MULTI-PHYSICS MODELLING ENVIRONMENT FOR CONTINUUM AND DISCRETE DYNAMICS

A. V. Smirnov
Mechanical & Aerospace Engineering Department
West Virginia University
Morgantown, USA
email: asmirnov@wvu.edu

Abstract

A model-development environment for simulation of discrete and continuum dynamics in complex 3D geometries is described. This environment is based on 3D libraries for manipulation of geometrical primitives, object-oriented multi-domain modeling paradigm, continuum solvers on unstructured meshes, discrete particle solvers, and a tool-assisted grid generation technique. The numerical approach is based on combined control-volume and finite-element methods of continuum mechanics and Lagrangian particle dynamics method.

Several prototype example cases of complex continuum mechanical systems are considered, such as flow in bifurcating channels, including particle transport and deposition, flow interactions with flexible membraines and rigid structures, and a simplified model of molecular dynamics represented by a system of particles and bonds.

KEY WORDS
Multi-paradigm Simulation; Physically-based Modelling; Continuous & Discrete Methodology; 3-Dimensional Modelling

1 Background

In the past few years a clear trend has emerged toward the integration of different modelling paradigms of continuum dynamics, such as fluid dynamics and structural analysis. On the one hand, we witness a further expansion of discrete dynamics modelling in such areas as molecular dynamics, particulate and aerosol transport, etc. The complexity of modelling a mechanical system grows exponentially with the number of different processes involved. A computational approach that simultaneously accounts for all the phenomena related to an event is called *multi-physics* [1, 2]. This term is becoming increasingly popular and is used to refer to modelling of more than one phenomenon in both engineering and natural systems [3]. Multiphysics simulations become especially popular with a growing usage of parallel computer platforms where the modularity of the approach can be fully exploited [4, 5, 6].

Examples of multiphysics systems include composite materials [7], complex fluid dynamics [8], with phase transitions [9], electromagnetic and acoustical effects [6, 5], general multi-component systems [10] etc. The number of publications that refer to multi-physics or *multi-modelling* is growing rapidly, and the number of studies that deal with multiphysics modelling without directly mentioning the term is already overwhelming. Most of these studies are concerned with continuum systems, where different phenomena are modeled by means of partial differential equations (PDE) discretized in the manner appropriate to each particular model: finite element, control-volume, or finite-difference. There are usually two different classes of problems: one with different phe-

nomena sharing the same region of space, and another, where the phenomena occur in different regions of space, and their interaction takes place at the boundaries.

The approach to solve the first set of problems is based on designing flexible discretization schemes that would enable the construction of composite continuum solvers [11, 6, 10]. The second class of problems is solved by decomposing the space into domains with different physics and arranging for coupling between the domains. The corresponding techniques are usually referred to as *domain decomposition* [12, 13, 14], or *domain partitioning* [4].

However, there is a class of problems where the regions of space governed by different physics may change in time or where the distinction between the continuum and discrete dynamics becomes diffuse. Examples of such problems include: fluid structure interaction, liquid interface tracking, fracture and disintegration of solids, interaction of discrete and continuum phases, etc. It is common to use application-specific solution methods for such systems. Solvers and discretization schemes used for one problem are usually not suitable for other problems. However, a unifying approach in all the discretization schemes is to define a given set of variables on a discrete set of elements (cells or nodes of a computational grid, or points in space, etc.), and then to implement the solution methods in algorithms that manipulate the discretized variables.

In this study we use the idea of a generalized discretization element as a primitive geometrical object of a variable dimension and connectivity, and generalized variables as properties of the elements. With these generalizations an object-oriented approach was developed, which provides solutions to a broad class of mechanical systems on the basis of progressive associations and inter-dependencies of geometrical primitives (elements). The elements are enveloped in classes and supplied with functions that enable their spatial manipulations and dynamic interconnections. The elements can be arranged into domains, defining the geometry, and functions can be combined into solvers, providing physical content. Together domains and solvers constitute a set of models.

This multi-domain multi-model framework makes it possible to model in a unified and general manner the interactions between various physical objects, such as material points (particles, bubbles, etc.), segments (struts, tethers, etc.), surfaces (membranes, shells, etc.) and continuum media (solid bodies, flow/force fields, see Sec.4).

As pointed out above, the number of integrated environments for multiphysics modelling is constantly growing. However, to the knowledge of the author, the method proposed and implemented in this study offers two new features: flexible discretization strategy based on generalized elements, and a unified modelling framework. This framework combines the standard pre-processor, model-execution and post-processor steps into a single continuous simulation process. The approach is general enough to include the mesh generation procedure itself as just another model in the system.

2 Method

The top-down approach of the method starts with a concept of a *model*. Each model consists of a *domain* and a *solver* (Fig.1). Domains populate a physical 4D space-time and have one-to-many relationships to each other in form of geometrical intersections in the common regions of space-time. Each domain consists of a connected set of *geometrical primitives* or *elements* and variables defined on these primitives. The domain is also supplied with a collection of functions to manipulate the variables and the primitives. These functions are executed in a certain sequence specific for each domain, which constitutes a *solver*. Domains have one-to-one relationship to solvers, whereby there can be only one solver for each domain. Solvers may have one to many relationships to domains, where one solver can operate in several domains. Solvers can also have inclusive relationships to each other, where each solver can include one or several other solvers.

The bottom-up approach to the modelling paradigm starts with the concept of an *element* or a geometrical primitive. One property of an element is its spatial dimension, which can span

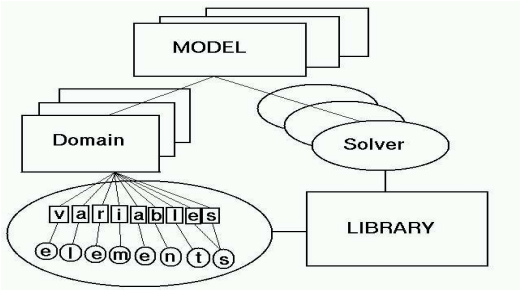


Figure 1: Modelling paradigm

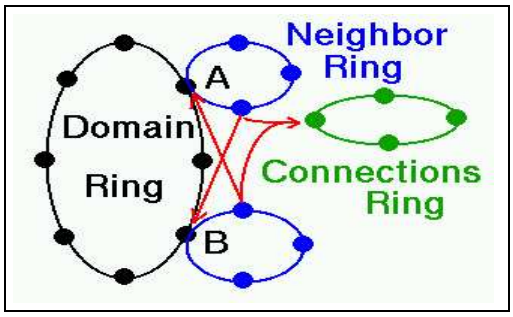


Figure 2: Dynamic connectivity of elements.

the range between 0 and 3. Another property is its type, like *point, node, edge, face*, and *cell*. In the current implementation there are two types of elements of dimension zero (points and nodes) and one type of element for each dimension higher than zero. Among the latter the faces are currently represented by triangles, and cells are represented by tetrahedrons. The elements can be connected to each other and to other elements. The connectivities between the elements are determined by their types and geometrical constraints. For example, points can have one-to-many connectivity to each other, one-to-one connectivity to the cells and faces (inclusion) and generally, no connectivity to the edges, etc. The possible connectivities are given in Table 1.

Table 1: Connectivity relationships between the elements

	points	nodes	edges	faces	cells
points	M	-	-	1*	1
nodes	-	-	M	M	M
edges	-	2	-	M	M
faces	M	3	3	3*	2
cells	M	4	4	4	4

* - only for domain-boundary elements

The connectivity matrix is asymmetric, which points to the asymmetry of element inclusion relationship. The one-to-many relationship (M) is usually represented by dynamically linked lists. It can also be seen from the table that the boundary elements of the domain may possess some extra connectivity constraints.

All elements are stored as linked lists, looped back on themselves (rings), and their connectivities are stored in the same type of lists (Fig.2). This provides high efficiency and flexibility in element creation/ destruction operations, as well as in modelling the element interactions. There are several functions defined inside the element-class, which realize operations on elements, such as creation/destruction, motion, connection, etc. These functions combine into an element library that is used to build domain specific solvers and models described below.

A 3D *mesh* is a connected set of elements, such as cells, faces, edges and nodes, excluding points. A mesh does not have to include elements of all types. So, a simple 3D mesh for the finite-element method can consist of connected set of cells and nodes only. Similarly, a 2D mesh can consist of faces, edges and nodes, or of faces and nodes only.

A *variable* is a property of an element which is changing in space-time, such as coordinates, velocities, mass, etc. Each element can contain a set of discrete or continuous variables of a general tensor type with a rank of 0 (scalar), 1 (vector), 2 (matrix), etc. For example, a finite-element incompressible Navier-Stokes solver may be defined on nodal elements with one variable of type scalar (pressure) and one of type vector (velocity). At the same time in a control-volume type formulation the pressure variable may be defined on the cells and velocity variable can be defined on the faces, providing a variant

of a staggered-grid formulation [15]. A node-cell arrangement with pressure defined at the nodes and velocity at cell-centers, with subsequent interpolation to the faces would represent still another possibility. This approach provides a great flexibility in the choice of numerical discretization schemes.

Domain is a connected set of elements and variables. A *solver* is a domain-specific sequence of operations on variables and elements. Element operations were introduced earlier, and variable operations include interpolation, differentiation, computing fluxes, etc. It is important to point out that the inclusion of element-operations into the solver extends the notion of the solver from a simple variable manipulator to domain manipulator. This not only allows domains with changing geometries, but also enables other complex modelling, like domain construction (grid-generators, tissue-growth, etc), and destructors (dissolution, fracture, collapse, etc).

A solver is based on a known *method*. For example, in the current implementation the solution of a Poisson equation on 3D tetrahedral meshes is realized using the finite-element method [16], and the solution of aerosol transport in bifurcating airways is realized using the control-volume method for unstructured meshes [15] together with the Lagrangian particle dynamics method [17]. Composite modelling can be accomplished by combining several models. For example, the current implementation of the incompressible Navier-Stokes solver is combined from the control-volume solver for convection-diffusion process and the finite-element Poisson solver. Such a solver has the advantage of using each method in its most appropriate context: a flux-conservative scheme for transport process, and a rigorous FEM scheme for the elliptic problem.

The creation of a model is divided into two steps: *domain setup* and *solver setup* (Fig.3). Domain setup is in turn split between: *geometry definition* and *variables definition*. Solver setup consists of two procedures: *initializer* and *iterator*. Each procedure is implemented in a high-level language (C++), and uses a set of library functions for manipulating geometrical primitives (elements) and performing operations on variables. The dependency lines in Fig.3 indicate that the variables are conditioned on the existence of elements and the iterator is conditioned on the initializer. During the program run initializers for all the models are called first, and then iterators of each model are executed one after another, or concurrently when in multiprocessor mode.

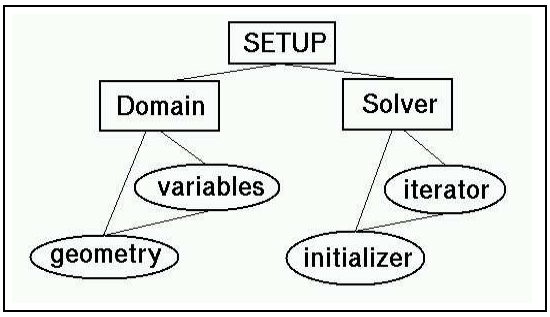


Figure 3: Model definition

The advantage of this scheme is that it provides both composite and distributed modelling environments, in which it is possible to combine different models into one, and also to assign different models to different regions of space. This feature can be of a great benefit in simulating complex multi-component systems, where different parts of the system are characterized by different physical processes.

The multi-domain feature of the system makes it easily parallelizable, with each domain or a group of domains assigned for execution by a separate process. A special algorithm of *domain coupling* (DOVE) [18] enables automatic setup of communication interfaces in the overlapping regions of domains. In contrast to similar approaches [4, 12, 13, 14], the DOVE technique can be used to track the connectivities of domains with changing geometries and moving with respect to each other.

Another feature of the modelling environment is the possibility of assigning scenarios for certain objects composed of groups of elements. These scenarios prescribe the sequence of positions for these objects and some features of interactivity with other elements. This way it is possible to simulate complex time dependent events where the behavior of certain elements of the system is pre-determined (Fig.6).

The integrated modelling-environment developed

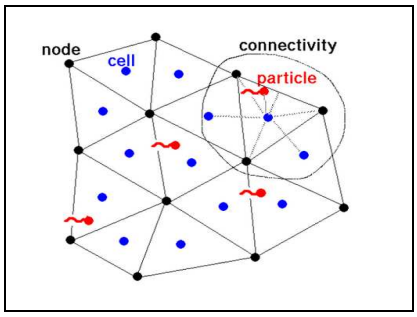


Figure 4: Connectivities in a multi-element framework.

in this work enables one to implement a unique visualization strategy, where there are no pre- and post-processor stages for data setup and visualization. Instead there is a single mode of monitoring and control of the state of the simulation via a graphic user interface (GUI). In this mode the information on geometry and variables for each model is displayed continuously, during all stages of model initialization and execution. Monitoring of three-dimensional domain data is done through the calls to 3D visualization routines built on top of the OpenGL graphics library (www.opengl.org).

3 Implementation

The unique feature of this approach not found in alternative multiphysics approaches [8, 4, 11, 9, 6] is the flexibility of associating variables of the problem with different elements of geometry, such as vertexes, edges, faces and cells (Fig.4). This provides an easy way of implementing control-volume (CV), finite-element (FE), and mixed control-volume/ finite-element methods [15, 16].

However, this flexibility comes at a cost of additional memory requirements needed to store the dynamic connectivities, when such are used in the model. The overall performance of the simulation will depend greatly on the implementation of the solver. The details of this implementation are not important for the purpose of this study as long as the solver uses the element class library. In this way any solver can take advantage of the dynamic interactivity between the elements, which makes it more versatile.

In our work we used explicit time marching schemes for the unsteady problems. A Jacobi or Gauss-Seidel scheme is employed in the Poisson FE solver. Implementation of accelerated solvers (CG, MG, etc.) is possible within this framework, and is planned as a future work.

3.1 Continuum solvers

There are two types of continuum solvers currently implemented in this framework: finite element Poisson solver, and a control-volume flow solver.

3.2 Finite element solver

The finite element solver currently implemented solves the boundary-value problem for the Poisson equation:

$$\Delta P(\mathbf{x}) = f(\mathbf{x})$$

$$P(\mathbf{x})\,|_{\mathbf{x}\in\partial_0 D} = f^0(\mathbf{x}) \qquad (1)$$

$$\frac{\partial P(\mathbf{x})}{\partial \mathbf{n}}|_{\mathbf{x}\in\partial_1 D} \equiv (\nabla P\cdot\mathbf{n})_{\mathbf{x}\in\partial_1 D} = f^1(\mathbf{x}) \qquad (2)$$

where $(*\cdot *)$ denotes a scalar product operation on two vectors, $\mathbf{x}$ represents a vector of coordinates $\mathbf{x} = \{x_i\}$, $i = 1:3$, $\mathbf{n}$ is the normal vector to the boundary $\partial_i D$ of the domain D with $i = 0$ representing Dirichlet and $i = 1$ - Neuman boundary.

After the finite element discretization is applied on tetrahedral meshes, the following algebraic system results:

$$\sum_{j=1}^{N} c_{ij}^1 p_j = -\sum_{j=1}^{N} c_{ij}^0 f_j$$

where the coefficients $c_{ij}^{0,1}$ are assembled in a loop over all the tetrahedral elements:

$$c_{ij}^0 = \sum_{k=1}^{N_e} \int_{E_k} \phi_i \phi_j d^3\mathbf{x} = \sum_{k=1}^{N_e} c_{\alpha_i \beta_j}^{0,k} \qquad (3)$$

$$c^1_{ij} = \sum_{k=1}^{N_e} \int_{E_k} \nabla\phi_i \nabla\phi_j d^3\mathbf{x} = \sum_{k=1}^{N_e} c^{1,k}_{\alpha_i\beta_j} \quad (4)$$

where N_e is the number of grid-elements or cells, and the subindexes α, β are now indexes of element vertexes corresponding to grid nodes i, j inside each elment. The coefficients $c^{*,k}_{\alpha\beta}$ are obtained by performing integration over each element k. For tetrahedral elements they can be computed in the following manner, that we provide here without derivation:

$$c^{0,k}_{\alpha\alpha} = \frac{1}{10} V_k \quad (5)$$

$$c^{0,k}_{\alpha\beta} = \frac{1}{20} V_k \quad (6)$$

$$c^{1,k}_{\alpha\beta} = \frac{(\mathbf{a}_\alpha \cdot \mathbf{a}_\beta)}{9 V_k} \quad (7)$$

where V_k is a volume of the element k and $\mathbf{a}_\alpha$ is the surface-normal area vector of the face opposite to node α.

After the coefficients of discretization are now well defined by relations (3) - (4), the solver applies an appropriate iterative procedure to solve the matrix system 3. It should be noted that in the case of a moving boundary problem the coefficients $c^{0,1}_{ij}$ should be recomputed at every iteration of the solver.

3.3 Control-volume solvers

Let's consider as an example the implementation of a control-volume solver for a compressible flow model. Brief examples related to other models are provided in Section 4. The governing equations are represented by mass and momentum conservation:

$$\dot{\rho} + (\rho U_i)_{,i} = 0 \quad (8)$$

$$\dot{U}_i + U_j U_{i,j} = \nu U_{i,jj} - \rho^{-1} P_{,i} \quad (9)$$

where ρ, U_i, P, ν are density, velocity, pressure and kinematic viscosity. System (8), (9) is not closed. A rigorous approach to close it would be to use a thermodynamic relation between pressure and density, such as an ideal gas law: $P = \rho RT$. However, this may introduce high-frequency acoustic modes into the problem, which in turn can slowdown the solution procedure considerably. To avoid this, usually a Poisson equation for pressure is formulated and solved. Currently both methods are implemented in the system.

In contrast to the FE method used for the solution of the Poisson equation, in CV flow solvers the location of the variables does not have to be vertex-based. On the contrary, a cell-centered scheme is often preferred [15]. Since the formalism described here allows an arbitrary association of variables to elements it enabled the implementation of various versions of CV method. A consistent approach is based on cell-centered pressure and face-centered velocity location, which corresponds to the computationally stable staggered grid arrangement [19]. However, on tetrahedral grids this discretization sometimes results in poorly formed control volumes for velocities. In addition to this a large number of faces on tetrahedral grids would require an excessive memory storage for face-centered vector variables. A more conventional approach involves placing the velocities at cell-centers and pressure at the cell vertexes, which will still retain the staggered nature of the discretization. In this case it is possible to use the FE procedure described above for the solution of the Poisson equation for pressure and the CV approach to handle the convective fluxes. This will result in a mixed finite element/ control volume formulation.

In some situations a completely vertex-based formulation will be of extra convenience, especially in problems with changing geometries. In this case, instead of solving the Poisson equation for pressure a pseudo-compressibility concept [20] can be employed. This approach proved to be especially convenient for tackling moving boundary problems and fluid-structure interaction effects [21] (see also Sec.4).

In problems involving turbulent flows and associated particle transport a sophisticated subgrid turbulence models are often used. One submodel of this type, RFG [23], is used in the current system to extend the flow solvers with turbulence modelling capabilities, and increase the accuracy of particle/bubble

transport simulations. A model like this is usually included as a sub-module for the host flow solver.

In the current framework it is possible to create composite solvers. Generally, when two sub-models are merged, a larger set of variables is created, which is a union of variables of the respective sub-models. The set of elements can likewise be extended if the respective sub-models operate on different elements. Whenever the variables of two models reside on the same elements and have the same names they are merged into a single variable. The solution procedures of each model will be applied to these variables in turn. If the variables are defined on different elements and are not connected by the model iterator they can in principle be solved concurrently. This amounts to a next level of parallelism that can be exploited on distributed memory systems.

3.4 Discrete phase solvers

From the point of view of multi-physics modelling pursued in this work the distinction between the continuum and discrete phases is rather vague. This is due to the fact that every element can be in motion and in interaction with other elements. A special case of a discrete phase - a particulate phase, including particles, droplets or bubbles, will be represented by zero-dimensional elements of type points or nodes. Thus points are considered to be discrete particles moving inside the mesh. Each point can have only one link to a single cell of the mesh and no links to other elements (Table 1). This makes points different from nodes, which are associated with the vertices of the mesh. In contrast to points, nodes can be shared by many cells, and consequently there is no special link set from nodes to cells, but rather cell-to-nodes links are used instead. Nodes can also create dynamic links to each other, which provides mechanisms for describing bonding in molecular dynamics, cytoskeletal structures in biological cells, struts in engineering structures, etc. (Sec.4).

While point-elements are mainly associated with discrete particulate phase, the nodes are used to represent both discrete and continuum properties

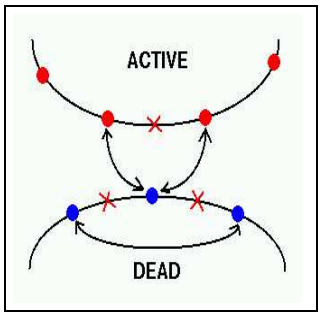


Figure 5: Controlling particle population by dynamic lists: reviving a dead particle.

depending on the application. The distinctive feature of discrete phase dynamics is the relative mobility and constantly changing connectivity of the elements that host the discrete model. Using dynamic connectivities between points, edges and cells in the modelling of the discrete phase enables the representation of a broad range of dynamical systems, the examples of which are shown in Sec.4.

The operation of creation and destruction of elements can be especially time consuming if memory allocation/deallocation is involved. In the case of discrete phase simulations it is important to do these operations efficiently, since particle creation/destruction events may be quite frequent. For the cases when the number of elements can change frequently a special dynamic population model is used where discrete elements are arranged in special *ring* lists (Fig.2). These lists have active and inactive (dead) areas. When element is created or destroyed it is simply moved from one area in the list to another, via pointer assignment operations (Fig.5) avoiding expensive dynamic memory calls. Particle dynamics is done in a Lagrangian framework and is based on Newton's equations of motion of the type:

$$m\frac{d\mathbf{V}}{dt} = F(\{P_j\}_{j=0,N_p}, \mathbf{U}) \qquad (10)$$

where m, $\mathbf{V}$ are mass and velocity of a particle, and F is a function of properties of other particles P_j, and other variables in the model, not necessary associated with particles, such as electromagnetic

fields, or a local flow velocity at particle's position ($\mathbf{U}$). This coupling between the discrete and continuum phases is usually done by means of interpolation of the continuum variables defined on the mesh to the particle positions inside each cell. The reverse effect of particles on the continuum variables can be computed in similar manner by means of accounting for particle contributions to each mesh-variable associated with the cell hosting a particle.

When particle-flow and particle-particle interactions are involved a special *probabilistic implicit interaction* scheme (PII) is used [17], which is characterized by a linear dependence of the execution time on the number of particles.

4 Applications

Currently the system has the following set of integrated models:

1. Flow models
   (a) Control-volume flow solver based on artificial compressibility concept (Sec.3.1) [22].
   (b) The random flow generation algorithm (RFG) for turbulent flow modelling developed by the author [23].
2. Particle models
   (a) Lagrangian models of particles and bubbles coupled with control-volume and RFG flow solvers (see items 1a,1b above) supplied with several different equations of motion [24, 25].
   (b) Particle interaction mechanisms based on PII scheme [17].
3. Tool-assisted mesh generation algorithm based on a tissue growth model [26]
4. Elastic membrane dynamics coupled with the flow solver (item 1a above).
5. FEM solver for the Poisson equation.
6. The dynamics and fracture of connected struts under gravity and inertial forces.
7. Molecular dynamics with bond formation.
8. Simple lattice structures (hexahedral, diamond, dodecahedral).

The modelling approach proposed here allows the building of composite solvers and executing of several models in a single simulation. Several examples below illustrate the flexibility of generating different models.

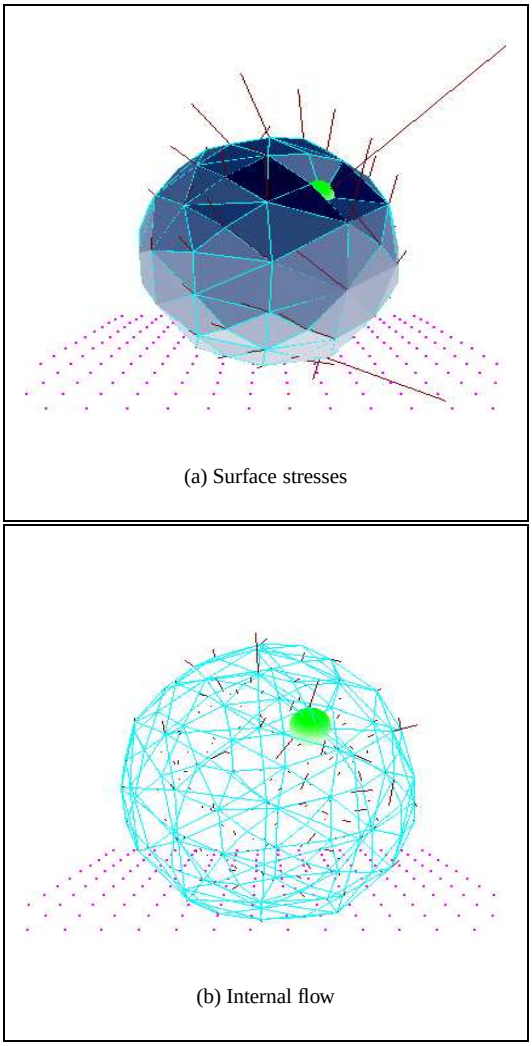
The example in Fig. 6 illustrates the application of a control-volume based flow solver formulated for unstructured meshes coupled with the finite-element elasticity solver for the membrane. The prototype simulation was capable of reproducing transient flow regimes and membrane deformations.

Figure 7 illustrates the mesh generation algorithm based on a *tool-assisted mesh generation* concept proposed by the author [26].

Figure 8 shows the process of particle transport in bifurcating ducts, and deposition on the walls. The simulation was done using a coupled flow-particle solver in Eulerian-Lagrangian formulation.

Figure 9 shows an example of a complex flow through overlapping domains. The domain coupling scheme enables the representation of topologically complex geometries, and can be used for parallel execution on distributed memory computer platforms.

Figure 10 shows an example of molecular dynamics simulation of atoms self-assembling around another group of atoms. Each atom is represented by a point-element with dynamic bonding to other elements. Two different types of atoms were used in the simulation: immobile atoms arranged in a Y-shaped structure representing a lattice, and mobile atoms that could interact with the atoms of the lattice and each other. The Lennart-Jones interaction forces of different types were prescribed to the different groups of atoms. After initializing the domain with the random distribution of the mobile atoms, the dynamics of the atoms results in an ordered arrangement, depending on the strengths and radii of the interaction potentials.



(a) Surface stresses

(b) Internal flow

Figure 6: Penetration of a solid object inside the elastic membrane filled with fluid

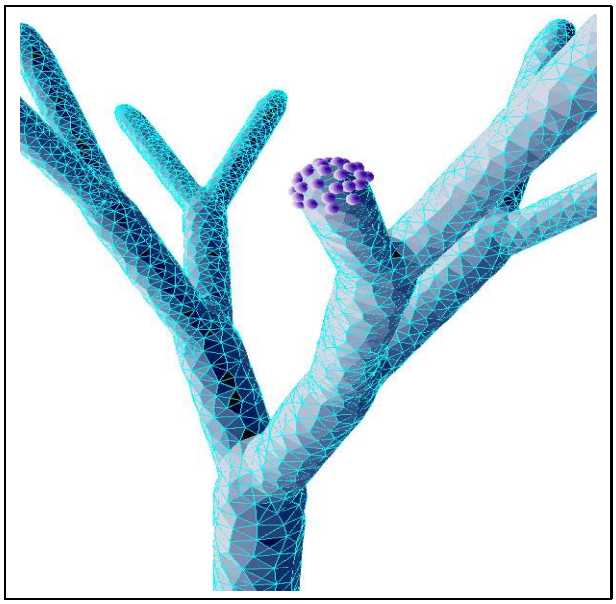


Figure 7: Generation of biological structures using tissue-growth model (mulphys.org/tam)

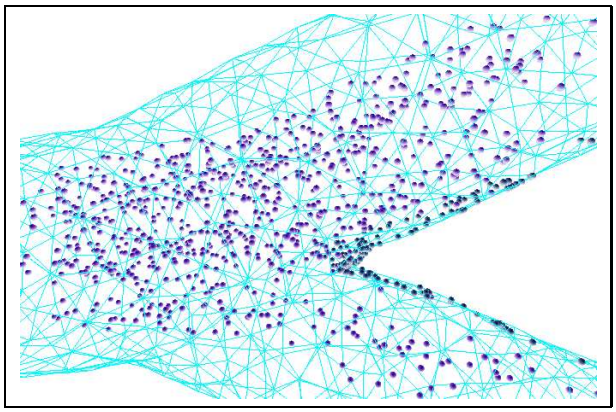


Figure 8: Particle transport and deposition in lungs (mulphys.org/biomed)

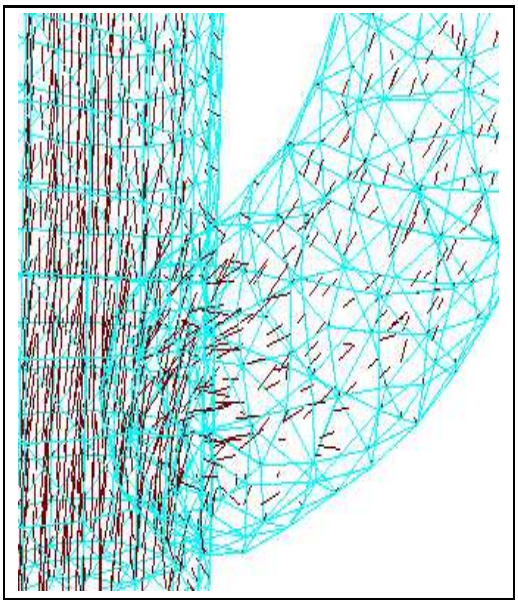


Figure 9: Flow through the coupled domains. http://mulphys.org/dove

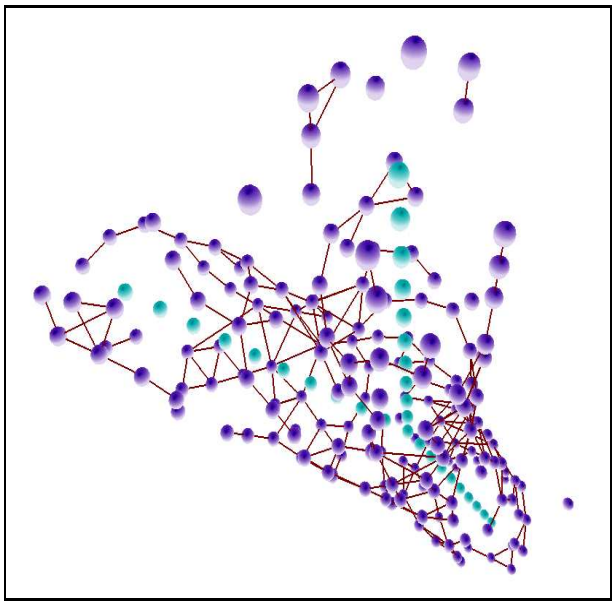


Figure 10: Self-assembly of atoms. http://mulphys.org/nano

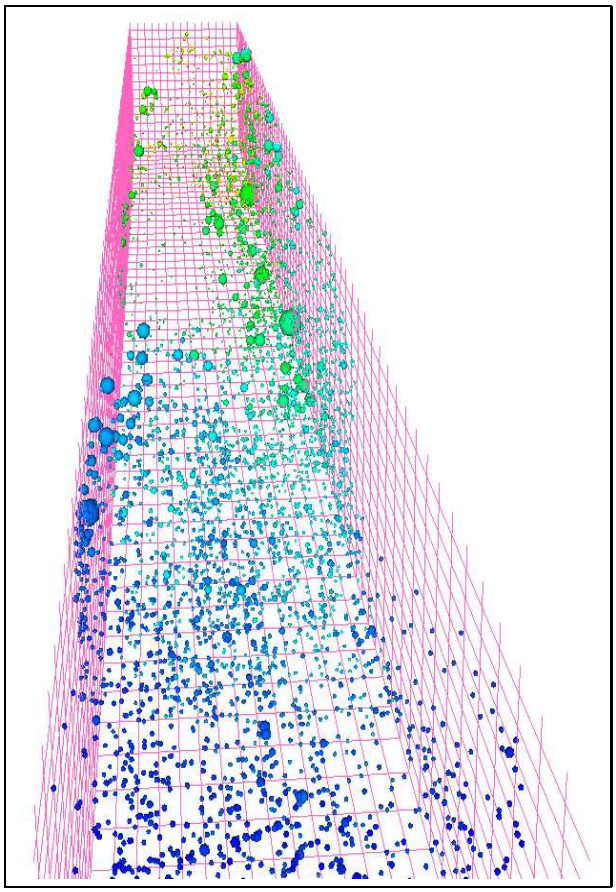


Figure 11: Coalescence of bubbles in a turbulent water stream (www.mulphys.org/particles).

Figure 11 shows the dynamics of coalescing bubbles in turbulent water stream computed with the PII scheme [17]. Other simulations involve particles in electromagnetic fields and colliding particle jets (mulphys.com/jets).

Figure 12 shows a fragment of a simulated building collapse. The model uses a concept of ”heavy” nodes connected by elastic struts. The braking of a strut can occur as its length or the angles it forms with other struts change beyond certain limits. Multibody interaction mechanism was implemented to enable the modelling of collisions of different previously not-connected elements of the structure as they col-

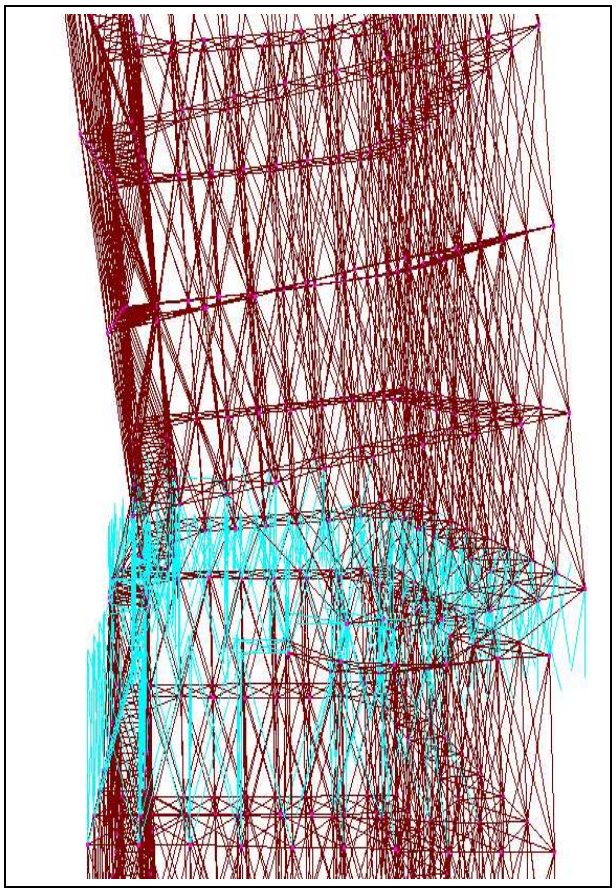


Figure 12: A damaged structure collapsing under its own weight (http://mulphys.org/collapse).

lapse toward the ground.

5 Conclusions

By creating a bridge between continuum and discrete dynamics the described approach enables modelling multi-phase flow systems, involving particle transport, deposition, and turbulence.

By using a multi-domain multi-model approach complex multi-physics modelling becomes possible with applications, involving fluid-structure interaction, fluid-acoustic, electromagnetic and radiative effects.

Based on a rich hierarchy of geometrical constructs and dynamic connectivities, the methods developed during this work can be successfully applied to the area of cellular mechanics, bio-mechanics and bio-fluids. A typical example is the problem of flexible tissue dynamics subjected to unsteady flow-fields.

In a broader perspective this approach has a potential to bridge together the paradigms of continuum mechanics and molecular dynamics in modelling sub-micron and nano-systems.

References

[1] Svante Littmarck. Math, models, motion, and more. *PT Design*, May 2000.

[2] Svante Littmarck. Solving differential equations. *The Industrial Physicist*, pages 21–23, February 2001.

[3] J. Thilmany. More than one force of nature. *Mechanical Engineering*, 124, 2002.

[4] M. McManus, K. andCross, C. Walshaw, S. Johnson, and P. Leggett. A scalable strategy for the parallelization of multiphysics unstructured mesh-iterative codes on distributed-memory systems. *International Journal of High Performance Computing Applications*, 14(2):137–174, 2000.

[5] S.M. Rifai, Z. Johan, W-P. Wang, T. J. R. Grisval, J-P. Hughes, and R. M. Ferencz. Multiphysics simulation of flow-induced vibrations and aeroelasticity on parallel computing platforms. *Computational Methods in Applied Mechanical Engineering*, 174(3-4):393–417, 1999.

[6] M. A. Troscinski. Real-world modeling keeps analysis honest. *Machine Design*, 68:66–8, 1996.

[7] J.P. Lemaitre. Multiphysics behaviors. In *Handbook of materials behavior models*, volume 3, pages xxvii, 1200. Elsevier Science & Technology Books, 2001.

[8] Magorzata Peszynska. Multiphysics coupling for two phase flow in degenerate conditions. In *Advanced techniques and algorithms for reservoir simulation: The IMA Volumes in Mathematics and its Applications*, volume 131, pages 21–39. Springer, New York, 2002.

[9] C. Bailey and S. Bounds. Multiphysics modeling and its application to the casting process. *Computer Modeling & Simulation in Engineering*, 4, 99.

[10] I. Yotov. A multilevel newton-krylov interface solver for multiphysics couplings of flow in porous media: Solution methods for large-scale non-linear problems. *Numer. Linear Algebra Appl.*, 8:551–570, 2001.

[11] C. Bailey, G. A. Taylor, M. Cross, and P. Chow. Discretisation procedures for multi-physics phenomena: Applied and computational topics in partial differential equations (gramado, 1997). *J. Comput. Appl. Math.*, 103(1):3–17, 1999.

[12] Barry Smith, Petter Bjørstad, and William Groupp. *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[13] Alfio Quanteroni and Alberto Valli. *Domain Decomposition Method for Partial Differential Equations*. Oxford Science Publications, 1999.

[14] Dihn, Q.V., Glowinski, R. and Periaux, J. Solving elliptic problems by domain decomposition methods with applications. In G. Birkhoff and A. Schoenstadt, editor, *Elliptic Problem Solvers II*. Academic Press, New York, 1984.

[15] J.H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer Verlag, 1997.

[16] P.M. Gresho and R.L. Sani. *Incompressible Flow and the Finite Element Method: Isothermal Laminar Flow*, volume 2. John Wiley & Sons, Ltd.; ISBN: 047149268X, 2000.

[17] A. Smirnov and I. Celik. A Lagrangian particle dynamics model with an implicit four-way coupling scheme. In *The 2000 ASME International Mechanical Engineering Congress and Exposition. Fluids Engineering Division*, volume FED-253, pages 93–100, Orlando, Fl, 2000.

[18] A.V. Smirnov. Domain coupling with the DOVE scheme. In *Parallel CFD 2003*, Moscow, Russia, 2003. Russian Academy of Sciences.

[19] S.V. Patankar. *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill, 1980.

[20] K.A. Hoffman and S.T. Chiang. *Computational Fluid Dynamics for Engineers*. Engineering Education System, Wichita, Kansas, 1993.

[21] A.V. Smirnov, W. Huebsh, and C. Menchini. A flow-solver with flexible boundaries. In *IASTED International Conference*, number 380-252 in Modelling and Simulation, Palm Springs, CA, 2003.

[22] K.A. Hoffman and S.T. Chiang. *Computational Fluid Dynamics for Engineers*, volume 1. Engineering Education System, Wichita, Kansas, 1993.

[23] A. Smirnov, S. Shi, and I. Celik. Random flow generation technique for large eddy simulations and particle-dynamics modeling. *Trans. ASME. Journal of Fluids Engineering*, 123:359–371, 2001.

[24] G. Sridhar and J. Katz. Drag and lift forces on microscopic bubbles entrained by a vortex. *Phys. Fluids*, 7(2):389–399, 1995.

[25] S. Elghobashi and J. Lasheras. Effects of Gravity on Sheared Turbulence Laden with Bubbles or Droplets. In *3-rd Microgravity Fluid Physics Conference*, Cleveland, OH, 1996.

[26] A.V. Smirnov. Tool assisted mesh generation based on a tissue-growth model. *Medical and Biological Engineering and Computing*, 41(4):494–497, 2003.