Periodic Boundary Conditions

For periodic boundaries, the boundary is treated like a multi-block interface, with full grid connectivity defined. In addition, for rotational coupling, the velocity vector must be rotated before passing information from one face to the other. The information required for this transformation is provided in the PeriodicCoupling node. If the PeriodicCoupling node does not exist, the usual interface condition should be applied. If the PeriodicCoupling node exists, the code should use the information.

Parents: Base_t/Zone_t/ZoneGridConnectivity_t/GridConnectivity_t
Name: PeriodicCoupling
Label: DataArray_t
Dimensions: 1
Dimension Values: 7
Data: real: 1: periodic coupling mode
0 – no periodic coupling
1 – rotation type
2 - translation
Rotation (Data(1) = 1)
2: X location of Center of Rotation
3: Y location of Center of Rotation
4: Z location of Center of Rotation
5: Rotation of Boundary about X axis (radians)
6: Rotation of Boundary about Y axis (radians)
7: Rotation of Boundary about Z axis (radians)
Translation (Data(1) = 2)
2: X translation of boundary
3: Y translation of boundary
4: Z translation of boundary

Notes:

- If non-cartesian velocity components are used, X,Y,Z becomes r,phi,z or x,r,phi as appropriate
- If a code does not know about PeriodicCoupling, the node would be ignored, and the user code might run along happily until the user saw they were getting garbage. Does this imply a check that the mid-level lib should force people to use, or do we let the user beware?
- To Karl Engel – I propose holding off on phase lag, since I presume that implies you are saving the boundary information over a period of time, and the phase lag tells you how far back in time to go to get the information. Thus, we also need to define a way to store the time history for the boundary point. Please let me know if I am mis-interpreting what you had in mind…
- To Steve Feldman – I am having a hard time figuring out how much of your capability this covers. For example, since we keep the full interface bc information (which nodes talk to which) I think we have the information needed for your partial and cyclic information? However, there is no info here on changes across the zone boundary (other than velocity rotation) and so I presume we miss your pressure drop, flow rate and bulk temperature info. Can we add that easily?
- To Chris Rumsey – I agree that often the boundary conditions are often over-specified by the SIDS, but in some cases there is necessary information we need to store. I hope the above fully defines this capability, without placing undue burden on developers…

BCRotorStator

This boundary condition allows a code to perform some sort of averaging operation before passing information across a zone boundary. This is useful, for example, in turbomachinery calculations where zones are in different frames of reference (one rotating, one not).

Parents: Base_t/Zone_t/ZoneGridConnectivity_t/GridConnectivity_t
Name: BCRotorStator
Label: DataArray_t
Dimensions: 1
Dimension Values: 7
Data: real: 1: BCRotorStator Mode
- 0 – no rotor stator interaction – use regular coupling
- 1 – steady mode
- 2 – unsteady mode

Steady Mode (Data(1) = 1)
2: averaging mode
- 0 – no averaging done
- 1 – average along xi grid direction
- 2- average along eta grid direction
- 3- average along zeta grid direction
- 4 – average along xi and eta
- 5 – average eta and zeta
- 6 – average xi and zeta
- 7 – average xi, eta and zeta
- note: must ensure that RotatingCoordinates_t is checked for both zones and ensure that the coordinate transformations are performed properly before passing data

Unsteady Mode (Data(1) = 2)
2: what is needed here?

Notes:
- I am making this one up. Please let me know how much I missed…
- This only works for structured grids (use xi,eta,zeta). Should we define x,y,z and r,theta,phi, and x,r,phi averaging as well, and assume that the application code has figured out how to do the correct averaging?
- If a code does not know about BCRotorStator, the node would be ignored, and the user code might run along happily until the user saw they were getting garbage. Does this imply a check that the mid-level lib should force people to use, or do we let the user beware?