UNIFIED CONTINUUM MODELING OF 3D FLUID-STRUCTURE INTERACTION

JOHAN HOFFMAN, JOHAN JANSSON, MICHAEL STÖCKLI

Abstract. In this article we describe an incompressible Unified Continuum (UC) model in Euler (laboratory) coordinates with a moving mesh for tracking phase interfaces as part of the discretization, allowing simple formulation, modeling and implementation. The model consists of conservation equations for mass and momentum, a phase convection equation and a Cauchy stress $\sigma$ and phase $\theta$ variable as data for defining material properties and constitutive laws. We target realistic 3D turbulent fluid-structure interaction (FSI) applications, where we show simulation results of a flexible flag mounted in the turbulent wake behind a cube as a qualitative test of the method, leaving adaptive error control for future work. We compute piecewise linear continuous discrete solutions in space and time by a general Galerkin (G2) finite element method (FEM). We introduce and compensate for mesh motion by defining a local Arbitrary Euler-Lagrange (ALE) map on each space-time slab as part of the discretization, allowing a sharp phase interface given by $\theta$ on cell facets. The Unicorn implementation is published as part of the FEniCS Free Software system for automation of computational mathematical modeling. Simulation results are given for a 2D stationary convergence test, indicating quadratic convergence, an established 2D dynamic flag benchmark test, showing a good match to other implementations and a 3D turbulent flag test as indicated above.

1. Notation. We occasionally use an indexed Einstein notation with the derivative of a function $f$ with regard to the variable $x$ denoted as $D_x f$, and the derivative with regard to component $x_i$ of component $f_j$ denoted as $D_{x_i} f_j = \nabla f_j$. Repeated indices denote a sum: $D_{x_i} f_i = \sum_{i=1}^d D_{x_i} f_i = \nabla \cdot f$. Similarly we can express derivatives with regard to any variable: $D_u u = 1$.

2. Introduction. The context of this article is simulation with adaptive error control for realistic (3D) applications in continuum mechanics, where specific aims are space-time average force (such as drag) computation for flexible objects in turbulent flow or turbulent flow through flexible objects (such as blood vessels/hearts). To realize such aims we begin by developing a general modeling framework which we call Unified Continuum (UC) modeling, where we use conservation laws for mass, momentum and energy for a continuum and keep a stress $\sigma$ and phase variable $\theta$ as data for defining properties of the continuum, such as constitutive laws and material parameters. We evaluate the equations in the fixed actual (Euler or laboratory) coordinate system. At this stage we assume an incompressible continuum which simplifies modeling by decoupling the energy equation:

$$\begin{align*}
\rho(D_t u_i + u_j D_j u_i) &= D_{x_i} \sigma_{ij} + f_i \\
D_{x_j} u_j &= 0 \\
D_t \theta + D_{x_i} u_j \theta &= 0
\end{align*}$$

(2.1)

An example of fluid-structure constitutive laws can take the form:

$$\begin{align*}
\sigma &= \sigma_D - pI \\
\sigma_D &= \theta \sigma_f + (1 - \theta) \sigma_s \\
\sigma_f &= 2 \mu_f \varepsilon(u) \\
D_t \sigma_s &= 2 \mu_s \varepsilon(u) + \nabla u \sigma_s + \sigma_s \nabla u^T
\end{align*}$$

To open the possibility for adaptive error control we formulate a general Galerkin (G2) finite element method (FEM) [6] for computing discrete solutions of the UC model. The method includes aspects such as Galerkin space-time discretization, a local coordinate map based on mesh velocity (referred to as arbitrary Euler-Lagrange or ALE FEM) with mesh smoothing, and streamline diffusion stabilization. The discretization can be seen as a generalization and unification of the work described in [9] where we compute solutions of solid mechanics models in actual configuration/Euler coordinates with a moving mesh, the work on G2 for incompressible turbulent flow in [6], and the local ALE approach described in [3].

We describe a free software/open source implementation Unicorn as part of the FEniCS [4] software project for automation of computational modeling, with aspects such as abstraction of low-level finite element assembly functions through a high-level description in mathematical language,
Fig. 2.1. A flag mounted behind a cube in turbulent flow. The fluid-structure interface, an isosurface of the pressure and a cut of the mesh is plotted.

abstraction of 2D/3D-specific implementation and scientific repeatability of results and plots by full disclosure through free software/open source implementation.

We show several computational results:
- A 3D flexible flag in turbulent flow
- 2D stationary convergence test for displacement and drag
- 2D dynamic flag benchmark in laminar flow

3. State of the art. Mathematical modeling of fluid-structure interaction is a very active research field [13, 5, 8], with many important applications in society, such as the blood flow in a heart or a flexible blood vessel, airflow-induced vibrations in airplane wings, bridges and other structures and acoustic noise generation. The mathematical modeling of turbulent flow and non-standard material models are very challenging tasks in themselves, and the dynamics of the coupled system is a highly non linear problem that poses new serious challenges.

3.1. Related work.

3.1.1. Weak FSI methods. A straightforward and widely used method for solving a coupled fluid-structure interaction problem is to divide the domain into a fluid and structure domain with the interface as a boundary [10]. The following fixed-point iteration is then performed:

1. Assume some displacement velocities on the fluid-structure boundary to solve a fluid model for the velocity in the fluid domain
2. Compute the stress/force acting on the fluid-structure boundary.
3. Use the stress/force from the fluid solver as a boundary condition to solve a structure model in the solid domain for the velocity.
4. Compute the velocity of the fluid-structure boundary.
5. Check for convergence, otherwise go back to 1.

This method performs reasonably well in many cases, but there is no stability guarantee, and the method breaks down for certain cases and does not converge at all, though there can be ways to work around this [10]. Since there is no global continuum model or discretization it is also not clear how to perform stability analysis, for example with regard to conservation of momentum or mass, or global error estimation and adaptivity in a systematic and efficient manner.

3.1.2. Strong FSI methods. A more robust approach is presented in [13], [14] where separate continuum models for the fluid and structure respectively are formulated and discretized by the FEM. Instead of a fixed-point iteration, the degrees of freedom on the fluid-structure interface are coupled in the same algebraic system, with presumably a general stability with a Newton-type method.
3.1.3. Monolithic FSI methods. Another approach is to directly try to formulate the fluid and structure equations as one continuum model. The same FEM discretization can then be used for both phases, and a standard framework for stability analysis and error estimation/adaptivity can be employed.

Hron and Turek in [8] present such a monolithic method where an arbitrary Lagrange-Euler (ALE) argument is used on the continuum level to glue together a Lagrangian structure model and Euler fluid model to one continuum. This approach appears successful in stable computation and 2D results are presented, together with a 2D benchmark for comparison. A displacement variable is introduced in the continuum model to describe the deformation in the structure phase and the mesh smoothing in the fluid phase.

However, both the continuum model and discretization are rather complex. The global ALE approach introduces many mappings between different states for the conservation equations (different for fluid and solid) in addition to the stress mappings (Piola-Kirchhoff) for the structure, where computations are carried out in the original (undeformed) state. The discretization of the solution...
inherits the complexity of the continuum formulation, and is done with a stable choice of finite element. No 3D results are presented and no implementation is published, making it difficult to examine how the results are actually computed (or reproduce results).

Dunne and Ranacher in [2] present a different variant of a monolithic description based on an Euler coordinate formulation of a fluid-structure model. Here the continuum model consists of the conservation equations with a general stress variable, where also a displacement variable is necessary for the structure, and also an additional equation for interface capturing. The computational mesh is fixed, which leads to the fluid-structure interface intersecting cells. Thus an “initial position” set (IP) method for explicitly capturing the interface is needed, and is incorporated into the continuum model.

The discretization is done with a stabilized FEM, and an adaptive algorithm based on duality is developed. 2D results are presented and the Hron/Turek benchmark is computed with a reasonable match. Again, no 3D results are presented and no implementation is published.

4. Unified Continuum model. We develop an incompressible unified continuum model in a fixed Euler coordinate system consisting of:

- conservation of mass
- conservation of momentum
- phase convection equation
- constitutive laws for stress as data

where the stress is the Cauchy (laboratory) stress and the phase variable is used to define material data such as constitutive law for the stress and material parameters. Note that in this continuum description the coordinate system is fixed (Euler), and a phase function (marker) is convected according to the phase convection equation.

We start with conservation of mass, momentum and energy, together with a convection equation for a phase function \( \theta \):

\[
\begin{align*}
D_t \rho + D_x (u_j \rho) &= 0 \quad \text{(Mass conservation)} \\
D_t m_i + D_x (u_j m_i) &= D_x \sigma_{ij} \quad \text{(Momentum conservation)} \\
D_t e + D_x (u_j e) &= D_x \sigma_{ij} u_i \quad \text{(Energy conservation)} \\
D_t \theta + D_x u_j \theta &= 0 \quad \text{(Phase convection equation)}
\end{align*}
\]  

We define incompressibility as:

\[
D_t \rho + u_j D_{xj} \rho = 0
\]

which together with mass and momentum conservation gives:

\[
\begin{align*}
\rho (D_t u_i + u_j D_{xj} u_i) &= D_{xj} \sigma_{ij} \\
D_{xj} u_j &= 0
\end{align*}
\]

where now the energy equation is decoupled and we can omit it.

We decompose the total stress into constitutive and forcing stresses:

\[
D_{xj} \sigma_{ij} = D_{xj} \sigma_{ij} + f_i
\]

We can then pose constitutive relations between the constitutive (Cauchy) stress component \( \sigma \) and other variables such as the velocity \( u \).

Summarizing, we end up with the total UC formulation as given in 2.1. This continuum modeling framework is simple and compact, close to the formulation of the original conservation laws, without mappings between coordinate systems. This allows simple manipulation and processing for error estimation and implementation. It is also general, we can choose the constitutive laws to model simple or complex solids and fluids in interaction, with individual parameters.

4.1. Motivation. The motivation for the UC comes from observing the similarity between models for incompressible Navier-Stokes fluid flow and elasticity with a rate constitutive law. We have the same conservation of momentum for both phases (omitting convective terms):
\[ D_t u_i - D_{x_j} \sigma_{ij} = 0 \] (4.2)

and constitutive laws in similar form (omitting rotational terms for the structure):

\[
\begin{align*}
\sigma_f &= E_f \epsilon \\
D_t \sigma_s &= E_s \epsilon
\end{align*}
\] (4.3)

In discrete form over a time interval \( k \), we can write using quadrature:

\[
\begin{align*}
\sigma^1_f &= E_f \epsilon \\
\sigma^1_s &= \sigma^0_s + k E_s \epsilon
\end{align*}
\] (4.4)

where we can now view the structure part as a fluid with viscosity \( k E_s \) and a memory \( \sigma^0_s \) from integration of the time derivative.

5. Discretization. We choose a G2 FEM for discretization of the UC, where we begin by describing the standard FEM applied to the model to establish basic notation, and proceed to describe streamline diffusion stabilization and local ALE map.

5.1. Standard Galerkin. We begin by formulating the standard cG(1)cG(1) FEM \cite{3} with piecewise continuous linear solution in time and space for 2.1 by defining the exact solution:

\[ w = [u, p, \theta] \]

the discrete solution \[ W = [U, P, \Theta] \] and the residual \( R(W) = [R_u(W), R_p(W), R_\theta(W)] \):

\[
\begin{align*}
R_u(W) &= \rho(D_t U_i + U_j D_j U_i) - D_{x_j} \Sigma_{ij} - f_i \\
R_p(W) &= D_{x_j} U_j \\
R_\theta(W) &= D_t \Theta + D_{x_j} u_j \Theta
\end{align*}
\]

where \( R(w) = 0 \) and \( \Sigma \) denotes a discrete piecewise constant stress.

To determine the degrees of freedom \( \xi \) we enforce the Galerkin orthogonality \( (R(W), v) = 0, \forall v \in V_h \) where \( v \) are test functions in the space of piecewise linear continuous functions in space and piecewise constant discontinuous functions in time. We thus have the weak formulation:

\[
\begin{align*}
(R^u(W), v^u) &= (\rho(D_t U_i + U_j D_j U_i) - f_i, v^u_i) + (\Sigma_{ij}, D_{x_j} v^u_i) - \int_{t_{n-1}}^{t_n} \int_{\Gamma} \Sigma_{ij} v^u_i n_j ds dt = 0 \\
(R^p(W), v^p) &= (D_{x_j} U_j, v^p) = 0 \\
(R^\theta(W), v^\theta) &= (D_t \Theta + D_{x_j} u_j \Theta, v^\theta) = 0
\end{align*}
\]

for all \( v \in V_h \), where the boundary term on \( \Gamma \) arising from integration by parts vanishes if we assume a homogenous Neumann boundary condition for the stress \( \Sigma \).

This standard finite element formulation is unstable for convection-dominated problems and due to choosing equal order for the pressure and velocity. Thus we don’t use the standard finite element formulation by itself but proceed to a streamline diffusion stabilization formulation, and a local ALE discretization for handling the phase interface.

5.2. Local ALE. If the phase function \( \Theta \) has different values on the same cell it would lead to an undesirable diffusion of the phase interface. By introducing a local ALE coordinate map \cite{3} on each discrete space-time slab based on a given mesh velocity (i.e. the material velocity of one of the phases) we can define the phase interface at cell facets, allowing the interface to stay discontinuous. We describe the details of the coordinate map and its influence on the FEM discretization in the appendix.

The resulting discrete phase equation is:

\[ D_t \Theta(x) + (U(x) - \beta_h(x)) \cdot \nabla \Theta(x) = 0 \] (5.1)
We thus choose the mesh velocity $\beta_h$ to be the discrete material velocity $U$ in the structure part of the mesh (vertices touching structure cells) and in the rest of the mesh we use mesh smoothing to determine $\beta_h$ to maximize the mesh quality according to a chosen objective, alternatively use local mesh modification operations (refinement, coarsening, swapping) on the mesh to maintain the quality [1]. Note that we still compute in Euler coordinates, but with a moving mesh.

5.3. Streamline diffusion stabilization. For the standard FEM formulation of the model we only have stability of $U$ but not of spatial derivatives of $U$. This means the solution can be oscillatory, causing inefficiency by introducing unnecessary error. We instead choose a weighted standard Galerkin/streamline diffusion method of the form $(R(W), v + \delta R(v)) = 0, \forall v \in V_h$ (see [3]). We also make a simplification where we only introduce necessary stabilization terms and drop terms not contributing to stabilization. For the UC model the stabilized method thus looks like:

\[
\begin{align*}
(R^u(W), v^u) &= (\rho(D_t U_i + U_j D_j U_i - f_i, v_i^u) + (\Sigma_{ij}, D_x j v_i^u) + SD^u(W, v^u) = 0 \\
(R^p(W), v^p) &= (D_x j U_j, v^p) + SD^p(W, v^p) = 0
\end{align*}
\]

for all $v \in V_h$, and:

\[
\begin{align*}
SD^u(W, v^u) &= \delta_1(U_j D_j U_i, U_j^u D_j v_i^u) + \delta_2(D_x j U_j, D_x j v_j^u) \\
SD^p(W, v^p) &= \delta_1(D_x, P, D_x, v^p)
\end{align*}
\]

6. Constitutive laws. The UC model allows us to choose a different constitutive law describing the behaviour of that particular material for each phase.

6.1. Fluid laws.
We consider here only Newtonian fluids for simplicity, where we thus only have one law. We expect non-Newtonian fluids to be compatible with the UC framework and Unicorn implementation where they could be seen as relatives of viscous and plastic solid constitutive laws as given below.

- For a fluid phase we typically choose a Newtonian law: $\sigma = 2\nu \epsilon - pI$

6.2. Solid laws.
For a solid phase there exists a multitude of choices for constitutive laws. We list several possible laws, and describe the framework for adding new laws in the actual, deformed state of the continuum. The primitives for describing laws is the deformation gradient $F$ and the velocity $u$. We present derivations of relations between $u$, $F$ and $\sigma$ in the appendix which we use to construct constitutive laws. The main relations are summarized as:

\[
\begin{align*}
D_t F &= \nabla u F \\
D_t F^{-1} &= -F^{-1} \nabla u \\
B &= FF^T
\end{align*}
\]  

(6.1)

Using the above relation to compute $F$ we are then free to express constitutive laws coupling the stress $\sigma$ to the deformation $F$, or typically in the form $B = FF^T$. We can also further eliminate $F$ and formulate stress rate laws only in terms of the stress $\sigma$ and the velocity $u$. We present some possible choices, with extension to plasticity through a stress rate law:

- A common example is a Neo-Hookean [7] [2] law: $\sigma = \mu B - pI$
- Selecting the component $\sigma_{D} = \mu B$ and differentiating with regard to time we can eliminate $B$ (see appendix) and express $D_t \sigma_{D} = 2\mu (u) + \nabla \sigma + \sigma \nabla u$.
- A (compressible) elasto-plastic variant [9] of this model is: $D_t \sigma + \nu^{-1} (\sigma - \pi \sigma) = E\epsilon(u)$, where $\nu$ is a viscosity coefficient and $\pi \sigma$ denotes the projection of $\sigma$ onto a (convex) set of plastically admissible stresses.

7. Discrete system. We use a block-diagonal Newton method, where we start by formulating the full Newton method and then drop terms. We also use the constitutive law as an identity to remove instability caused by iterating between $\sigma$ and $U$.

We consider Newton’s method as a linearization of the continuum model, where we then compute the discretization of each successive iteration. In general we can consider equations of the type
\( F(U) = -D_t U + G(U) \) where \( G \) can include differential operators in space. Newton’s method is then: \( F'(U_0)(U_1 - U_0) = -F(U_0) \) or equivalently: \(-D_t U_1 + G'(U_0)(U_1 - U_0) = -G(U_0)\).

We formulate Newton’s method for the system \( F(X) = (F_u(X), F_\sigma(X), F_p(X))^\top = 0 \), with \( X = (U, \sigma, P)^\top \):

\[
\begin{align*}
F_u &= -D_t U + \nabla \cdot \sigma + U \cdot \nabla U + \nabla P \\
F_p &= -\Delta P + \nabla \cdot U \\
F_\sigma &= -D_t \sigma + \mu \epsilon(U) - \sigma \nabla v - \nabla w \sigma
\end{align*}
\]

We compute elements of the Jacobian matrix \( J \):

\[
(D_U F_u)w = -D_t w + (U \cdot \nabla) w + \nabla U \cdot w + k \mu \nabla \cdot \epsilon(w) - k \nabla \cdot (\sigma \nabla w + \nabla w \sigma)
\]

... where we used the identity \( \sigma = \text{sigma}_0 + k \mu \epsilon(U) + k \sigma \nabla U + k \nabla U \sigma \)

Newton’s method thus becomes:

\[
\begin{align*}
X^1 &= X^0 - J^{-1} F(X^0) \\
J \Delta X &= -F(X^0)
\end{align*}
\]

with \( \Delta X = X^1 - X^0 \).

Since \( J \) is block-diagonal we are left with solving a linear system for each component of \( F \), and a fixed-point iteration between components.

By using the constitutive identity in the \( D_U F_U \) computation for exchanging the \( \sigma \) term, we are left with an iteration between \( P \) and \( U \). This gives the time step limit: \( k < \min(h) / \max(U) \) observed in computation and consistent with a dimensional analysis of the fixed-point iteration.

**8. Elastic mesh smoothing.** The mesh velocity \( \beta_h \) has to satisfy several requirements:

1. \( \beta_h = U \) in the solid phase part of the mesh.
2. Bounded mesh quality \( Q \) in the fluid part of the mesh. Preferably the mesh smoothing should improve \( Q \) if possible.
3. Maintain mesh size \( h(x) \) close to \( \hat{h}(x) \) given by adaptivity.

We briefly describe a mesh smoothing method we’ve used in the presented results based on an elastic model. We formulate a simplistic variant of the UC model where we only consider a solid, and we omit the incompressibility equation. We use a constitutive law \( \sigma = E(I - (FF^\top)^{-1}) \) where we recall \( F \) as the deformation gradient. We use the update law: \( D_t F^{-1} = -F^{-1} \nabla u \) where we thus need an initial condition for \( F \). We set the initial condition \( F_0 = \hat{F} \) where \( \hat{F} \) is the deformation gradient with regard to a scaled equilateral reference cell, representing the optimal shape with quality \( Q = 1 \).

Solving the elastic model can thus be seen as optimizing for the highest global quality \( Q \) in the mesh. We also introduce a weight on the Young’s modulus \( E \) for cells with low quality, penalizing high average, but low local quality over mediocre global quality. We refer to the source code for more details, and leave deeper analysis and testing to future work.

As an alternative to mesh smoothing we can in the future consider using local mesh modification operations (refinement, coarsening, swapping) on the mesh to maintain the quality [1].

**9. Implementation in Unicorn/FEniCS.** FEniCS is a Free Software project for automation of computation, providing automated evaluation of variational forms together with solution of discrete systems.

We implement the G2 discretization of the UC in a general interface for time-dependent PDE where we give the forms \( a(U, v) = (D_U F_U, v) \) and \( L(v) = (F_U, v) \) for assembling the linear system given by Newton’s method for a time step in figure 9.1. The language used is Python, where we use the FFC form notation from FEniCS.

**10. Simulation results.** We present results from running the Unicorn implementation with data for 3 different problems:

1. a 2D stationary convergence test for verifying convergence of solution output with regard to mesh size
2. a 2D dynamic flag benchmark for comparing results with other methods and implementations
3. a 3D turbulent flag problem representing a realistic problem we can expect in real-world applications
10.1. 2D stationary convergence test. We seek to verify the correctness of the implementation by performing a convergence test with regard to displacement $U$ of a point $p$ in the structure. We define a geometry as illustrated in figure 10.2, where $p$ is the bottom right corner point, and set an inflow velocity in the left side of the square to $(1, 0)$. We run until until time $T = 60$ where we observe a steady state.

We start with a uniform mesh and run one simulation for each level of a uniform mesh refinement using the Rivara algorithm [11], where we split the longest edge of each cell in half. We choose the displacement $U_{\text{ref}}$ on the finest mesh as reference and plot the log($h$)/log($U_{\text{ref}} - U$) in figure 10.4 to observe an approximate convergence rate in $h$. We can see that the output converges with $h$, and that the convergence rate is approximately 2 from a curve fit, which is a reasonable expectation of the method for this problem.

We also run test for the solid model only where we specify zero inflow velocity and apply a force along the x-axis, with everything else being similar to the fluid test. The convergence results can be seen in figure 10.3.

For more details of the convergence tests we refer to the source code.
10.2. 2D dynamic flag benchmark. Since there exists a multitude of methods and implementations for fluid-structure interaction it’s interesting to make comparisons on the same benchmark problem. We choose a benchmark by Turek, et. al. [8]: FLUSTRUK-A, variant 3 which is 2D flow around a fixed cylinder (circle) with a thin flexible bar attached to the downstream side of the cylinder, see illustrations in figure 2.2. However, we make a modification in the benchmark in the constitutive model. The benchmark specifies a compressible constitutive model for the structure, where we only have incompressible models. We thus choose the Neo-Hookean model with matching material parameters, corresponding to choosing Poisson’s ratio $\nu = 0.5$ instead of $\nu = 0.4$ as specified in the benchmark.
Again, we perform uniform mesh refinement using the Rivara algorithm where we split the longest edge of each cell in half.

The mean drag for the finest refinement level is 461N and the y-displacement in the oscillation regime varies between 0.0353m and -0.332m. This is just 1-2% percent difference from the Hron/Turek results, i.e. a reasonable match, given that the constitutive model is different. However, the drag variation is significantly (ca. a factor 2) lower than the benchmark, we have not found the reason for this.

10.3. **3D turbulent flag.** We are targeting realistic applications of the UC model and Unicorn implementation which thus includes flexible structures interacting with turbulent flow in 3D. We define
a model problem with these properties consisting of a fixed cube in turbulent flow with a thin flexible flag mounted in the downstream wake. We choose an inflow (mean flow) speed of 1.0e2 m/s, a cube of 1.0e-2 m side and a flag mounted at the top of the back face of the cube with a length of 3.0e-1 m and a thickness of 5.0e-2 m. A viscosity of the fluid of 1.0e-4 Pa s (density $\rho = 1$) gives a representative Reynolds number $Re \approx 1.0e5$. We choose no-slip boundary conditions on the cube and flag with slip boundary conditions on the surrounding box walls, and a zero pressure outflow condition.

Snapshots of the simulation output can be seen in figure 10.1, where we observe violent bending and torsion motion along the long axis of the flag.

11. Conclusion and Future Work. We have formulated an incompressible Unified Continuum model targeting realistic 3D continuum mechanics applications including turbulence and fluid-structure interaction. The model is described in Euler coordinates and consists of conservation equations for momentum and mass and a phase convection equation where we keep a stress $\sigma$ and phase variable $\theta$ as data for determining material constitutive laws and properties.

We compute discrete solutions with the General Galerkin method where we introduce a mesh motion to track phase interfaces and mesh smoothing based on elasticity, still computing and evaluating quantities in actual, Euler coordinates. We publish the Unicorn implementation as part of the FEniCS Free Software project for automation of computational mathematical modeling, where we define variational forms representing a space-time discretization, which then automatically generate source code for assembling the discrete system.

We demonstrate three main simulation results with the UC and fluid-structure data: a basic 2D stationary convergence test indicating a quadratic convergence rate, a dynamic 2D benchmark test indicating a reasonably good match with previously published solutions and a 3D turbulent test showing qualitatively features of strong interaction between a flexible structure and turbulent flow, indicating possibility for realistic application.

The UC and the Euler coordinates with moving mesh together with rate laws for stress/deformation makes modeling intuitive and simple and the diagonal Newton/segregated fixed-point treatment of
the discrete system makes implementation simple and allows straightforward addition of alternative constitutive laws.

Future directions are:
1. Duality-based adaptivity for turbulent FSI, where we plan to follow a similar direction to [6] and [2].
2. Using local mesh operations (refinement, coarsening, swapping) with the MeshAdapt package from [1] as a more general alternative to mesh smoothing to handle large rotations/translations.
3. Modeling self-contact for the structure with a similar model as given in [9].
4. Implementation efficiency profiling and optimization and parallelization in connection with the primitives given by the FEniCS interface.

12. Appendix. We use an appendix for containing ancillary derivations which would otherwise occlude the presentation of the main results.

12.1. Constitutive modeling. We define the position of a particle in the continuum as the function $P(t, X)$, where we identify particles by their original position $P(t_0, X) = X$, and where the velocity is $u(t, P) = D_t P(t, X)$.

We seek to remove the dependence on $X$, to avoid having to map back and forth between an actual state and an original state in $X$. We express the deformation gradient $F$ in terms of velocity $u$:

$$D_t F(t, P(t, X)) = D_t D_X P(t, X) = D_X u(t, P(t, X)) = D_P u(t, P(t, X)) P(t, X) = \nabla u(t, P(t, X)) F(t, P(t, X))$$

where $\nabla u(t, P)$ is the gradient of $u(t, P)$ with respect to $P$.
We can thus compute the deformation gradient \( F \) in the actual state according to:

\[
D_t F = \nabla u F \\
D_t F^{-1} = -F^{-1}\nabla u
\]

(12.1)

12.2. Equivalence of rate and non-rate models. We show here for reference the equivalence of the original stress formulation of the Neo-Hookean model and its stress rate formulation, where we eliminate the deformation gradient \( F \).

We recall the Neo-Hookean constitutive model: \( \sigma_D = \mu(B - I) \). We take the time derivative using \( D_t B = \nabla u B + B \nabla u^T \):

\[
D_t \sigma_D = \mu(\nabla u B + B \nabla u^T).
\]

(12.2)

and using \( B = \mu^{-1}\sigma_D + I \) we get the rate constitutive model:

\[
D_t \sigma_D = \mu 2\epsilon(u) + \nabla u \sigma + \sigma \nabla u^T
\]

(12.3)

Since the forms are equivalent we are thus free to choose.

12.3. Coordinate systems. When evaluating a quantity \( q(t, x) \) in a continuum there are two main coordinate systems to consider: an Euler coordinate system is fixed in space and the continuum is moving, and a Lagrange coordinate system is defined by the motion of the continuum (see for example [3]). We can define the position of a particle in the continuum as the function \( \phi(t, X) \) given by:
where $u$ is the velocity of the continuum. 12.4 thus gives a map from material Lagrange coordinates $X$ to the fixed Euler coordinates $(t, X) \rightarrow (t, x) = (t, \phi(t, X))$. We can then define a mapped quantity $Q(t, X) = q(t, x) = q(t, \phi(t, X))$.

From these definitions, we can see that using Lagrange coordinates means that we evaluate quantities $Q(t, X)$ on the original (undeformed) state of the continuum, while using Euler coordinates, we evaluate quantities $q(t, x)$ on the actual (deformed, or laboratory) state of the continuum.

The consequence of choosing a Euler coordinate system is that we can then use the Cauchy (laboratory) stress, which we cannot in the Lagrange coordinate system, instead we must compute with a mapped stress.

In the presented work we evaluate quantities in Euler coordinates and use a local ALE map as part of the discretization to be able to introduce and compensate for a mesh velocity (moving mesh).

### 12.4. Local ALE

We define a local ALE coordinate map as part of the discretization, where it’s used to be able to introduce a mesh velocity. We still compute with global Euler coordinates, but with a moving mesh.

The $cG(1)cG(1)$ space-time FEM discretization gives a space-time “slab” $T_n \times I_n$ with $I_n = (t_n-1, t_n)$ and $T_n$ a triangulation of $\Omega$ for each time step composed of the mesh cells over one time step.

To be able to define and compensate for an arbitrary mesh velocity $\beta_h$ we define a local coordinate map $\phi$:

\[
\begin{align*}
D_t \phi(t, \bar{x}) &= \beta_h(t, \bar{x}) \\
(x, t) &= \phi(\bar{x}, t)
\end{align*}
\]

(12.6)

Application of the chain rule gives the relation:

\[
D_t U(x) + U(x) \cdot \nabla U(x) = D_t \bar{U}(\bar{x}) + (\bar{U}(\bar{x}) - \beta_h) \cdot \nabla \bar{U}(\bar{x})
\]

(12.7)

Space-time integration in a FEM gives (choosing simple quadrature for clarity):

\[
\begin{align*}
x_n &= F(\bar{x}) = \bar{x} + (t_n - t_{n-1})\beta_h(\bar{x}) \\
J &= D_x F \\
\int_K u(t, x) dx dt &= \int_K \bar{u}(t, \phi(t, \bar{x}))[\det J] d\bar{x} dt
\end{align*}
\]

(12.8)

We approximate $F = I$ in the integration and plan to account for this in a posteriori error estimation. $F$ is readily computable and $\|F - I\|$ is typically small.

Choosing $\beta_h = U$ in the solid part of the mesh gives a trivial solution of the phase convection equation, and we can remove it from the system.

### REFERENCES


