removed var TEMP_UN. Modified FIND_LAPLAC and U_BOUNDARY and REMESH

Attempting

- Find all the subroutines where something is done to boundary nodes

List of Work

- [x] Disable *boundary_vel*
- [x] Added remesh frequency as user input REMESHFREQ. 0 to disable.
- [x] Probable bug of 3D gradP calculation in GRADIENT_POM_SHA
- [ ] Probably disable GRADIENT_2P for type8 nodes
- [x] Remove TEMP_UN for type8 in U_BOUNDARY
- [ ] Collision for wall nodes
- [ ] Initialsing large temporary matrices
- [x] Remove or edit _FIND_LAPLAC_

Flag definitions

Particle IDs - NODEID

Particle ID	Type
0	Inner fluid
4	Free-surface
8	Left wall - wavemaker
2	Bottom wall
3	Right wall
1	Near sidewall
7	Far sidewall
9	Cylinder wall
-6	Ghost nodes, except cylinder ghost nodes
-9	Cylinder ghost nodes

Note: Never make a NODEID(I) = -2
In *JUDGEFREESURFACE()*, in the first pass for type0 and type4 particles using the VLIM algorithm, if we find a type0 particle which is suspected of being a FS node, then we mark it as NODEID(I) = -2 to identify it in the second pass which i done for the confirmation algorithm. So never make a particle id -2.

Wall ID - NWALLID

Variable	Value	Meaning
NWALLID(:,1)	0	Appears to be wall node with 0 velocity. Used in *U_BOUNDARY*. Assigned to bottom corner of type3 in CYLIND3,4.
NWALLID(:,1)	1	Used in FIND_LAPLAC(). Identifies as XY plane point. Assigned to type2 in CYLINDs
NWALLID(:,1)	2	Used in FIND_LAPLAC(). Identifies as YZ plane point. Assigned to type8 and type3 nodes in CYLINDs.
NWALLID(:,1)	3	Used in FIND_LAPLAC(). Identifies as XZ plane point. Assigned to type1 and type7 in CYLINDs
NWALLID(:,1)	4	Sets pressure = 0 for **non-wall** nodes using Dirichlet BC in *ASSEMATRIX_MLPG_SHA* In *FILL_MATRIX_SHA* it does no operation on A and B of AX=B Sets pressure = 0 in *PRESSURE_SMOOTH_SHA*, if the smoothened pressure is negative on this node. It is assigned to type9 nodes in CYLIND3. Effectively that only matters for type9 nodes in the pressure smooth function, coz in assemble matrix the condition is only used for non-wall nodes. Anyway disabled this in CYLIND4,5 for now. It is also used in _U_BOUNDARY()_, where all vels for this node are put 0.
NWALLID(:,1)	5	There is a condition in _GRADIENT_2P_ where it will not update the minimum pressure loop for these nodes in gradP for near and far sidewall node if this value is set. It is actually assigned to the bottom layer of the near and far sidewall nodes. It has no real significance.
NWALLID(:,1)	6	
NWALLID(:,1)	7	It is assigned to wall nodes at the intersection of two wall surfaces. Used in *WEIGHTF_GRAD_2P*. Here the loop runs either if the node and its neighbour are of same type, or if the neighbour has nwallid(:,1)=7
NWALLID(:,2)	1	It is set for all bnd nodes under the waterline and decides if the node will be used for gradP calculation. If this is set to = 1 then the bnd node is in water, otherwise its set to -10 where its outside water. For all bnd nodes except cylinder this value in set in *CYLIND3*. For cylinder nodes its set in *SRI2* based of if the cylinder node is under the waterline. Used inside *GRAD_PCYL_SHA*. If the value is not = 1 then gradient of P is not calculated at that cylinder node In *GHOSTPART*, where the pressure of ghost node is calculated, the value as 1 decides that the ghost node will be used in gradient calculation of pressure.
NWALLID(:,2)	-11	Sets pressure to 0 using Dirichlet BC inside *ASSEMATRIX_MLPG_SHA* wall nodes loop and *PRESSURE_SMOOTH_SHA*. Done for FS nodes on boundaries.

Variable	Value	Meaning
NWALLID(:,2)	-10	Does a lot of functions for ghost nodes. In Remeshing, it avoids using ghost nodes during remeshing. In *ASSEMATRIX_MLPG_SHA* it sets pressure = 0 using Dirichlet BC at ghost nodes. In all *WEIGHT* functions it avoid using ghost nodes. In *FILL_MATRIX_SHA* it does no operation on A and B of AX=B Sets pressure = 0 in *PRESSURE_SMOOTH_SHA* In gradient calculation a value of -10 will ensure the ghost node or fluid node is not used in gradP calculation. If its set to 1 for ghost node, then the node will be used for calculation of gradient. In *GHOSTPART* and *GHOSTPART_V* it determines if the ghost node associated with this cylinder node will be used in gradP or pressure smooth or velocity smooth. In *Output* it places the dry nodes on cylinder (and their respective ghost nodes) away so that only wet nodes (and their respective ghost nodes) are shown in Paraview. In *U_BOUNDARY* it sets the velocity of ghost wall nodes 0
NWALLID(:,3)	9	Pressure calculation done using MLS interpolation of nearby nodes inside *ASSEMATRIX_MLPG_SHA* wall nodes loop In *FILL_MATRIX_SHA* it does no operation on A and B of AX=B
NWALLID(:,4)	3	No operation set for this. Its only used in *ASSEMATRIX_MLPG_SHA*

Additionally note that *FIND_LAPLAC_ used a lot of conditions for NWALLID(:,1). As the subroutine isnt used I am not documenting this.*
***Turns out _FIND_LAPLAC* was the main reason behind spoiling the initial attempt at weak coupling with Bsnq. Check its function** here

Storing Bnd nodes

- BNDNP Total number of boundary and ghost nodes
- BNDXY(BNDNP, 3)
  - All bnd and ghost nodes
- BNDFIX(3, 0:BNDNP)
  - **Assuming tank bottom is at 0**
  - Bndnodes along the bottom
    - BNDFIX(1,1:BNDFIX(1,0))=(/ BOTTOM POI /)
    - BNDFIX(2,1:BNDFIX(2,0))=(/ NEAR SIDEWALL /)
    - BNDFIX(3,1:BNDFIX(3,0))=(/ FAR SIDEWALL /)
- BNDFS(0:BNDNP)
  - Free surface nodes for sidewall boundary from the initial mesh.

FNPT coupling related variables

- TEMP_UN
  - For storing normal velocity for the wavemaker
    However its mostly useless, cos the normal of the type8 particles keeps changing and irrelevant.

- PRESS_DR
    - Drichlet BC for pressure for type8
- FP is the object of FNPTCPLTYP
    - Is used to read the FNPT file and interpolate the values where required.

NODELINK_3_SHA()

- The radius is different for fluid and wall particles

GENERATEGHOST()

- Run from NODEID(-2)+1 to NODEID(-7) only, i.e. all wall particles except for type9
- It generates ghost particle for all wall particles (except type9) along the outward unit normal
- For type8, the outward unit normal will only be based on the initial shape of the type8 surface.
    - Hence the outward unit normal will not be correct when the type8 surface bends with the fluid movement.
    - However this may be a good thing as the host nodes there are only needed for FS detection and it works will if they remain along the normal to the original type8 surface.

SRI2()

- Only for type9 bnd particles
- Identifies the waterline along the type9 surface.

u* Calculation

- It appears that for all particles viscosity is assumed to be 0.

JUDGEFREESURFACE_SHA()

- It is only run for nodes till NODEIF(-2), i.e. fluid and FS nodes.
- The walls are assumed to have same FS particles as initial mesh for all time steps, except type9 wall
- For type9 wall *SRI2()* is used to find the waterline.

FIND_LAPLAC() [2020-06-05]

- This subroutine is doing two functions
    i. Picking up the viscous component for wall nodes from the nearest water node.
    ii. **Remove the normal component of U* vector**

The second function mentioned there is quite important actually.

- In intermediate velocity calculation we do
  u* = u(n-1) + viscosity term
- The u(n-1) already has the normal velocity BC applied in the previous time step.
- However the vicosity term may introduce additional normal velocity components.
- This function calculates the intermediate velocity and then assuming that the viscosity term is applied, it removes the normal component of u*
  - **In Bsnq weak coupling that was causing a issue!**
  - In Bsnq coupling pressure Dirichlet BC is not applied due to the reasons discussed in *BQML* (refer log_wbqml_v0001.md) and instead MLS extrapolation is applied.
  - So this purely relies on the knowledge of velocity gradient in the vicinty of the relaxation zone to calculate the pressure.
  - Now probably this function was doing the right thing by removing the normal component of u* on *walls*, not 100% sure though.
    - **Is BC supposed to be applied on u* ?**
  - However because the walls in all our cases till now are non-viscous, it was not necessary, because for every wall node u* = u(n-1)
  - Moreover in the Bsnq coupling the type8 node is no longer a wall so this BC of u_normal = 0 is not even applied.
  - Therefore the artifical application of u* . n = 0 was causing the pressure to come wrong.
    - This issue did not show up in FNPT coupling as we were directly using the pressure from FNPT.
    - Hence a lot of things thankfully worked in FNPT purely because we had velocities and pressures!

FIND_ACCN()

- This function is run only for wall nodes.
- It gives the u_normal along outward unit normal
- For type8, it used TANK_A as a known and calculated the u_normal
- Its not used anywhere.
- **Can remove this subroutine**
- **This subroutine is required for moving or floating body**

ASSEMATRIX_MLPG_SHA()

- It is run for wall nodes, i.e. NODEID(-2)+1 to NODEID(-1)
  - Applied ∇P . n = 0 for all wall nodes, here n is the outward unit normal, including for type8.
  - It relies on outward unit normal, hence is incorrect for type8 which moves like internal fluid in this code due to coupling
    - Anyway for FNPT coupling I reset the SKK for type8 to apply Dirichlet BC using values for FNPT within this subroutine itself.
    - **Cannot do that in Bsnq as the pressure is not known accurately.**
      - **Maybe I can instead using just MLS interpolation for all type8 nodes for Bsnq.**
      - That seems most correct approach comapred to doing ∇P . n = 0 where n is unknown as unnecessary; or Dirichlet BC for pressure as pressure is unknown.

    - Earlier when a piston type wavemaker was used maybe ∇P . n = 0 would have been of some use as n did not have to update.
        - But in FNPT I reset it to Dirichlet anyay so that part is useless
        - For Bsnq I cannot use this apprahc anyway.
- It is also run for FS nodes, i.e. type4

GHOSTPART()

- Applied only to cylinder ghost nodes
- Calculated the mirror pressure on the ghost node of type-9

GRADIENT_POM_SHA()

- Applied to all real nodes, i.e. till NODEID(-1)
    - This is probably a mistake.
        - **Not a mistake**
        - This function does 3D gradient for all nodes (till NODEID(-1)).
        - GRADIENT_2P then updates for wall nodes, where it keep the gradient normal to the wall from this 3D gradient function itself, but updates the in plane gradient only using the in plane nodes.
    - I have checked from the oldest versions of that code that this was done in the original code.
    - Although as the motion is restricted in the normal direction for the wall particles, the calculation of along normal gradient isnt too big an issue.

GRAD_PCYL_SHA(()

- Calculated gradient of pressure using the mirror nodes and fluid nodes for cylinder node type9

GRADIENT_2P()

- Calculated gradient in 2D only using particles of the same kind.
    - Inside *WEIGHTF_GRAD_2P*, it is checked that neighbour NODEID is same as the particle under consideration's NODEID
- Executed for all wall nodes
- The 3D gradient is calculated for all nodes (till NODEID(-1)) in *GRADIENT_POM_SHA()*.
- This function then retains the normal gradient from the *GRADIENT_POM_SHA()*.
- However the in plane gradients are recalculated only using the in plane nodes.
- Separate condition to do YZ plane for type8 and type3
- Separate condition to do XZ plane for type1 and type7
- **CAUTION :** Using for type8 too
    - As the nodes of same type only are used in gradient calculation, it is possible that for type8 the gradient may fail once the type surface become too skewed due to the lagrangian motion.

U_UPDATE_POW2()

- Applied to all real nodes, i.e. till NODEID(-1)
- Does `u(n+1) = u(n) - gradP*dt/rhoW`

U_BOUNDARY()

- Applied only to wall nodes, i.e. NODEID(-2)+1 to NODEID(-1)
- It makes the normal component of velocity = 0 for all wall nodes, except type8
- Here TEMP_UN is used to get the T(n+1) velocity along the initial outward unit normal from FNPT.
  - The entire implementation is quite wrong is frankly saved by the fact that later FNPT velocity and pressure are just forced onto these nodes.
  - Not correcting this mistake for now as it'll take time and doesnt make any difference to the solution.
- **The TEMP_UN is not required to be done in Bsnq**
- Made all vels 0 for type2
- Also for type8 and type2 the following is implemented for some reason. Removed for type2. **Note that this condition will set 0 for UY for type8 which wont work in 3D wave**

```
IF(NODEID(INOD).EQ.8.or.nodeid(inod).eq.2)THEN
  UY(INOD,IK) = 0.D0
  UY(INOD,IK1) = 0.D0
ENDIF
```

COLLISION_3D_SHA()

- So far its run only for fluid nodes, i.e. till NODEID(-2)
- However it might need to made to work for boundary nodes
- **Especially for Bsnq coupling on all side** it may be needed to enable collision for all wall nodes too which is going to be tricky.

GHOSTPART_V()

- Applied to ghost nodes of cylinder partciles only, i.e. type-9.
- Mirror the velocity at type-9 nodes.

VELOCITY_SMOOTH_SHA()

- Smoothing of velocity done for all real nodes, i.e. till NODEID(-1)
- **NOTE :** There is weight of the actual point velocity and the smoother point velocity.
  - vel = (0.7 x velAct) + (0.3 x velSmooth)
- **BUG :**
  - In this subroutine and many others like *PRESSURE_SMOOTH_SHA*, I am initialising quite large matrices.
  - And these are not allocatable type matrices.
  - Sometimes the sizes are huge

- For example the following from this function (NODN = NODEID(0))

```
REAL(KIND=8)::PHI(NLMAX),W(NLMAX),B(MBA,NLMAX)
REAL(KIND=8)::A(MBA,MBA),AINV(MBA,MBA),PT(MBA)
REAL(KIND=8)::PB2(MBA),PP2(MBA,MBA)
REAL(KIND=8)::XQ,YQ,ZQ,AA(MBA,NLMAX),PINT,DEW,WWI
REAL(KIND=8)::RIAV,DR,DR2,GAMMA,GAMMA2,CCIAV,RCI,ETMP
REAL(KIND=8)::UXC1(NODN),UYC1(NODN),UZC1(NODN)
REAL(KIND=8)::UXC3(NODN),UYC3(NODN),UZC3(NODN)
```

UPDATE_CO()

- Applied to all real nodes, i.e. till NODEID(-1)
- Updates the coordinate using Euler explicit method.

boundary_vel()

- This subroutine was doing something with type-8, type-7, type-1, type-2. These do not exist anymore.
- **Disabling this subroutine**

NEWCOOR4()

- type8
  - Particle are readjusted vertically according to the T0 mesh
- type2, type1 and type7
  - All type2 $\Delta z = 0$
  - typ1 and type7 at bottom layer only $\Delta z = 0$
- **Note :** A lot of stuff here assumes unidirection and non-y-variational flow. **Will not work like this for Bsnq.**

FIXCYLINDER()

- Fixes the cylinder to a stationary position, based on the initial mesh.
- Applied only to type9

FINDCYLINFORCE()

- Calculates the force on cylinder by integration pressure using trapezoidal rule.
- **Assumes uniform cylinder nodes and known number of nodes along the lenght and radius.**

Removed var TEMP_UN. Modified FIND_LAPLAC and U_BOUNDARY and REMESH

- _FIND_LAPLAC_ subroutine is doing two functions
  i. Setting some normal velocities based on boundary type. This is quite confusing and seems unnecessary.
  ii. **Remove the normal component of U* vector**

The second function of this subroutine mentioned there is quite important actually.

- In intermediate velocity calculation we do
  u* = u(n-1) + viscosity term
- The u(n-1) already has the normal velocity BC applied in the previous time step.
- However the vicosity term may introduce additional normal velocity components.
- This function calculates the intermediate velocity and then assuming that the viscosity term is applied, it removes the normal component of u*
  - **In Bsnq weak coupling that was causing the issue as shown in above observations!**
  - In Bsnq coupling pressure Dirichlet BC is not applied due to the reasons mentioned within Approch 1 description and instead MLS extrapolation is applied.
  - So this purely relies on the knowledge of velocity gradient in the vicinty of the relaxation zone to calculate the pressure.
  - Now probably this function was doing the right thing by removing the normal component of u* on *walls*, not 100% sure though.
    - **Is BC supposed to be applied on u* ?**
  - However because the walls in all our cases till now are non-viscous, this function was not necessary, because for every wall node u* = u(n-1)
  - Moreover in the Bsnq coupling the type8 node is no longer a wall so this BC of u_normal = 0 is not even supposed to be applied.
  - Therefore the artifical application of u* . n = 0 was causing the pressure to come wrong and further causing particle crowding issues.
    - This issue did not show up in FNPT coupling as we were directly using the pressure from FNPT.
    - Hence a lot of things thankfully worked in FNPT purely because we had velocities and pressures! We did make a lot of mistakes there but those didnt matter.
  - **So I have made the changes of removing var TEMP_UN from *U_BOUNDARY()_ and _FIND_LAPLAC()* for type8 if I_WM = 15.**
  - The same description is written in log_wbqml01_v0001.md

Additionally, I have added a _U_BOUNDARY()_ call after REMESH to ensure that after the remeshing the required boundary conditions for the vel are applied. It was seen in some tests that this was causing issues.