CFD General Notation System (CGNS)
Usage for unstructured grids

Edwin van der Weide
Stanford University
Example Unstructured Grid
Unstructured grid storage

• Several possibilities to store an unstructured grid.
  – Every element type is stored in a separate Elements_t node. Recommended.
  – One Elements_t node, which stores all elements using the MIXED Element type.
  – Store all elements as arbitrary polygons, NGON_n Element type.
  – Arbitrary combinations of the possibilities above.

– **Pros**
  • Flexibility.

– **Cons**
  • Reading becomes complicated.
Connectivities (linear elements)

HEXA_8

PYRA_5

QUAD_4

PENTA_6

TETRA_4

TRI_3

See http://www.grc.nasa.gov/WWW/cgns/sids/conv.html#unstructgrid for all supported elements.
Info in the zone

- \# elements = \# elements of highest dimension.
  - E.g. for a 3D problem the number elements of the surface grid should NOT be stored in the zone.

Number of grid points

Number of volume elements
Single Zone vs. Multiple Zones

**Single Zone**
- No relative motion

**Multiple Zones**
- Relative motion or non-matching grids

Multiple zones can be used to store a domain decomposition

**Drawback:** not very flexible

**Better:** use the partial read/write functions
#include "cgnslib.h"

/* Open the CGNS for reading and check if the file was found. */
if(cg_open(gridFile, MODE_READ, &fileInd) != CG_OK)
    Terminate("readGridCGNS", cg_get_error());

/* Determine the of bases in the grid. This example assumes */
/* one base. However it is allowed to have multiple bases. */
if(cg_nbases(fileInd, &nBases)!= CG_OK)
    Terminate("readGridCGNS", cg_get_error());
if(nBases != 1)
    Terminate("readGridCGNS", "This example assumes one base");
base = 1;

/* Check the cell and physical dimensions of the bases. */
/* Both should be 3. */
if(cg_base_read(fileInd, base, cgnsName, &cellDim,
    &physDim) != CG_OK)
    Terminate("readGridCGNS", cg_get_error());
Example – CGNS Code (2)

/* Read the number of zones in the grid. */
/* This example assumes one zone.            */

if(cg_nzones(fileInd, base, &nZones) != CG_OK)
    Terminate("readGridCGNS", cg_get_error());
if(nZones != 1)
    Terminate("readGridCGNS", "This example assumes one zone");
zone = 1;

/* Check the zone type. This should be Unstructured. */

if(cg_zone_type(fileInd, base, zone, &zoneType) != CG_OK)
    Terminate("readGridCGNS", cg_get_error());
if(zoneType != Unstructured)
    Terminate("readGridCGNS", "Unstructured zone expected");

/* Determine the number of vertices and volume elements in this */
/* zone (and thus in the grid, because one zone is assumed).    */

if(cg_zone_read(fileInd, base, zone, zoneName, sizes) != CG_OK)
    Terminate("readGridCGNS", cg_get_error());
nVertices    = sizes[0];
nVolElements = sizes[1];
/* Determine the number and names of the coordinates. */

if(cg_ncoords(fileInd, base, zone, &nCoords) != CG_OK)
    Terminate("readGridCGNS", cg_get_error());

if(cg_coord_info(fileInd, base, zone, 1, &dataType, name) != CG_OK)
    Terminate("readCGNS", cg_get_error());

/* Read the x-coordinates. The y and z-coordinates can be read similarly. Just replace CoordinateX by CoordinateY and CoordinateZ respectively. This assumes Cartesian coordinates in double precision. Note that CGNS starts the numbering at 1 even if C is used. */

one = 1;
if(cg_coord_read(fileInd, base, zone, "CoordinateX", realDouble,
                 &one, &nVertices, coorX) != CG_OK)
    Terminate("readGridCGNS", cg_get_error());

/* Determine the number of sections for this zone. Note that surface elements can be stored in a volume zone, but they are NOT taken into account in the number obtained from cg_zone_read. */

if(cg_nsections(fileInd, base, zone, &nSections) != CG_OK)
    Terminate("readGridCGNS", cg_get_error());
Example – CGNS Code (4)

/* Loop over the number of sections and read the element */
/* connectivities. As CGNS starts the numbering at 1 the */
/* for-loop starts at 1 as well. */

for(sec=1; sec<=nSections; sec++) {
  /* Determine the element type and set the pointer for the */
  /* connectivity accordingly. */

  if(cg_section_read(fileInd, base, zone, sec, secName, &type,
                    &eBeg, &eEnd, &nBdry, &parentFlag) != CG_OK)
    Terminate("readGridCGNS", cg_get_error());

  switch (type) {
  case TETRA_4:
    conn = connTetra; break;
  case PYRA_5:
    conn = connPyra; break;
  case PENTA_6:
    conn = connPrisms; break;
  case HEXA_8:
    conn = connHexa; break;
  }
```c
Example – CGNS Code (5)

    case TRI_3:
        conn = connTri;    break;
    case QUAD_4:
        conn = connQuad; break;
    default:
        Terminate("readGridCGNS", "Unsupported element encountered.");
        break;
    }

    /* Read the connectivity. Again, the node numbering of the   */
    /* connectivities start at 1. If internally a starting index */
    /* of 0 is used (typical for C-codes) 1 must be subtracted */
    /* from the connectivities read.                           */

    if(cg_elements_read(fileInd, base, zone, sec, conn, NULL) != CG_OK)
        Terminate("readGridCGNS", cg_get_error());
    }
```
Example – CGNS Code (6)

/* Determine the number of boundary conditions for this zone. */

if(cg_nbocos(fileInd, base, zone, &nBocos) != CG_OK)
    Terminate("readGridCGNS", cg_get_error());

/* Loop over the number of boundary conditions. */

for(boco=1; boco<=nBocos; boco++)
{
    /* Read the info for this boundary condition. */

    if(cg_boco_info(fileInd, base, zone, boco, bocoName, &bocoType,
        &ptsetType, &nBCElem, &normalIndex,
        &normListFlag, &normDataType, &nDataSet) != CG_OK)
        Terminate("readGridCGNS", cg_get_error());

    /* Read the element ID’s. */

    if(cg_boco_read(fileInd, base, zone, boco, BCElemRead,
        NULL) != CG_OK)
        Terminate("readGridCGNS", cg_get_error());

    /* And much more to make it fit into the */
    /* internal datastructures. */
}
Conclusions

• CGNS can store a wide variety of unstructured mesh types.

• Midlevel API offers many functions to read/write CGNS files, see http://www.grc.nasa.gov/WWW/cgns/midlevel/index.html

• Simple example to read a grid has been given.

• In a real code more API-functions will be used for checking the available data, etc.