Nektar++ 5.0

Target: Jan 2018

v4.5.0

NekMesh:

- Add periodic boundary condition meshing in 2D (!733)
- Adjust boundary layer thickness in corners in 2D (!739)

Library

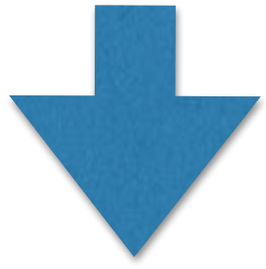
- Added in sum factorisation version for pyramid expansions and orthogonal expansion in pyramids (!750)

FieldConvert:

- Add input module for Semtex field files (!777)

Documentation:

- Added the developer-guide repository as a submodule (!751)



v5.0

C++ 11

Dear all,

As part of our ongoing efforts to modernise our codebase, we are planning to transition to the C++11 standard in the near future.

A consequence of this is that you will in future require a compiler which supports the C++11 standard (either as an option, such as --std=c++11, or by default).

Current versions of popular compilers all fully support the standard, but older compilers may only have partial support. We recommend the following compiler versions to minimise the likelihood of problems going forward:

- GCC: 4.8 or later
- Clang: 3.3 or later
- MSVC: 19.0 or later
- Intel: 15.0 or later
- PGI: 2015 or later

We encourage users to test out support for basic C++11 by compiling the branch 'feature/libutils-c++11' from the code repository and updating build scripts and configuration as necessary.

Please let us know if you have any problems or concerns. We will be making the transition after the Nektar++ Workshop in mid-June.

Cheers,
Nektar++ Development Team

C++ 11

- Variadic Templates
- Auto
- Lambda functions
- Native shared pointers
- Unordered maps

Managing external dependencies

- C++ 11 allows us to remove loki
- Reduce duplication of Boost/C++
- Remove specialised "mod"-metis and replace with scotch
  - Will allow CMake to use default packages
- Investigating CCMIO library for star-ccm interfacing

NekMesh 3D

- Independent web site & release for people who do not need full Nektar++ release

- Reduction of duplicate code between NekMesh and existing libraries

- Integrating CAD engine into Spatial Domains library

    - Allows for "on the fly" mesh movement

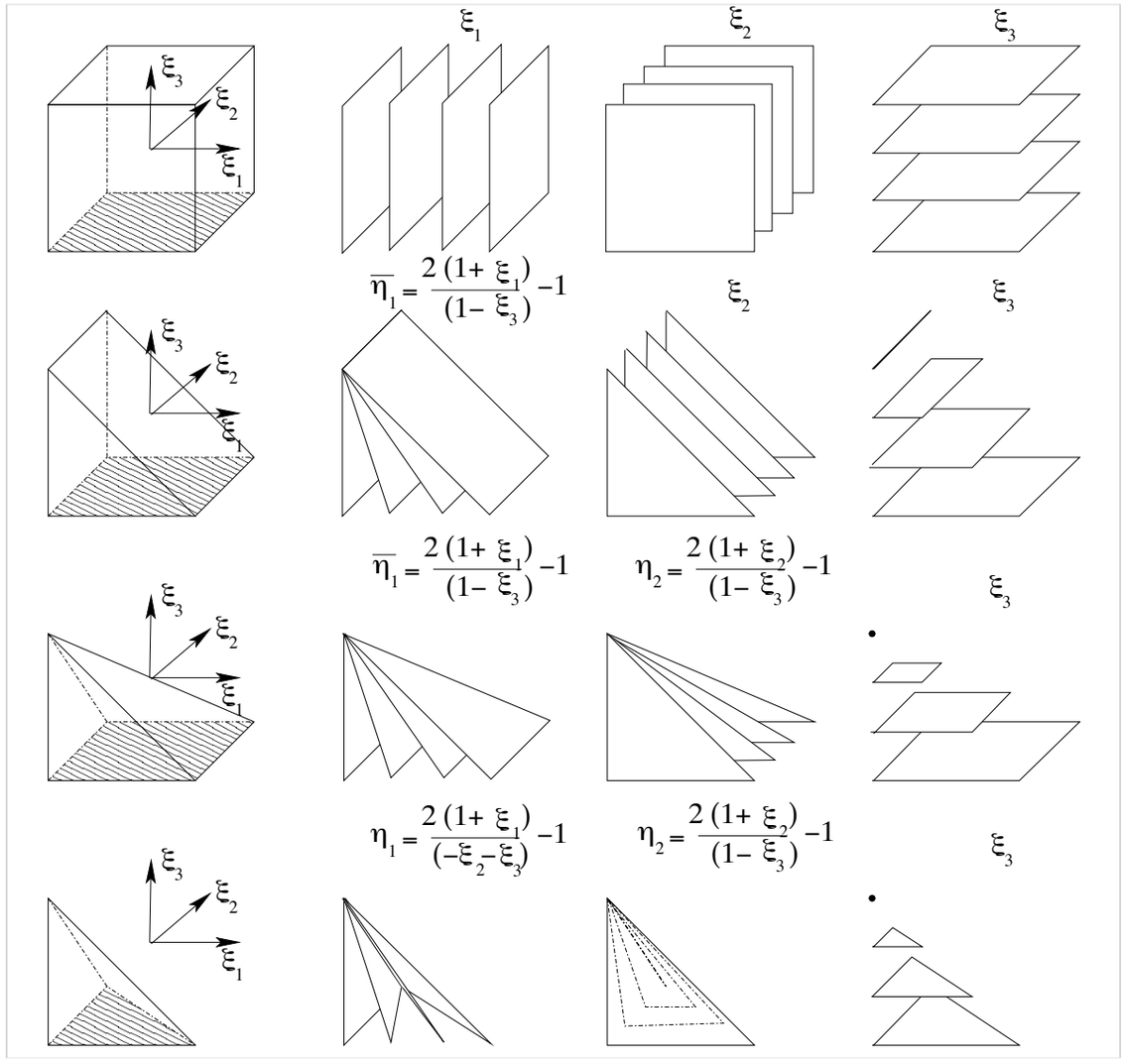
HDF5 Geometry

- Currently severe limitations on big meshes: > 10K partitions, 10M elements
- Key bottleneck is xml format
  - Slow/conflicted reading
  - Partition then requires a write
- Switch to binary based hdf5 format
  - Parallel partitioning parmetis/ptscotch
  - Will maintain xml backwards compatibility

Accelerator & Memory Layout

- Workshop on Thursday afternoon:
- AoSoA Blend of AoS and SoA
- Kokkos vs OpenMP 4.5
- Targeting Explicit code performance

Pyramids & 3D Variable P



Hexahedral expansion: $\phi_{pqr}(\xi_1,\xi_2,\xi_3) = \widetilde{\psi}^a_p(\xi_1)\ \widetilde{\psi}^a_q(\xi_2)\ \widetilde{\psi}^a_r(\xi_3),$

Prismatic expansion: $\phi_{pqr}(\xi_1,\xi_2,\xi_3) = \widetilde{\psi}^a_p(\overline{\eta_1})\ \widetilde{\psi}^a_q(\xi_2)\ \widetilde{\psi}^b_{pr}(\xi_3),$

Pyramidic expansion: $\phi_{pqr}(\xi_1,\xi_2,\xi_3) = \widetilde{\psi}^a_p(\overline{\eta_1})\ \widetilde{\psi}^a_q(\eta_2)\ \widetilde{\psi}^c_{pqr}(\eta_3),$

Tetrahedral expansion: $\phi_{pqr}(\xi_1,\xi_2,\xi_3) = \widetilde{\psi}^a_p(\eta_1)\ \widetilde{\psi}^b_{pq}(\eta_2)\ \widetilde{\psi}^c_{pqr}(\eta_3),$

- Pyramids now in similar pattern to other shapes
- Now have 3D preconditioners for pyramids and variables P

Developers Guide

- Need to formalise …. Mike and Dave (Developer)!

FieldConvert Refactor

- As discussed by Douglas.

Co-Simulation coupling

- As discussed by Kilian.

Any comments?