Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

ModFEM - use cases and the internal flow of calculations

ModFEM – use cases

- download and install ModFEM framework
  - a suitable manual (user's guide) chapter should be created (KM?)
  - take into account possible use of ModFEM as part of grid infrastructure (KM?)
- compile (build) a set of particular ModFEM application
  - a suitable manual (user's guide) chapter should be created (JB?)

ModFEM – use cases

- prepare mesh files
  - use OpenSource mesh generators
    - the list of generators that can cooperate with ModFEM should be available in ModFEM documentation (Webpage, PDF version) (KM?)
      - the list should indicate whether:
        - » the code of generators is included with ModFEM or
        - » ModFEM documentation contains links to pages with proper versions of code
    - ModFEM documentation should at least sketch the way in which mesh input files should be created by the selected mesh generators (KM?)

ModFEM – use cases

- prepare mesh files
  - import files from OpenSource or commercial mesh generators
    - the list of accepted file formats should be included in ModFEM documentation (KM?)
    - available options for importing files to ModFEM should be specified and described in documentation (in a user's guide fashion) (KM?)
      - options may include creation of blocks with internal contact boundaries, creation of “boundary layers”, specification of per element data (materials) or per face data (boundary conditions)
      - mesh quality control, improvement

ModFEM – use cases

- prepare configuration input files
  - the process of creating input files should be described in the user's guide for each file:
    - problem specification
    - materials specifications (PC?, KM?)
    - boundary conditions specification (PC?, KM?)
    - optional specification of solver input file (KB?)
  - the content of input files (especially problem definition) is related to the options for the flow of calculations that will be described later
    - the different sections in problem specification files should be described in different sections of user's guide (and by different persons)

ModFEM – use cases

- start the code for a given problem
  - shell scripts or other means seem to be necessary for proper starting of ModFEM for at least the following situations
    - MPI+...
    - ...+OpenCL
    - MPI+...+OpenCL
  - it is advisable to prepare a guidelines for preparing and several example realizations for such shell scripts (KM?, JB?)

ModFEM – example problems

- convection-diffusion
  - diff_in_cube (also diff_in_box)
    - single linear problem
    - example with the known exact solution
    - good for testing new approximation types (quadratic, DG etc.)
    - good for testing adaptivity (error reduction)
    - diff_in_box good for testing parallel adaptivity (large initial mesh), good for testing scalability

ModFEM – example problems

- convection-diffusion
  - diff_1d_in_cube
    - example with the known exact solution
    - for testing stabilization and convergence as a function of local Peclet number
    - for testing “boundary layers” of prismatic elements and anisotropic adaptivity
    -

ModFEM – example problems

- convection-diffusion
  - wave_in_box (or wave_in_cube)
    - for testing dynamic adaptivity
    - for testing parallel dynamic adaptivity and load balancing
    - simple case – wave_in_box along x-dimension
    - another case – wave_in_cube arbitrary direction
    -

ModFEM – example problems

- incompressible Navier-Stokes
  - LDC – steady-state problem
    - good for testing pseudo-time integration methods
    - good for testing accuracy for different weak formulations (stability terms)
    - existing reference solutions in the literature
    - good for testing adaptivity for NS equations (but the problem is not singular...)
  - BFS – similar to LDC

ModFEM – example problems

- incompressible Navier-Stokes
  - VKV – transient problem
    - good for testing accuracy of time integration algorithms
    - good for testing accuracy for different weak formulations (stability terms)
    - existing reference solutions in the literature
    - good for testing adaptivity for NS equations
    - good for testing parallel adaptivity with dynamic load balancing

ModFEM – example problems

- heat transfer
  - free cooling examples
    - comparison with experimental data
    - good for testing boundary conditions

ModFEM – example problems

- incompressible Navier-Stokes + heat transfer
  - HDC
    - comparison with reference solutions
    - good for testing time integration/nonlinear solution/linear solution strategies

ModFEM – example problems

- incompressible Navier-Stokes + heat transfer
  - quenching examples
    - comparison with experimental and commercial software results
    - good for testing solution strategies
    - good for testing adaptation strategies

ModFEM – example problems

- plasticity with flow formulation...
  - cone
  - cube
  - no friction
  - friction

ModFEM – flow of calculations

- each process reads necessary input files and initializes proper data structures
  - parallel reading of input files should be implemented and described in user's guide (KM?)
  - for parallel (distributed memory) execution domain decomposition must be ensured
    - either by reading already prepared files for each process (see above)
    - or by performing DD by suitable tools (Metis, ParMetis interface – described in theory manual and technical specification)(KM?)

ModFEM – flow of calculations

- each process starts execution of (depending upon the problem solved):
  - time integration
  - non-linear system solution
  - solver of linear equations
- the choice of particular solution strategy depends on the problem type (ModFEM executable) and particular problem (specified by e.g. problem name)

ModFEM – flow of calculations

- starting execution:
  - standard from terminal with working directory as parameter
  - mpirun, mpiexec with suitable parameters
  - scripts:
    - for MPI
      - e.g. for debugging
      - e.g. for clusters
    - for OpenCL
      - copying of suitable kernels
    - for MPI/OpenCL
      - ?

ModFEM – flow of calculations

- reading input files:
  - problem
  - mesh
  - field
  - BC
  - materials
- MPI
  - one node reads all
    - Metis domain decomposition
  - all nodes read all
    - initial load balance – ParMetis?

ModFEM – flow of calculations

- how the hell parallel (MPI) solution works?
  - domain decomposition with overlap
  - each process holds data for:
    - owned mesh and DOF entities
    - overlap mesh and DOF entities
  - three distinct problems:
    - parallel mesh adaptation (and load balance)
    - parallel solution
      - linear solution only – time integration and non-linear solution are just replicated on each node
    - parallel input/output

ModFEM – flow of calculations

- how the hell parallel (MPI) solution works?
  - standard mesh and approximation modules are agnostic to parallel (MPI) execution
  - all necessary mesh and approximation procedures are in separate overlay modules for handling domain decomposition
    - mmpd_prism and mmpd_adapter
    - appd_std_lin and appd_dg_prism
  - as usual, only problem dependent procedures and general utilities call overlay procedures
  - solver interface call problem dependent procedures for necessary operations

ModFEM – flow of calculations

- starting simulation:
  - MPI: synchronisation (broadcast)
- time integration or single problem solution
- each solution strategy (related to particular problems) leads finally to a sequence of solutions of linear problems
- between solutions mesh adaptations (and dynamic load balancing for parallel examples) can be performed
- solution strategies are contained in time integration sub-directories of problem modules

ModFEM – flow of calculations

- mesh adaptations are performed by mesh management modules
  - technical details can be described in manuals and user's guides related to particular mesh management modules
  - interesting variations include:
    - isotropic refinements
    - anisotropic refinements
    - strategies for near boundary regions (“boundary layers”)
  - parallel (MPI) mesh adaptations are accompanied by load balancing
    - details left to particular parallel mesh management modules (currently mmd_prism and mmd_adapter)

ModFEM – flow of calculations

- mesh adaptivity can be based on:
  - error estimation
  - any other user defined indicators (possibly problem dependent)
- error estimation
  - the only method in ModFEM is ZZ (Zienkiewicz-Zhu) error estimate based on derivative recovery
- for selected examples exact solutions exist and errors can be computed not estimated

ModFEM – flow of calculations

- time integration procedures
  - single stage procedures only
    - non-linear implicit first order Euler scheme
      - inner non-linear iterations
        - » different strategies for transient and steady-state problems
        - » breaking criteria based on the number of iterations or the norm of update of unknowns
        - » how about ns_supg?
    - linear implicit second order Crank-Nicolson (trapezoidal rule)
      - does it work at all??? (PC!!!)
  - adaptive time stepping strategies
    - CFL based
    - non-linear convergence based

ModFEM – flow of calculations

- time integration procedures
  - adaptive time stepping strategies
    - CFL based
    - non-linear convergence based
  - after each time step
    - time step length adaptation
    - solution rewriting
    - dumping
      - graphics data
      - checkpoint-restart data
    - breaking condition calculations
  - after mesh adaptation (and initially)
    - solver data structure recreation
    - DOFs exchange tables recreation

ModFEM – flow of calculations

- time integration procedures
  - non-linear solver strategies
    - Picard iterations only...
    - two kinds of stopping criteria:
      - based on absolute value of update norm
      - based on relative to initial value of update norm
  - opportunities for OpenMP parallelization
    - reading, writing and rewriting DOFs
    - calculating (error) norms
  - MPI (PARALLEL) modifications
    - exchange tables creation
    - control data broadcasts

ModFEM – flow of calculations

- how the hell parallel (MPI) solution works?
  - the key procedure: utr_get_list_ent
    - all (active) integration entities in process data structures can be included
      - if their SMs contain rows corresponding to not owned DOF entities their will simply not be assembled
    - there must be strict division of DOF entities
      - owned entities will have rows of global SM assembled (and used in solution)
      - ghost (overlap) entities will have no rows, but will have DOF structures and blocks, and will be used in matrix-vector products
      - owned entities must be first on the list
        - » they should not be considered in renumbering...

ModFEM – flow of calculations

- how the hell parallel (MPI) solution works?
  - iterative solver needs only:
    - global vector norm
    - global scalar product
    - exchange of DOFs for matrix-vector product
  - all procedures are called back to problem module
    - finally that is parallel approximation module that handles all these procedures

ModFEM – flow of calculations

- how the hell parallel (MPI) solution works?
  - exchange tables
    - creating
      - all-to-all communication
      - establishing local-to-global mapping, i.e.:
        - » for a given sending/receiving processor and a DOF entity with local position on sending/receiving list
        - » set the position in subdomain vector of DOFs (including ghost (overlap) DOFs (entities))
    - using
      - for each sending/receiving processor (communication only with neighbours)
      - just rewrite from one vector (sending/receiving list) to another vector (subdomain vector of DOFs)

ModFEM - performance

- Phases important for performance:
  - read data from CPU data structures
    - geometry
    - problem dependent coefficients
    - (shape functions and integration points data for reference elements must be present in fast memory during calculations)
  - send possibly to accelerator memory