Google Test, HDF5 revisited and Polyhedral Cells in FSDM

Johannes Holke, Margrit Klitz, Alexander Rüttgers
Simulation and Software Technology, DLR

FLOWSIM Experts Meeting
Airbus Helicopters

Using the Google Test framework

- Goal: **Use the Google Test library for all unit tests in FSDM**
  - Rewrite existing tests as Google Tests
  - Write additional unit tests

- Challenge: **Google Test is not 'MPI Aware'**
  - A test may fail on rank 1, but not on rank 0 and will be reported as PASSED in the output (**false positive**).

  - Assertions in parallel programs can cause **deadlocks**

Example output

```
[----------] 1 test from testFSMeshIOHDF5SimpleNoGlobalIDs
[ RUN      ] testFSMeshIOHDF5SimpleNoGlobalIDs.IOConsistency
[       OK ] testFSMeshIOHDF5SimpleNoGlobalIDs.IOConsistency (48 ms)
[----------] 2 tests from SimpleNoGlobalIDs
[ RUN      ] SimpleNoGlobalIDs.otherProcNumbers
/localdata2/source/fsdm/fsdm/tests/src/testFSMeshIOHDF5_SimpleNoGlobalIDs.cpp:666: Failure
Value of: 1
Expected: 0
[  FAILED  ] SimpleNoGlobalIDs.otherProcNumbers (43 ms)
[ RUN      ] SimpleNoGlobalIDs.exportVTK
[       OK ] SimpleNoGlobalIDs.exportVTK (12 ms)
[==========] 59 tests from 27 test cases ran. (42048 ms total)
[  PASSED  ] 58 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] SimpleNoGlobalIDs.otherProcNumbers

 1 FAILED TEST
```

```
// Check dimension of intArray
ASSERT_EQ (intArray.NDims(), 1);
// broadcast intArray from proc 0
intArray.Broadcast(mClac, 0);
```

If some processes fail here

The rest is stuck here

Using the Google Test framework

- Solution: **We extended the functionality of Google Test**
  - Test results are synchronized between the MPI processes.
    A failure on one processes implies a failure on all
    ⟶ **No more false positive test**

  - Implemented new Macros **ASSERT_EQ_MPI, EXPECT_EQ_MPI**, etc.
    - MPI synchronized
    - If one process fails, every process fails

```
// Check dimension of intArray
ASSERT_EQ_MPI (intArray.NDims(), 1);
// broadcast intArray from proc 0
intArray.Broadcast(mClac, 0);
```

If some processes fail here, all will fail

Either all processes call this, or none

⟶ **No more deadlocks**

Using the Google Test framework

- Googletest with MPI is avaible for FSDM since this summer.

  ⟶ **Use it!**

- But take care:

```
if (procID == 0) {
  ASSERT_EQ_MPI (intArray.NDims(), 1);
}
```

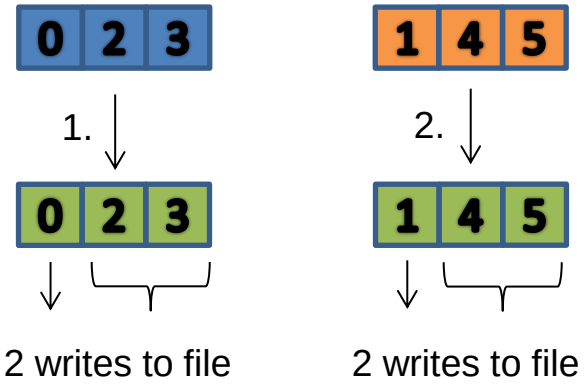
⟵ Process 0 waits forever

- In this case, use **Expect_EQ** instead

- See also: thirdparty/gtest-gmock-1.8.0-mpi/**MPIGuide.md**

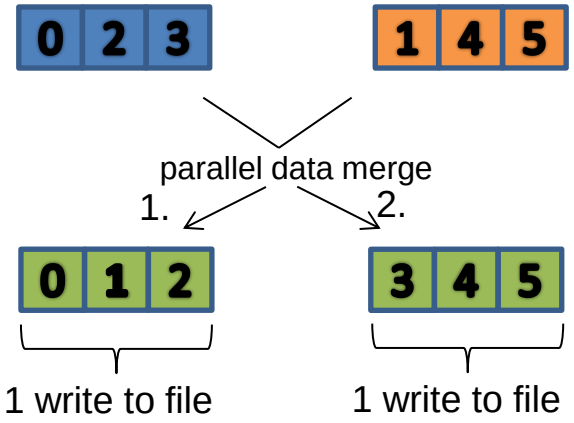
Enhancing HDF5 export with parallel data merge

- **One process** collects the mesh IDs and data and **writes connected** chunks to the HDF5 file
  - Before:



worst case: as many File I/O operations as data items

  - Now:

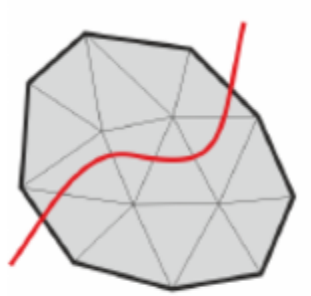


Always writes chunks of size C.
Reduces File I/O operations drastically

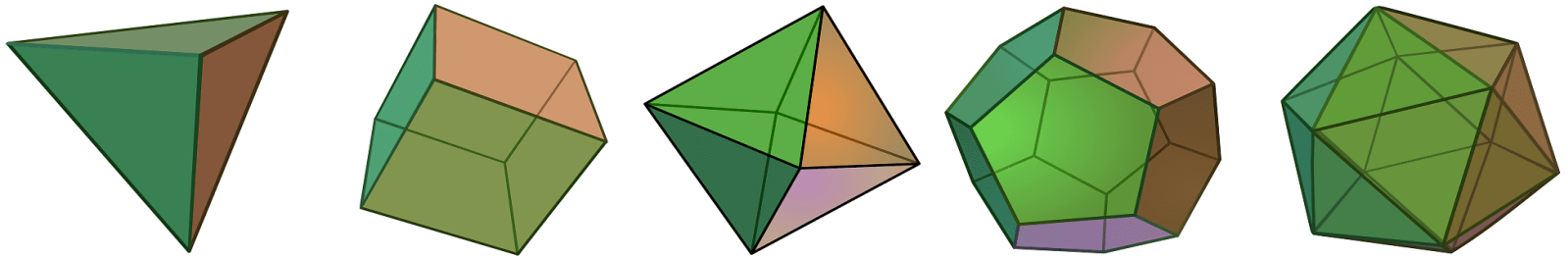
Polyhedral Cells in FSDM

- **Motivation:**
  - A geometry cuts our mesh (consisting of simplex elements)
  - Simplex elements degenerate to polyhedral elements

- **Problem:**
  - FSDM expects elements to have a fixed maximum number of
    - Corners
    - Faces
    - Edges
  - Disrupts current strategy of saving and addressing elements

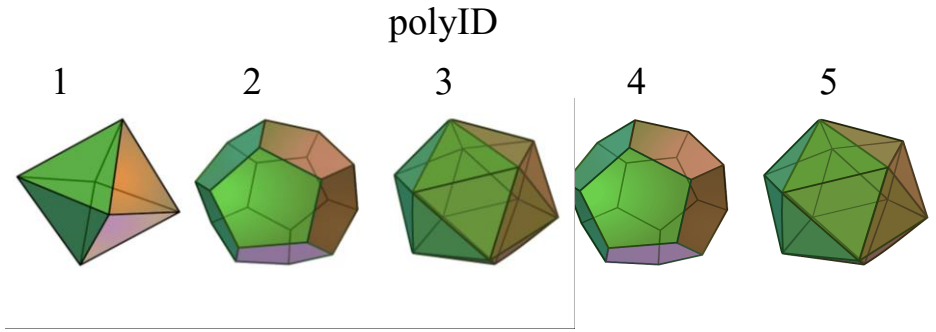


Cutting configuration



Figures from Wikipedia and from https://www.researchgate.net/figure/Illustration-of-di-ff-erent-methods-for-incorporating-cuts-into-a-tetrahedral-mesh-a_fig3_273890104

Polyhedral Cells Implementation Strategy



- New cell type CT_Poly
- Arbitrary number of faces, edges and corners
- All polyhedral cells get an identification number polyID

```
void
FSPolyCellInfo::PolySetFaceNodes (intT PolyId, intT Face, FSIntArrayT & Nodes)
{
   mPolyFaceNodes(PolyId, Face) = Nodes;
}
```

- New class FSPolyCellInfo derived from FSCellInfo

FSCellInfo	FSPolyCellInfo
intT GetNFaces(FSMeshEnums::CellType t)	intT GetNFaces(FSMeshEnums::CellType t, FS_intT PolyId)
static const intT cNFaces[FSMESH_NCELLTYPES+1]	FSIntArrayT mPolyNFaces
intT cFaceNodes[FSMESH_NCELLTYPES+1][FSMESH_NMAXCELLFACES] [FSMESH_NMAXFACENODES];	FSRegister<FSIntArrayT> mPolyFaceNodes

FSPolyCellInfo

- Challenge: `PolyCellInfo` has to be instantiated and initialized

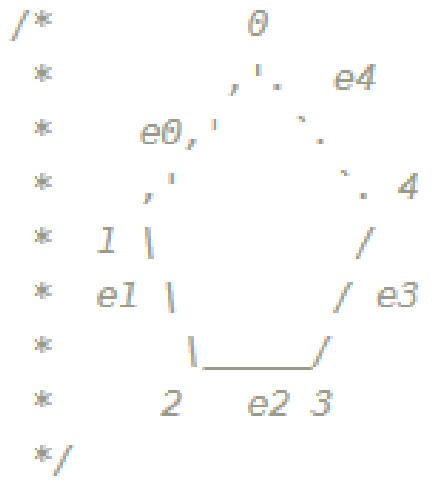
Before:

- `FSCellInfo::GetNCorners(FSMeshEnums::CT_Tri3);`
- No instantiation needed. Independent of the mesh.

Now:

```
FSPolyCellInfo mCellInfo(numPoly);
    mCellInfo.PolySetNCorner(3, 5)
      .
      .
      .
    mCellInfo.GetNCorners(FSMeshEnums::CT_Poly, 3);
```

- Instantiation needed. Depends on the mesh.



polyID = 3, # corners = 5

Polyhedral Cells
Status of Implementation

- Implemented construction routines for the polyhedral cells «FSPolyCellInfo»
- Implemented three construction tests (two in 2d and one in 3d)
- Started with a new cell pool «FSUnstructPolyCellPool»

```
// Build a simple pentagon as Polyhedron
TEST(testFSPolyCellInfo, pentagon) {
    FSPolyCellInfo PCellInfo(1);

    /*         0
     *        ,'.  e4
     *    e0,'   `.
     *    ,'       `. 4
     *  1 \         /
     *  e1 \       / e3
     *      \_____/
     *     2   e2 3
     */

    PCellInfo.PolySetNCorners(0, 5); // Set number of Corners
    PCellInfo.PolySetNNodes(0, 5);   // Set number of Nodes (same as corners)
    PCellInfo.PolySetNEdges(0, 5);    // Set number of edges
```

Conclusion

- Enhanced Google Test with MPI.
- Integrated Google Test into FSDM.
- Implemented parallel data merge for HDF5 output
    - Next step: Benchmarks
- Started working on polyhedral cells

Snippets

```
void
FSPolyCellInfo::PolyPrepareRegisters ()
{
    mPolyEdgeNodes.Prepare();
    mPolyFaceCorners.Prepare();
    mPolyFaceNodes.Prepare();
    mPolyFaceTypes.Prepare();
    // We now add empty data to each register, in order
    // to be able to fill them later possibly out of order.
    for (intT i = 0;i < mNumPolys;i++) {
        for (intT j = 0;j < mPolyNEdges[i];j++) {
            mPolyEdgeNodes.Add(i, FSIntArrayT(0));
        }
        for (intT j = 0;j < mPolyNFaces[i];j++) {
            mPolyFaceTypes.Add(i, FSMeshEnums::CT_Undefined);
            mPolyFaceCorners.Add(i, FSIntArrayT(0));
            mPolyFaceNodes.Add(i, FSIntArrayT(0));
        }
    }
}
```