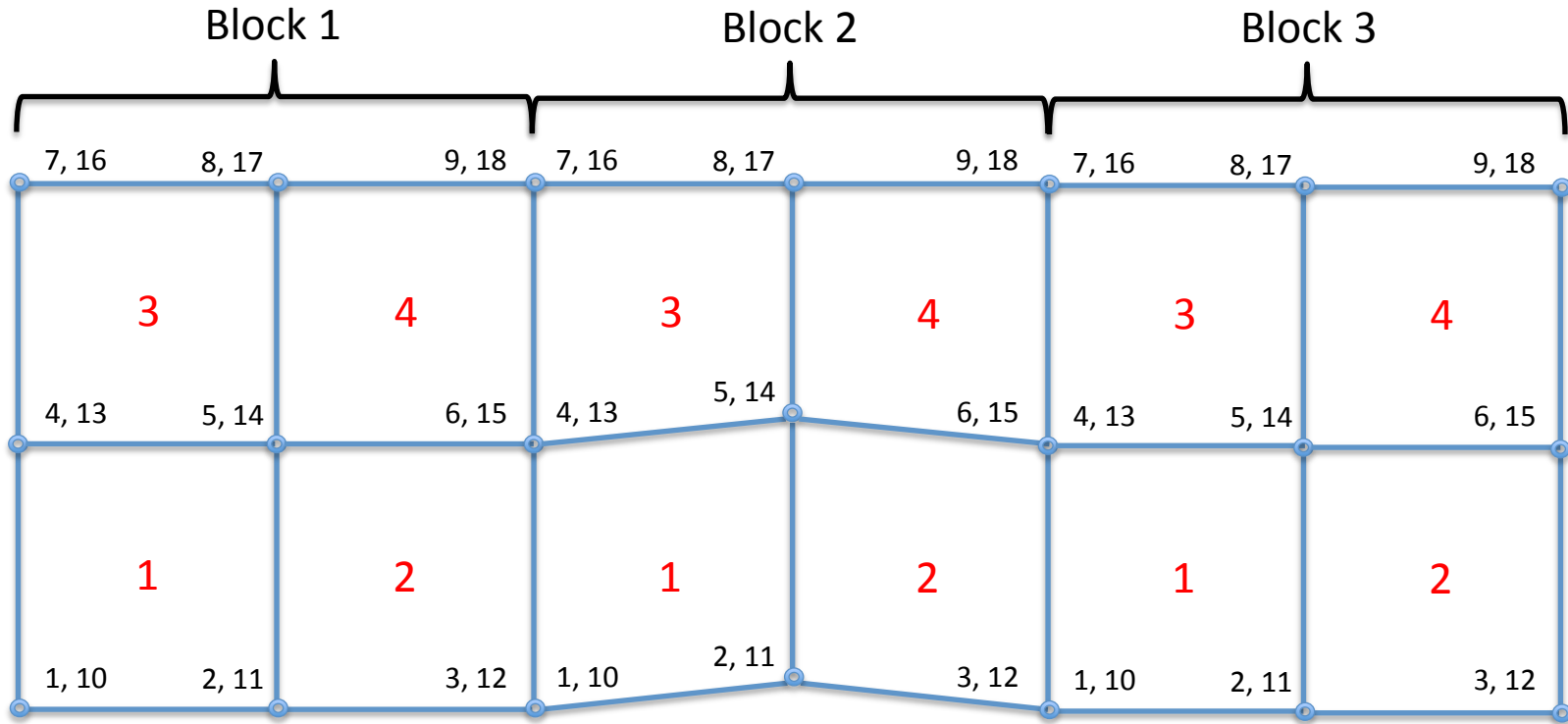


# Making CGNS a little more usable for multi-block unstructured grids

Micah Howard

Srini Arunajatesan

# Example multi-block unstructured grid



Including two optional pieces of information in an unstructured CGNS file makes it easier for the user...

- CGNS allows for this additional information to be included
- However, it doesn't seem to be standard practice for unstructured grids (based on what we've seen included in the CGNS file written out by a few different grid generators)
- It may be beneficial to have a "best practices" document that outlines what information is best to include for (multi-block) unstructured grids

# What you get with in an unstructured CGNS file currently...

- The following CGNS file was generated with Pointwise
  - Sorry to pick on Pointwise but it's what we primarily use and seems to have the best CGNS support for unstructured grids (that we know of at least)

# What you get with an unstructured CGNS file currently...

The screenshot shows the CGNSview interface. On the left is a 'Node Tree' with a hierarchical structure. The 'ElementConnectivity' node under 'blk-1' is circled in red. An arrow points from this node to the 'Node Description' panel on the right. In the 'Node Description' panel, the 'Node Name' is 'ElementConnectivity' and the 'Node Label' is 'DataArray\_1'. The 'Data Description' section shows 'Data Type' as 'i', 'Dimensions' as '8', and 'Bytes' as '32'. Below this, the 'Node Data' section contains two lines of integers: '16 13 4 7' and '13 10 1 4', with the second line circled in red. An arrow also points from this circled data to the explanatory text on the right.

Node Description

Parent Node |Base/blk-1/bc-inflow  
Node Name |ElementConnectivity  
Node Label |DataArray\_1

Link Description

Link File |  
Link Node |

Data Description

Data Type |i  
Dimensions |8  
Bytes |32

Node Data

16 13 4 7  
13 10 1 4

Face connectivity is given but it is very difficult to determine the parent element ID. This should be given.

# What you get with an unstructured CGNS file currently...

The screenshot shows the CGNSview interface. On the left is a 'Node Tree' with a hierarchical structure. Two nodes, '1to1 Connection1' and '1to1 Connection2', are circled in red. Each circled node contains sub-nodes: 'GridConnectivityType', 'GridLocation', and 'PointRange'. On the right, the 'Node Description' panel is active, showing details for a 'PointRange' node. The 'Parent Node' is '/Base/blk-1/ZoneGridConnectivity/1to1 Connection1'. The 'Node Name' is 'PointRange' and the 'Node Label' is 'IndexRange\_t'. The 'Link Description' section has empty fields for 'Link File' and 'Link Node'. The 'Data Description' section shows 'Data Type' as 'i4', 'Dimensions' as '1 2', and 'Bytes' as '8'. Below this are buttons for 'create', 'modify', 'read', 'clear', and 'delete'. The 'Node Data' section shows the values '19' and '20'. At the bottom, there is a status bar with 'Line 1 (1)' and 'Values:Line 1'.

Merging connectivities between blocks is more difficult than it needs to be.

One currently has to find the matching “GridConnectivity\_t” nodes between two shared blocks. Then look at the PointRanges and go to the “Elements\_t” nodes that matches those ranges. Then look at the face connectivities of those elements and start matching nodes.

Including two optional pieces of information in an unstructured CGNS file makes it easier for the user...

1. Adding parent element data to the boundary “Elements\_t” nodes in an unstructured block
  - This information provides both the “parent” element number and a means to get the connectivity of the face
  - Knowing the “parent” element number of a boundary face (without having to search for it) is very convenient for FE and cell-centered FV codes

This additional information will be shown on the following slides in comparison to a CGNS file as written by Pointwise

# Parent element data

CGNSView : bump\_3df\_006x002x001\_unstr\_native\_pw.cgns

Node Tree

- CGNSLibraryVersion
- Base
  - Information
  - bik-1
    - ZoneType
    - FamilyName
    - GridCoordinates
    - HexElements
    - bc-inflow
      - ElementRange
      - ElementConnectivity
    - ZoneBC
    - bc-symm-top
    - bc-symm-wall
    - bc-symm-z1
    - bc-symm-z2
    - dom-14
    - ZoneGridConnectivity
  - Unspecified
  - bc-inflow
  - bc-symm-top
  - bc-symm-wall
  - bc-symm-z1
  - bc-symm-z2
  - bik-2
  - bik-3
  - bc-outflow

Node Description

Parent Node: /Base/bik-1/bc-inflow  
Node Name: ElementConnectivity  
Node Label: DataArray\_t

Link Description

Link File:  Browse  
Link Node:  Browse

Data Description

Data Type: i4  
Dimensions: 8  
Bytes: 32

create modify read clear delete

Node Data

```
16 13 4 7
13 10 1 4
```

Line 1 (1) ValuesLine 4

CGNSView : bump\_3df\_006x002x001\_unstr.cgns

Node Tree

- CGNSLibraryVersion
- Base
  - Information
  - bik-1
    - ZoneType
    - FamilyName
    - GridCoordinates
    - HexElements
    - bc-inflow
      - ElementRange
      - ElementConnectivity
      - ParentElements
      - ParentElementsPosition
    - ZoneBC
    - bc-symm-top
    - bc-symm-wall
    - bc-symm-z1
    - bc-symm-z2
    - dom-14
    - ZoneGridConnectivity
  - Unspecified
  - bc-inflow
  - bc-symm-top
  - bc-symm-wall
  - bc-symm-z1
  - bc-symm-z2
  - bik-2
  - bik-3
  - bc-outflow

Node Description

Parent Node: /Base/bik-1/bc-inflow  
Node Name: ParentElements  
Node Label: DataArray\_t

Link Description

Link File:  Browse  
Link Node:  Browse

Data Description

Data Type: i4  
Dimensions: 2 2  
Bytes: 16

create modify read clear delete

Node Data

```
1 3
0 0
```

Line 1 (1) ValuesLine 2



Including two optional pieces of information in an unstructured CGNS file makes it easier for the user...

2. Providing “PointList” and “PointListDonor” information in the “GridConnectivity\_t” nodes of the “ZoneGridConnectivity\_t”
  - This information provides a more direct path to “merging connectivity” between unstructured element blocks
  - If a user assigns unique node ids starting with the first block and working towards the last and uses the “PointList” and “PointListDonor” information to merge nodes, it is straightforward to get a consistent connectivity for all blocks

# PointList and PointListDonor grid connectivity

The image displays two side-by-side screenshots of the CGNSview application, illustrating the configuration and data for a 'PointList' node within a 'PointListDonor' grid connectivity structure.

**Left Screenshot (CGNSview: bump\_3df\_006x002x001\_unstr\_native\_pw.cgns):**

- Node Tree:** Shows a hierarchy where 'ZoneGridConnectivity' contains '1to1Connection1', which in turn contains 'GridConnectivityType', 'GridLocation', and 'PointRange'.
- Node Description:** Parent Node: /Base/blk-1/ZoneGridConnectivity/1to1Connection1; Node Name: PointRange; Node Label: IndexRange\_t.
- Link Description:** Link File and Link Node fields are empty.
- Data Description:** Data Type: I4; Dimensions: 1 2; Bytes: 8.
- Node Data:** Contains values 19 and 20.

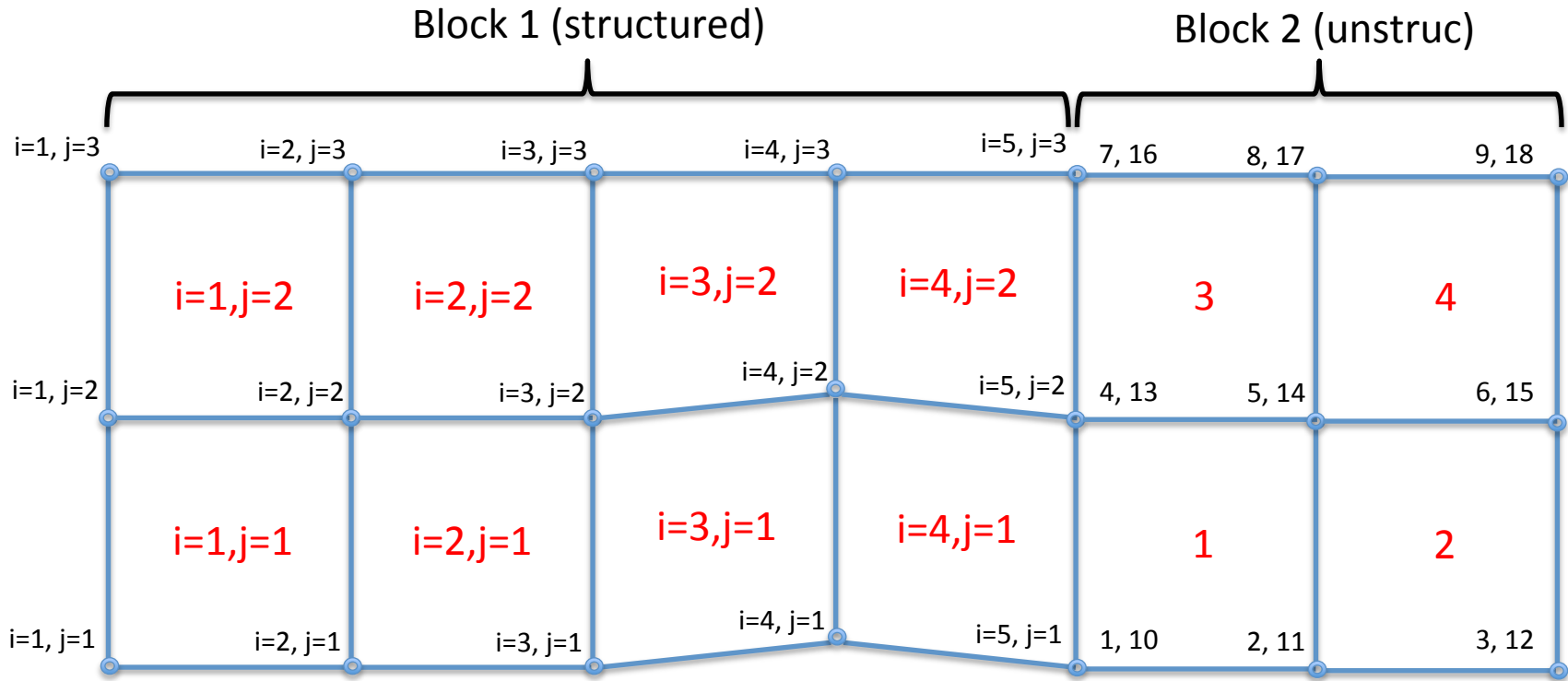
**Right Screenshot (CGNSview: bump\_3df\_006x002x001\_unstr.cgns):**

- Node Tree:** Shows a hierarchy where 'ZoneGridConnectivity' contains '1to1Connection1', which contains 'GridConnectivityType', 'PointList', and 'PointListDonor'. 'PointList' and 'PointListDonor' are circled in red.
- Node Description:** Parent Node: /Base/blk-1/ZoneGridConnectivity/1to1Connection1; Node Name: PointList; Node Label: IndexArray\_t.
- Link Description:** Link File and Link Node fields are empty.
- Data Description:** Data Type: I4; Dimensions: 1 6; Bytes: 24.
- Node Data:** Contains values 3, 6, 9, 12, 15, and 18.

An arrow points from the circled 'PointList' node in the right screenshot to a box containing the following values:

1
4
7
10
13
16

# Example hybrid grid



# Hybrid GridConnectivity\_t (structured->unstructured)

The screenshot shows the CGNSview interface. The 'Node Tree' on the left lists various grid connectivity types, with 'PointRange' under 'HybridConnectivityType' circled in red. The 'Node Description' panel on the right shows details for 'PointRange', including its parent node and label. The 'Data Description' panel shows a data type of 4, dimensions of 3 2, and 24 bytes. The 'Node Data' window at the bottom right displays the values '1' and '3' on separate lines. A text box with '1' and '3' is positioned below the data window, with an arrow pointing to the 'PointRange' node in the tree.

This is supported by CGNS for a structured block abutting with an unstructured block. But currently no grid generator will write this information (let alone a hybrid CGNS grid)

1  
3

# Hybrid GridConnectivity\_t (unstructured->structured)

**Node Tree**

- CGNSLibraryVersion
  - Base
    - blk1-structured
    - blk2-unstructured
      - ZoneType
      - FamilyName
      - GridCoordinates
        - CoordinateX
        - CoordinateY
        - CoordinateZ
      - HexElements
      - ElementRange
      - ElementConnectivity
    - bc-outflow
    - bc-symm-top
    - bc-symm-wall
    - bc-symm-z1
    - bc-symm-z2
    - hybrid-interface
    - ZoneGridConnectivity
      - HybridConnectivity1
        - GridConnectivityType
        - GridLocation
        - PointList
        - PointListPosition
        - PointList
        - PointRangeDonor
        - PointList
      - bc-inflow
      - bc-outflow
      - bc-symm-top
      - bc-symm-wall
      - bc-symm-z1
      - bc-symm-z2

This is NOT supported by CGNS for an unstructured block abutting with a structured block. UserDefinedData is being used here.

4  
4

This gives the face position of the unstructured element on the hybrid interface.

4 1 1  
4 2 1

This gives the i,j,k range of cells on the structured side of the hybrid interface.