Proposal for the addition of iterative or time-accurate data

to the CFD General Notation System

March 2000

Christopher Rumsey, NASA Langley Research Center
Robert Bush, United Technology Research Center
Diane Poirier, ICEM CFD Engineering
Stuart Ochs, Adapco
Mark Fisher, Boeing

In order to keep a record of time dependent or iterative data, two new data structures called `BaseIterativeData_t` and `ZoneIterativeData_t` are proposed. The `BaseIterativeData_t` data structure is recorded directly under the `CGNSBase_t` node. It contains information about the number of time steps or iterations being recorded, and the time and/or iteration values at each step. In addition, it may include the list of zones and families for each step of the simulation, if these vary throughout the simulation.

The `ZoneIterativeData_t` data structure is located under the `Zone_t` node. It allows to record pointers to zone data for each recorded step of the simulation.

SIDS definition:

1. The `BaseIterativeData_t` data structure under the `CGNSBase_t` data structure:

```
CGNSBase_t:=
  {
  BaseIterativeData_t   BaseIterativeData                                    (o)
    {
    int NumberOfSteps                                                        (r)
    DataArray_t<real,1,NumberOfSteps> TimeValues ;                           (o/r)
    DataArray_t<real,1,NumberOfSteps> IterationValues ;                      (r/o)

    DataArray_t<int,1,NumberOfSteps> NumberOfZones ;                         (o)
    DataArray_t<int,1,NumberOfSteps> NumberOfFamilies ;                      (o)
    DataArray_t<char,3,[32,MaxNumberOfZones,NumberOfSteps]> ZonePointers ;   (o)
    DataArray_t<char,3,[32,MaxNumberOfFamilies,NumberOfSteps]>FamilyPointers ;(o)

    List(DataArray_t<> DataArray1 ... DataArrayN)                            (o)
    List( Descriptor_t Descriptor1 ... DescriptorN ) ;                       (o)
    DataClass_t DataClass ;                                                  (o)
    DimensionalUnits_t DimensionalUnits ;                                    (o)
    }
  ...
  }
```

Notes:

- The `NumberOfSteps` is a required element of the `BaseIterativeData_t` data structure. It holds either the number of time steps or the number of iterations.
- `TimeValues` or `IterationValues` must be defined. If both are used, there must be a one-to-one correspondence between them.

- The fields NumberOfZones and ZonePointers are optional. They are used if different zone data structures apply to different steps of the simulation. Similarly, if the geometry varies with time or iteration, then the fields NumberOfFamilies and FamilyPointers are used to record which Family_t data structure(s) correspond(s) to which step.

- The DataArray_t for ZonePointers and FamilyPointers are defined as tri-dimensional arrays. For each recorded step, the name of all zones and families being used for this step may be recorded. Note that the names are limited to 32 characters, as usual in the SIDS. The variables MaxNumberOfZones and MaxNumberOfFamilies represent the maximum number of zones and families that apply to one step. So if NumberOfSteps=5 and NumberOfZones={2,2,3,4,3}, then MaxNumberOfZones equals 4.

- When NumberOfZones and NumberOfFamilies vary for different stored time step, the name "Null" is used in ZonePointers and FamilyPointers as appropriate for steps in which the NumberOfZones or NumberOfFamilies is less that the MaxNumerOfZones or MaxNumberOfFamilies.

- The DataClass_t, DimensionalUnits_t and Descriptor_t nodes may optionally be specified under the ZoneIterativeData_t node.

- Any numbers of extra DataArray_t nodes are allowed. These should be used to record data not covered by this specification.

2. The ZoneIterativeData_t data structure under the Zone_t data structure:

```
Zone_t<int CellDimension, int PhysicalDimension > :=
  {
  ZoneIterativeData_t ZoneIterativeData <NumberOfSteps> :=                    (o)
    {
    DataArray_t<char,2,[32,NumberOfSteps]> RigidGridMotionPointers ;          (o)
    DataArray_t<char,2,[32,NumberOfSteps]> ArbitraryGridMotionPointers ;      (o)
    DataArray_t<char,2,[32,NumberOfSteps]> GridCoordinatesPointers ;          (o)
    DataArray_t<char,2,[32,NumberOfSteps]> FlowSolutionsPointers ;            (o)
    DataArray_t<char,2,[32,NumberOfSteps]> ZoneBCPointers ;                   (o)
    DataArray_t<char,2,[32,NumberOfSteps]> ZoneGridConnectivityPointers ;     (o)

    List(DataArray_t<> DataArray1 ... DataArrayN)                             (o)
    List( Descriptor_t Descriptor1 ... DescriptorN ) ;                        (o)
    DataClass_t DataClass ;                                                   (o)
    DimensionalUnits_t DimensionalUnits ;                                     (o)
    }
  ...
  }
```

- The data arrays xxxPointers contain lists of associated data structures for each time step or iteration. They refer by name to data structures within the current zone. The name “Null” is used when a particular time or iteration does not have a corresponding data structure to point to.

- The DataClass_t, DimensionalUnits_t and Descriptor_t nodes may optionally be specified under the ZoneIterativeData_t node.

- Any numbers of extra DataArray_t nodes are allowed. These should be used to record data not covered by this specification.

- The ZoneIterativeData_t data structure may not exist without the BaseIterativeData_t under the CGNSBase_t node. However BaseIterativeData_t may exist without ZoneIterativeData_t.

Example 1: Rigid grid motion

In this example, the whole mesh moves rigidly so the only time-dependant data are the grid coordinates and flow solutions. However, since the mesh moves rigidly, the grid coordinates need not be recorded at each time step. Instead, a RigidGridMotion_t data structure is recorded for each step of the computation.

The number of steps and time values for each step are recorded under BaseIterativeData_t:

```
CGNSBase_t {
  BaseIterativeData_t {
    NumberOfSteps = N
    TimeValues = time1, time2, ..., timeN
  }
}
```

The multiple rigid grid motion and flow solution data structures are recorded under the zone. RigidGridMotionPointers and FlowSolutionPointers keep the list of which RigidGridMotion_t and FlowSolution_t nodes correspond to each time step:

```
Zone_t Zone {

 time independent data:
  GridCoordinates_t GridCoordinates
  ZoneBC_t ZoneBC
  ZoneGridConnectivity_t ZoneGridConnectivity

 time dependant data:
    RigidGridMotion_t RigidGridMotion#1
    RigidGridMotion_t RigidGridMotion#2
    ...
    RigidGridMotion_t RigidGridmotion#N

    FlowSolution_t Solution#0
    FlowSolution_t Solution#1
    FlowSolution_t Solution#2
    ...
    FlowSolution_t Solution#N

    ZoneIterativeData_t {
      RigidGridMotionPointers = {"RigidGridMotion#1, "RigidGridMotion#2", ...}
      FlowSolutionPointers = {"Solution#1", "Solution#2, ..., "Solution#N")
    }
}
```

Note that there may be more solutions under a zone than those pointed to by FlowSolutionPointers. In this example, Solution#0 could correspond to a restart solution.

Example 2: Deforming grid motion

In this example, velocity vectors are node dependant allowing for mesh deformation. In such a case, it is difficult or even impossible to recompute the mesh at each time step. Therefore the grid coordinates are recorded for each step.

Multiple GridCoordinates_t and Flowsolution_t data structures are recorded under the zone. In addition, the data structure ArbitraryGridmotion_t is recorded for each step. GridCoordinatesPointers, FlowSolutionPointers and ArbitraryGridMotionPointers_t keep the list of which grid coordinates definition, flow solution and arbitrary grid motion definition correspond to each time step:

```
Zone_t Zone {

  time independent data:
    ZoneBC_t ZoneBC
    ZoneGridConnectivity_t ZoneGridConnectivity

  time dependant data:
    List (GridCoordinates_t GridCoordinates MovedGrid#1 MovedGrid#2... MovedGrid#N)
    List (FlowSolution_t Solution#0 Solution#1 Solution#2... Solution#N)
    List (ArbitraryGridMotion_t ArbitraryGridMotion#1, ... ArbitraryGridMotion#N)
    ZoneIterativeData_t {
      GridCoordinatesPointers = {"MovedGrid#1", "MovedGrid#2", ..., "MovedGrid#N"}
      FlowSolutionPointers = {"Solution#1", "Solution#2, ..., "Solution#N")
      ArbitratyGridMotionPointers = {"ArbitraryGridMotion#1", ... ,
                                     " ArbitraryGridMotion#N" }
    }
}
```

Example 3: Adapted unstructured mesh

In this example, the mesh size varies at each remeshing, therefore new zones must be created. `ZonePointers` is used to keep a record of the zone definition corresponding to each recorded step. Lets assume that the solution is recorded every 50 iterations, and the grid is adapted every 100 iterations.

The number of steps, iteration values for each step, number of zones for each step, and name of these zones are recorded under `BaseIterativeData_t`:

```
CGNSBase_t {
  BaseIterativeData_t {
    NumberOfSteps = 4
    IterationValues = {50, 100, 150, 200}
    NumberOfZones = {1, 1, 1, 1}
    ZonePointers = {"Zone1", "Zone1", "Zone2", Zone2"}
  }
}
```

Each zone holds 2 solutions recorded at 50 iterations apart. Therefore the `ZoneIterativeData_t` data structure must be included to keep track of the `FlowSolutionPointers`:

```
Zone_t Zone1 {
  constant data:
    GridCoordinates_t GridCoordinates
    Elements_t Elements
    ZoneBC_t ZoneBC
  variable data:
    List (FlowSolution_t InitialSolution, Solution50, Solution100)
    ZoneIterativeData_t {
      FlowSolutionPointers = {"Solution50", "Solution100", "Null", "Null"}
    }
}
```

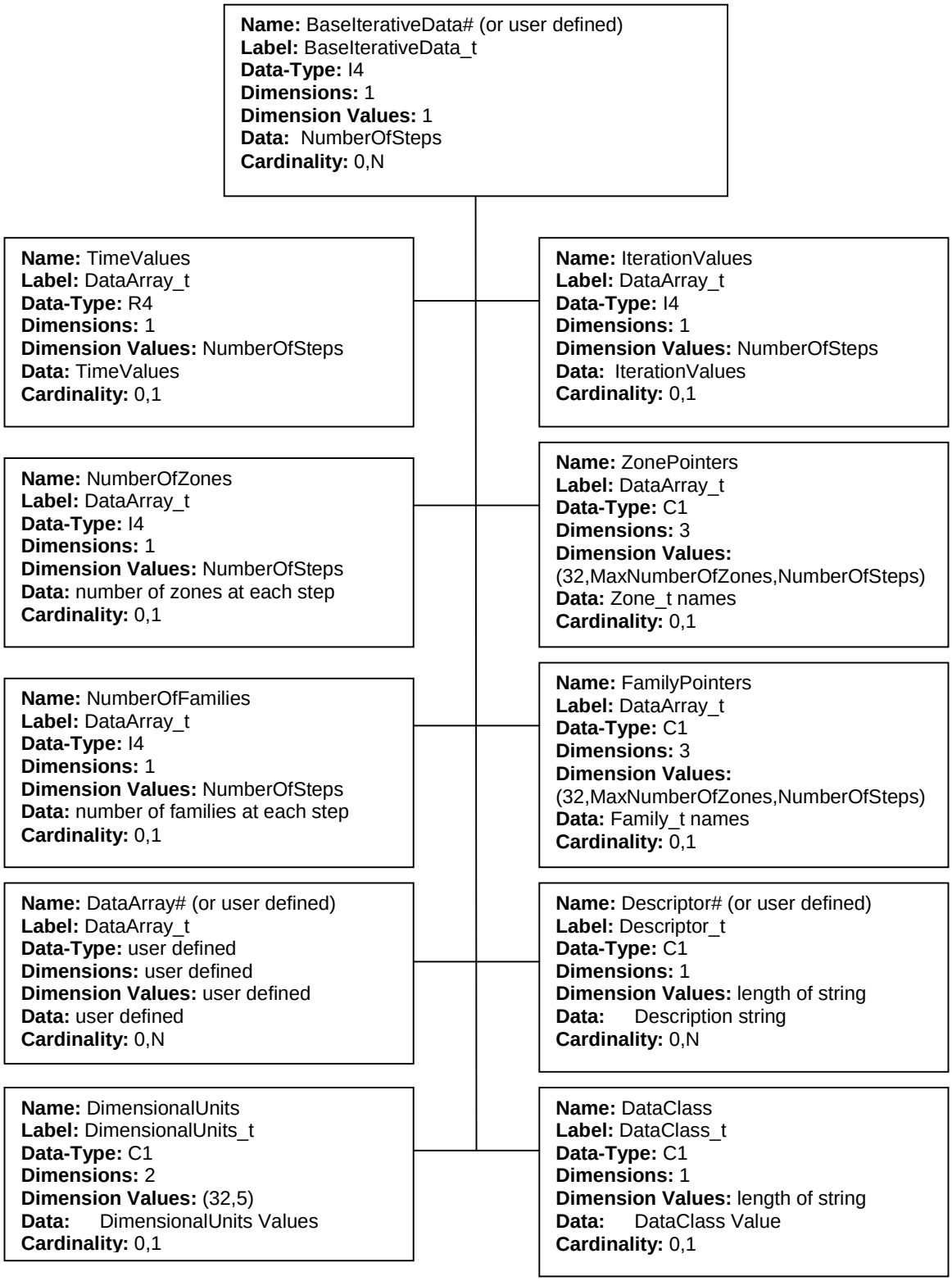
```
Zone_t Zone2 {
  constant data:
    GridCoordinates_t GridCoordinates
    Elements_t Elements
    ZoneBC_t ZoneBC
  variable data:
    List (FlowSolution_t RestartSolution, Solution150, Solution200)
    ZoneIterativeData_t {
      FlowSolutionPointers = {"Null", "Null", "Solution150", "Solution200"}
    }
}
```

Notes:

- If the solution was recorded every 100 iterations instead of every 50 iterations, then each zone would have only one `FlowSolution_t` node and the data structure `ZoneIterativeData_t` would not be required.

- Note that `FlowSolutionPointers` is always an array of size `NumberOfSteps` even if some of the steps are defined in another zone.

ADF file mapping definition of the `BaseIterativeData_t` data structure:



ADF file mapping definition of the `ZoneIterativeData_t` data structure: