EuroHPC
Joint Undertaking

Data driven Computational Mechanics at EXascale

DCoMEX

Data driven Computational Mechanics at EXascale

Work program topic: EuroHPC-01-2019
Type of action: Research and Innovation Action (RIA)

DCOMEX-BIO PROTOTYPE (Software Module)

DELIVERABLE D7.1

Version No 1

http://www.dcomex.eu/
This project has received funding from the European High-Performance Computing Joint Undertaking Joint Undertaking ('the JU'), under Grant Agreement No 956201

DOCUMENT SUMMARY INFORMATION

Project Title	Data driven Computational Mechanics at EXascale
Project Acronym	DCoMEX
Project No:	956201
Call Identifier:	EuroHPC-01-2019
Project Start Date	01/04/2021
Related work package	WP 7
Related task(s)	Task 7.1
Lead Organisation	UCY
Submission date	18/02/2024
Re-submission date	
Dissemination Level	PU

Quality Control:

	Who	Affiliation	Date
Checked by internal	George Stavroulakis	NTUA	31/09/2021
Checked by WP Leader	T. Stylianopoulos	NTUA	31/09/2021
Checked by Project Coordinator	Vissarion Papadopoulos	NTUA	31/09/2021

Document Change History:

Version	Date	Author (s)	Affiliation	Comment

Contents

1. Description

Deliverable 7.2 is associated to WP7 “XXX YYY” of the DCoMEX project, and it provides the algorithmic description of DCOMEX-BIO software module, as this is implemented and integrated in MSolve (see: https://github.com/mgroupntua/DrugDeliveryModel.Tests) . The implementation of this coupled Multiphysics problem in MSolve consists of 4 main building blocks:

i) The main solution script that implements a workflow that:
- manages the input data of the analysis (parameters, mesh solution time-stepping)
- builds the main analysis objects required for the solution procedure (initialized the shared quantities data structures)
- prescribes and executes a stepwise staggered solution process (manages the state variables of the models)
- monitors the solution process (iterative statistics of staggered and nested solvers, convergence of the problems)
- keeps logs of the solution and exports the desired data

ii) Subproblem model providers (MGroup.DrugDeliveryModel.Tests.Integration PorousModelNonLinearProvider.cs, PorousModelSmallStrainProvider.cs, TCellModelProvider.cs) that:

- build or update the separate MSolve models (MGroup.MSolve.Discretization.Entities Model.cs) with appropriate boundary and initial conditions for each PDE (or a pointwise distributed ODE) for the specific parameters defined by the solution workflow and for external loading resulting from the solution state of the rest of the PDEs of the coupled problem during the staggered solution process
- build object instances of MSolve solution modules such as solvers MGroup.Solvers.Direct SklineSolver.cs or analyzers MGroup.NumericalAnalyzers.Dynamic GeneralizedAlphaDynamicAnalyzer.cs)
- update data structures that are dependent on the primal variable that the PDE describes, differ for each staggered solution step and are necessary for the calculation of external loading for the rest of the PDE models.

iii) Tailored coupled solution coordinators (MGroup.DrugDeliveryModel.Tests.Integration Coupled5eqNLProvider.cs)
- Coordinate the interaction of the subproblem providers and the update of the shared quantities
- Gather arrays of MSolve algorithmic solution components and implements a model array creation function that describes the sequence of update steps for the shared quantities and the model variables.
- And passes the finalized arrays to the Stepwise Staggered analyzer, and executes the model creation function

iv) StaggeredAnalyzer implements the partitioned solution scheme for a coupled muti-physics problem by managing a set of solution components (analyzers and solvers) each one describing the equilibrium of the separate subproblems. At each iteration an update of these components is performed in a way determined by the coupled solution coordinator.
Next the contructor of the StaggeredAnalyzer class is given:

2. Algorithmic implementation in the MSolve software

The constructor of the porous model provider is given:

public PorousModelNonLinearProvider(ComsolMeshReader modelReader,
double Param1, Param2 …… ParamN, List<(BoundaryConditionCase, StructuralDof[], double[][], double[])> BCList1,BCList2,…… BCList4, double timeStep, double totalTime, Dictionary<int, double> lambda, Dictionary<int, double[][]> pressureTensorDivergenceAtElementGaussPoints, int unodeIdToMonitor, StructuralDof udofTypeToMonitor)

ComsolMeshReader modelReader: *a preprocessor component that provides mesh data*

double Param1, Param2 …… ParamN: *parameter values such as modulus of elasticity etc.*

List<(BoundaryConditionCase, StructuralDof[], double[][], double[])> BCList1,BCList2,…… BCList4: *a compact representation of BCs and ICs to be applied to the model*

double timeStep, double totalTime: *time stepping parameters*

Dictionary<int, double> lambda: *example of a shared quantity that is set as prescribed for the porous model provider*

Dictionary<int, double[][]> pressureTensorDivergenceAtElementGaussPoints: *example of a shared quantity that porous model provider updates upon solution but is used by other model builders*

Four basic methods are also given:

```
public Model GetModel(): creates a model for the porous problem at each staggered solution step

public void AddBoundaryConditions(Model model): updates model's boundary conditions and loading

public (IParentAnalyzer analyzer, ISolver solver, IChildAnalyzer loadcontrolAnalyzer) GetAppropriateSolverAnalyzerAndLog(Model model, double pseudoTimeStep, double pseudoTotalTime, int currentStep, int nIncrements): creates solution components for a given staggered solution step

internal void UpdatePressureDivergenceDictionary(Dictionary<int, double[][]> pressureTensorDivergenceAtElementGaussPoints, ISolver solver, IChildAnalyzer childAnalyzer, Model model, IAlgebraicModel algebraicModel): updates the dictionary containing data for a shared quantiy used by another equation
```

StaggeredAnalyzer Constructor

StaggeredAnalyzer(IParentAnalyzer[] analyzers, ISolver[] solvers, CreateNewModelDelegate createNewModel, int maxStaggeredSteps, double tolerance)

IParentAnalyzer[] analyzers: implement the iterative solution update procedure selected for each subproblem (Generalized a solution procedure for time dependent problems and typical Newton-Raphson solution method for static/quasi static problems)

ISolver[] solvers: implement a solution process for the linearized system of equations that emanates from the spatial discretization of each PDE

CreateNewModelDelegate createNewModel: a model array creation function that describes the sequence of update steps for the shared quantities and the model variables
int maxStaggeredSteps: is used as a termination criterion
double tolerance: another termination criterion