DOI:
10.1137/15M1014097

Document Version
Peer reviewed version

SIAM J. SCI. COMPUT.
Vol. 0, No. 0, pp. 000–000

MULTIPHYSICS COMPUTATIONAL MODELING IN *C*Heart*

J. LEE†, A. COOKSON†, I. ROY†, E. KERFOOT†, L. ASNER†, G. VIGUERAS†, T. SOCHI†, S. DEPARIS‡, C. MICHLER†, N. P. SMITH§, AND D. A. NORDSLETTEN¶

Abstract. From basic science to translation, modern biomedical research demands computational models which integrate several interacting physical systems. This paper describes the infrastructural framework for generic multiphysics integration implemented in the software ***C*Heart**, a finite-element code for biomedical research. To generalize the coupling of physics systems, we introduce a framework in which the geometric and operator relationships between the constituent systems are rigorously defined. We then introduce the notion of topological interfaces and define specific operators encompassing many common model coupling requirements. These interfaces enable the evaluation of weak form integrals between mesh subregions of arbitrary finite-element bases' orders, types, and spatial dimensions. Equation maps are introduced which provide abstract representations of the individual physics systems that can be automatically combined to permit a monolithic matrix assembly. Flexible solution strategies for the resulting coupled systems are implemented, permitting fine-tuning of solution updates during fixed point iterations, and subgrouping where several problems are being solved together. Partitioning of coupled mesh domains for optimal load balancing is also supported, taking into account the per-processor cost of the entire coupled problem within the graph problem. The demonstration of the performance is illustrated through important real-world multiphysics problems relevant to cardiac physiology.

Key words. cardiac modeling, multiphysics, parallel computing, coupled problems, modeling software

AMS subject classifications. 92-04, 92-08, 74F10, 74S05, 76M10, 68W10, 92C30

DOI. 10.1137/15M1014097

1. Introduction.

Computational modeling of biophysical phenomena has become a common approach for investigating problems that arise in medical sciences. From subcellular processes [22, 86] to whole-organ function [65, 81], mathematical models continue to demonstrate their capacity to predict physiology, provide added insight, and address questions of both scientific and clinical interest. The demand for solving computational models has led to the development of a variety of software packages [82, 83, 43, 74, 59, 24, 52, 73, 47, 71, 10, 75]. The success of many of these tools stems from their ability to offer efficient and scalable computation for specific physics problems. These packages widely employ finite-element methods

*Submitted to the journal's Software and High-Performance Computing section March 27, 2015; accepted for publication (in revised form) March 28, 2016; published electronically DATE. This work was supported by the British Heart Foundation (NH/11/5/29058), the Engineering and Physical Sciences Research Council (EP/G0075727/2), and the European Commission funded euHeart project (FP7-ICT-2007-224495:euHeart). It was also supported by the Wellcome Trust-EPSRC Centre of Excellence in Medical Engineering (WT 088641/Z/09/Z) and the NIHR Biomedical Research Centre at Guy's and St.Thomas' NHS Foundation Trust and KCL. The views expressed are those of the authors and not necessarily those of the NHS, the NIHR, or the DoH.

http://www.siam.org/journals/sisc/x-x/M101409.html

†Department of Biomedical Engineering, King's College London, London, UK (jack.lee@kcl.ac.uk, andrew.cookson@kcl.ac.uk, ishani.roy@kcl.ac.uk, eric.kerfoot@kcl.ac.uk, liya.asner@kcl.ac.uk, guillermo.vigueras_gonzalez@kcl.ac.uk, taha.sochi@kcl.ac.uk, christian.michler@kcl.ac.uk).

‡Mathematics Institute of Computational Science and Engineering, EPFL, Lausanne, Switzerland (simone.deparis@epfl.ch).

§Department of Biomedical Engineering, King's College London, London, UK, and Faculty of Engineering, University of Auckland, Auckland, New Zealand (np.smith@auckland.ac.uk).

¶Corresponding author. Department of Biomedical Engineering, King's College London, London, UK (david.nordsletten@gmail.com).

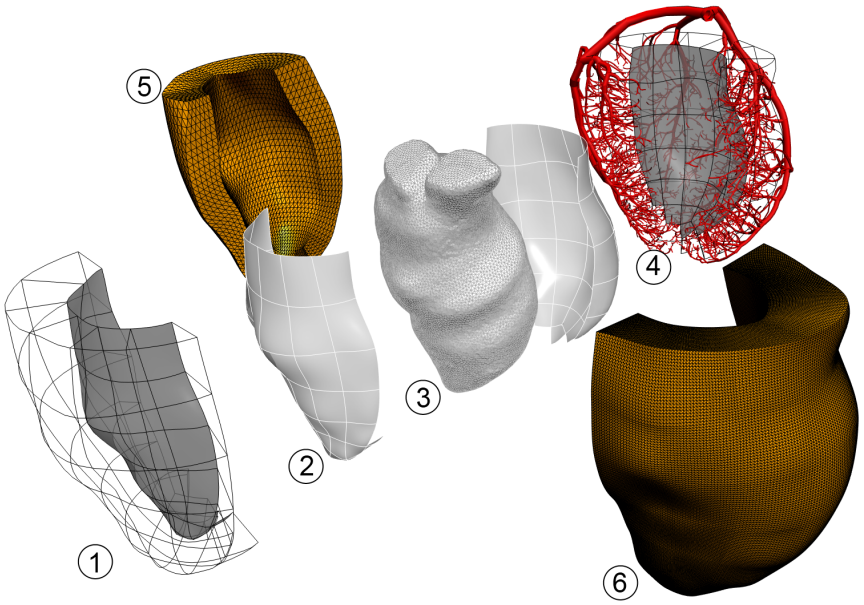


FIG. 1.1. *An example of finite-element meshes used for multiphysics simulations. Illustrated is the coupling between overlapping domains (hexahedral* (1) *and tetrahedral meshes* (5, 6) *for solid mechanics, monodomain, and poroelasticity problems) and over partially overlapping domains (endocardial* (2) *and cavity mesh* (3) *for fluid-structure interaction (FSI) problems and vascular mesh* (4) *for the coronary flow model).*

(FEMs) for numerical approximation of differential equations and have introduced a range of techniques that span the computational spectrum. From support for low order h-refined techniques [82, 83], to high order p-FEMs [43], to varied mixed methods [24, 10, 33] and h-adaptive techniques [73, 52], diverse approaches for problem solving have been enabled. A number of software packages also utilize mesh partitioning (such as ParMETIS [44], Scotch [72], or Zoltan [19]) and parallel linear solver libraries (such as Trilinos [34] and PETSc [5, 4, 6]) for efficient multicore parallelization. These strategies for scability are necessary to address the computational challenges presented by many mathematical models (e.g., hemodynamics [74, 43, 52, 24] and electrophysiology [59, 82, 83, 73]).

Despite these advances, the seemingly simple task of multiphysics modeling by integrating existing computational models remains challenging. For a general coupling between arbitrary problems, many aspects of the individual problem—including numerical discretization and solution schemes, disparate solution domains, and data accessibility issues—have to be formalized, presenting new implementational challenges. For instance, in the context of cardiac modeling a variety of numerical techniques are employed (see Figure 1.1), spanning from the one dimensional (1D) network manifolds used in vascular flow modeling [79], to curvilinear high order elements used in mechanics [61], to extremely refined tetrahedral meshes used in electrophysiology [62]. The resulting equations also require the ability to solve large scale linear systems as well as highly nonlinear systems. The varied nature of these physical models leads to many different forms of coupling, where integrated models may require equating kinematics/force (as found in fluid-solid interaction), mass transport/force (as found in poromechanics or multilevel transport), or strain/tension (as found in electromechan-

ics) between models on heterogeneous domains. The permutation of these factors has made the generalized coupling of multiphysics systems a significant challenge to the scientific community. While the integration of specific physical systems has been explored in the literature [24, 52, 33], there is still much that can be gained from a focused examination of the fundamental design that consolidates a multiphysics solver.

In this paper we describe an infrastructural framework for generalizing the integration of physical phenomena within $\mathcal{C}$**Heart**, a multiphysics software engine developed at King's College London. To date, it has been used to solve a variety of problems in physiological modeling, including the Navier–Stokes equations for fluid flow [17], nonlinear solid and porous mechanics [13, 32, 31, 3, 2], scalar transport equations [60, 41], elastic wave mechanics [49], pressure Poisson for approximating pressure from Navier–Stokes using known velocities [48, 20], and electrophysiology [84]. These essential single-physics building blocks provide the components for assembly and linking to form multiphysics models. This is achieved through a software infrastructure using what we call *object-sets* that act as generic sets to store different core FEM objects that may be constructed, used, or shared by any multiphysics problem. This strategy is used to manage bases, meshes and variables, topological interfaces, equation abstraction through equation mappings, and a series of general physics couplings to flexibly interlink heterogeneous physical models. These processes enable partitioned coupling, monolithic coupling, or a mixed approach for different components of the physical systems allowing for a multitude of possible solution strategies. The infrastructural organization outlined has enabled the interlinking of a variety of models, including fluid-solid coupling [67, 68, 70, 56, 18, 57, 58], vascular-porous flow [50, 51], and electromechanics [84].

The remainder of the paper will describe a general strategy for multiphysics integration and demonstrate its application. In section 2 we outline the physics systems which provide core simulation components for multiphysics integration. The general object-set approach, construction of function and variable spaces as well as the introduction of topological interfaces, is provided in section 3. A series of single-physics coupling strategies is described in section 4. The generic construction and handling of these different systems is introduced in section 5 with particular emphasis on their abstraction through the use of equation mappings. In section 6, we propose a graph parallelization strategy which enables efficient multicore processing. These techniques are then validated and demonstrated on a series of multiphysics systems relevant to cardiac physiology (see the online supplement (M101409_01.pdf [local/web 4.23MB])), with a specific example for fluid-structure interaction (FSI) provided in section 8.2.

2. Core single-physics module. In the finite-element framework the governing equations for physical systems are provided in standard weak form. In this section we introduce a generic notation used for defining operators, or functionals, for different physical systems. These functionals provide the building blocks for integrated multiphysics systems.

Different physical systems and their weak form functionals vary in the terms, function spaces, number of variables, and their inclusion of different fields. Here we consider each problem as a specialization of a general weak functional, $f : \boldsymbol{V}^h(\Omega_h) \times \boldsymbol{Y}^h(\Omega_h) \to \mathbb{R}$,

(2.1)
$$f(\mathbf{v}_h, \mathbf{y}_h; \Omega_h) = \int_{\Omega_h} a(\mathbf{v}_h) \cdot \mathbf{y}_h + c(\mathbf{v}_h) : \nabla \mathbf{y}_h \, d\Omega_h, \quad \mathbf{v}_h \in \boldsymbol{V}^h(\Omega_h), \quad \mathbf{y}_h \in \boldsymbol{Y}^h(\Omega_h),$$

TABLE 2.1
Summary of available options for different physics systems and physics-based operations.

Physics-based operators & operations	
ODE/DAE systems	Adaptive time stepping, implicit/explicit schemes (Euler, RK4, etc.) Lumped parameter cardiovascular models (incl. [77, 85]) Various cell models (incl. ten Tusscher 2006 [80], Luo–Rudy 1991 [53]) Dynamic loading of precompiled cell models
Solid mechanics	Incompressible, penalty, weak penalty, nearly incompressible formulations Arbitrary mixed formulations: displacement, velocity, pressure Arbitrary constitutive laws (isotropic, orthotropic, anisotropic laws) Summable constitutive laws, i.e., $\boldsymbol{\sigma}(\mathbf{x}) = \sum_k w_k(\mathbf{x})\boldsymbol{\sigma}_k(\mathbf{x})$
Scalar transport	Static/transient, inclusion/exclusion of advection and/or reaction Optional SUPG stabilization Solve as linear or nonlinear system
Fluid mechanics	Stokes and Navier–Stokes, static and transient Eulerian and arbitrary Lagrangian–Eulerian (ALE) formulations Conservative and nonconservative ALE formulations General mixed formulations SUPG formulation for all forms
1D fluid flow	Static/transient linear/nonlinear formulations Area-flow or area-velocity formulations Riemann invariant or compatibility-based boundary condition formulations Simplification to Poiseuille network flow (compliant and rigid vessel wall) Accommodates arbitrary wall constitutive models Primitive variable or Windkessel-type boundary conditions
Pressure Poisson	Stokes and Navier–Stokes, static and transient
Elastic wave equation	Linear and nonlinear formulations
Norms & measures	Compute L^2, H^1 norms/seminorms of variable expressions Compute general inner-product expressions
Projections	Compute L^2, H^1, and H_{div} projection operators
Parameter estimation	Reduced-order unscented Kalman filter Support for arbitrary spatially varying parameters

where $\mathbf{v}_h$ is an n-dimensional variable (which may consist of one or more state variable(s)), $\mathbf{y}_h$ is the nth-dimensional test function, $\boldsymbol{V}^h(\Omega_h)$ and $\boldsymbol{Y}^h(\Omega_h)$ are the set of trial and test spaces, respectively, and $\Omega_h \subset \mathbb{R}^d$ is the FEM domain description. Here, $a : \boldsymbol{V}^h \to \left[\mathbf{L}^2(\Omega_h)\right]^n$, $c : \boldsymbol{V}^h \to \left[\mathbf{L}^2(\Omega_h)\right]^{n\times d}$ are specific functionals of the variable, $\mathbf{v}_h$, which depend on the underlying physical system. In a single-physics example, the functional f would be used to find the solution $\mathbf{v}_h$ in the weak form statement: find $\mathbf{v}_h \in \boldsymbol{V}^h(\Omega_h)$ such that

$$f(\mathbf{v}_h, \mathbf{y}_h; \Omega_h) = 0 \quad \forall\ \mathbf{y}_h \in \boldsymbol{Y}^h(\Omega_h). \tag{2.2}$$

As the variability between physics arises predominantly in the a and c functionals, this abstract definition of the weak form problem enables generic handling of the core FEM procedures. In the remainder of this section we introduce the specific core physics currently implemented in ***CHeart***. A summary of these modules is also provided in Table 2.1.

2.1. Scalar transport. The conduction of electrical current, movement of metabolites, and conduction of simple flow potential problems are a few examples of a wide group of transport processes. Their simulation can be achieved through the solution of the transient scalar advection-diffusion-reaction equation or some simplification of it (see Table 2.1). As a result, scalar transport equations encompass a broad range of problems ranging from the most simplified form, Laplace, to the full transport system. In the case of a transient advection-diffusion-reaction equation

discretized using the backward Euler scheme, the operators in (2.1) have the form $a(v_h) := \frac{1}{\delta_t}(v_h^k - v_h^{k-1}) + \mathbf{u} \cdot \nabla v_h^k - \lambda v_h^k + s$, $c(v_h) := \mathbf{D}\nabla v_h^k$, where $v_h^k \in V^h = \boldsymbol{V}^h$ is the unknown scalar value at the kth time step, δ_t is the time step interval, $\mathbf{u}$ is an advective velocity field, λ is a linear reaction coefficient field, $\mathbf{D}$ is a tensor diffusion coefficient field, and s is a source/sink field.

2.2. Solid mechanics. Solid mechanics in cardiac applications is solved using large strain finite deformation theory [55, 9, 7] within quasi-static or transient frameworks. The specific formulations can vary depending on whether the material is modeled as compressible, incompressible, or nearly incompressible [32] as well as the specific selection of constitutive laws (see Table 2.1). As an example, a quasi-static incompressible material can be modeled by (2.1) using the a and c operators $a(\mathbf{u}_h, p_h) := (\rho\mathbf{b},\ \det|\nabla\mathbf{u}_h + \mathbf{I}| - 1)$, $c(\mathbf{u}_h, p_h) := (\boldsymbol{\sigma}(\mathbf{u}_h) - p_h\mathbf{I},\ 0)$, where $(\mathbf{u}_h, p_h) \in U^h \times W^h = \boldsymbol{V}^h$ are displacement and pressure, ρ and $\mathbf{b}$ denote material density and body force, respectively, $\boldsymbol{\sigma}(\mathbf{u}_h)$ is the deviatoric Cauchy stress tensor which may be defined as an arbitrary sum of core constitutive laws (see Table 2.1) scaled by a weighting field, and $\boldsymbol{I}$ is the identity tensor. Currently, a collection of isotropic (i.e., neo-Hookean, Mooney–Rivlin [9]), general anisotropic constitutive laws (i.e., Costa [14], Guccione [30], and Holzapfel–Ogden [36]) and viscoelastic models are implemented and can be selectively applied to the total or the deviatoric component of the strain tensor. Extensions to the existing constitutive laws are achieved by the generalized implementation of the stress tensor, enabling straightforward inclusion of new materials.

2.3. Fluid mechanics. Fluid mechanics can be solved for a variety of problems in the heart (see Table 2.1), ranging from steady Stokes flow in small coronary vessels to the more complex arbitrary Lagrangian–Eulerian (ALE) form of Navier–Stokes equations [35, 40, 37, 64, 25, 66, 70] in the cardiac heart chambers. As an example, in the nonconservative form of the Navier–Stokes equations discretized using backward Euler, the operators a and c of (2.1) are written as $a(\mathbf{v}_h, p_h) := (\frac{\rho}{\delta_t}(\mathbf{v}_h^k - \mathbf{v}_h^{k-1}) + \rho(\mathbf{v}_h^k - \mathbf{w}_h^k) \cdot \nabla\mathbf{v}_h^k,\ \nabla \cdot \mathbf{v}_h^k)$, $c(\mathbf{v}_h, p_h) := (\mu\boldsymbol{\sigma}(\mathbf{v}_h^k) - p_h^k\mathbf{I},\ 0)$, where $(\mathbf{v}_h^k, p_h^k) \in V^h \times W^h = \boldsymbol{V}^h$ are the fluid velocity/pressure at the kth time step with step-size δ_t; ρ, μ are the density and viscosity; and $\boldsymbol{\sigma}(\mathbf{v}_h)$ is the stress operator (see Table 2.1). By using various bases, $V^h \times W^h$ may be defined for a wide range of inf-sup stable spaces (see section 3.4), such as $\mathbb{P}^\kappa - \mathbb{P}^{\kappa-1}$ Taylor–Hood elements [11], $\mathbb{Q}^\kappa - \mathbb{P}^{\kappa-1}$ Nicolaides–Boland [8] elements, and Crouzeix–Raviart [16] spaces as well as others. Time step handling, in general, is consistent with that shown in [70] for ALE formulations. Further, as stabilization is often required for practical problems, the test and trial spaces may be different, enabling general introduction of streamline upwinding Petrov–Galerkin (SUPG) methods [38].

2.4. Darcy flow. Darcy's law in combination with conservation of mass is used to model the flow of a fluid through a permeable porous medium [15, 60, 13]. In the steady form, the operators a and c of (2.1) are written as $a(\mathbf{v}_h, p_h) := (\mu\mathbf{K}^{-1}\mathbf{v}_h,\ \nabla \cdot \mathbf{v}_h - s)$, $c(\mathbf{v}_h, p_h) := (-p_h,\ 0)$, where $(\mathbf{v}_h, p_h) \in V^h \times W^h = \boldsymbol{V}^h$ are the Darcy velocity and pore pressure, respectively, μ is the dynamic viscosity, $\mathbf{K}$ is the permeability tensor, and s is a volumetric flow rate source or sink. Currently, this implementation relies on inf-sup stable spaces V^h and W^h.

2.5. 1D fluid flow. The 1D fluid flow equations in an elastic vessel can be obtained from the incompressible Navier–Stokes equations written in cylindrical coordinates by assuming axisymmetric flow and by absorbing the dependence on the radial

coordinate into a lumped parameter [39, 26, 76]. This results in an equation set defined by $a\,(Q_h, A_h) := (\frac{Q_h^k - Q_h^{k-1}}{\delta_t} + \kappa \frac{Q_h^k}{A_h^k},\ \frac{A_h^k - A_h^{k-1}}{\delta_t})$, $c(Q_h, A_h) := -(\alpha \frac{(Q_h^k)^2}{A_h^k} + \int c_w^2 dA,\ Q_h^k)$, where $(Q_h^k, A_h^k) \in V^h \times W^h = \boldsymbol{V}^h$ is the fluid volumetric flow rate/vessel cross-sectional area at the kth time step with step-size δ_t, κ is a friction coefficient, c_w is the local pulse wave velocity, and α is the momentum correction factor which depends on the assumed fluid velocity profile. At junctions where branching occurs, the above equations are further augmented by a set of coupling conditions, i.e., mass conservation and the Bernoulli equation [76].

2.6. ODE and DAE systems. Various systems of ordinary differential equations (ODEs) and differential algebraic equations (DAEs) ranging from $0D$ Windkessel models [85, 78, 1] to tissue-level cell models [23, 53, 80] can be described using the equation $v' = g(v)$, where g is known as the rate function. The solution of such problems can be obtained using explicit, semi-implicit, or implicit time integration with constant time steps or adaptive time stepping. For example, the standard explicit Euler scheme can be written in the form of (2.1) using the a and c operators, $a(v_h) := \frac{1}{\delta_t^k}(v_h^{k+1} - v_h^k) - g(v_h^k)$, $c(v_h) := 0$, where $v_h^k \in V^h = \boldsymbol{V}^h$ is the state variable, and δ_t^k is the step-size at the kth time step. The domain of the problem is normally either a point or a series of points which effectively decouples the solution to individual nodes or modes (i.e., for a system with n variables, $v_h \in [\mathbf{L}^\infty(\Omega_h)]^n$).

In practice, directly incorporating a wide spectrum of ODE and DAE models into the code base requires frequent editing and recompilation. Flexibility is achieved in ***CHeart*** through dynamic loading of user-defined rate functions g. This allows, for example, a direct C-code export from online model repositories such as CellML[1] to be integrated without global recompilation.

3. FEM weak form construction. The core physics-based operators introduced in the previous section rely on a series of routines common to all FEM codes. The additional challenge of multiphysics comes with the efficient reuse, integration, and cross-communication of fields, domains, and weak form functionals. Critical to this is the formation of *object-sets*—representing a collection of core objects—that may be reused freely. An illustrated example of how different object-sets within ***CHeart*** interact together is shown in Figure 3.1.

In this section we review the core routines implemented to facilitate multiphysics integration. As *basis* interpolation schemes may be widely different, we formulate a basis object-set for which we developed a single strategy for computation and interpolation of different basis functions. These basis tools form the foundation for the generation of *topologies*—meshes with specified interpolation—which are integrated into a topology object-set forming flexible structures to house meshes of different order, size, dimension, etc. To enable cross-communication between fields and topologies, we introduce a general construct of *interfaces* that allows the formulation of functionals which relate variables defined on different topologies. With these tools, we construct *variable* object-sets, which may be used in the general functionals introduced above to assemble weak form statements.

3.1. Basis generation. The core physics modules presented in section 2 require a range of different interpolation functions that may rely on different reference elements, interpolation orders, and expansion types. This is handled by construction of a basis object-set $\mathcal{B}$ with each individual basis object, B, being defined

[1] www.cellml.org

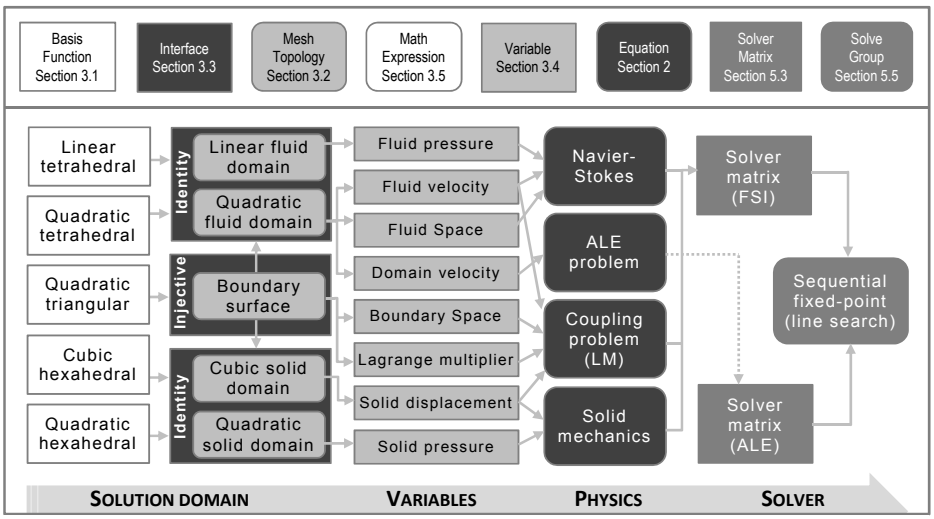


FIG. 3.1. *Illustration relating various object-sets used by* $\mathcal{C}$**Heart** *to solve physics problems. In this case, we consider the coupled fluid-solid interaction (see Results). The fluid is solved on a tetrahedral grid using* $\mathbb{P}^2 - \mathbb{P}^1$ *Taylor–Hood elements and the ALE Navier–Stokes formulation introduced in section* 2.3. *The solid is solved on a hexahedral curvilinear grid using* $\mathbb{Q}^3 - \mathbb{Q}^2$ *Taylor–Hood elements and the solid mechanics formulation from section* 2.2. *Object-set* $\mathcal{B}$ *is constructed using five basis object types with varying order shape and dimension. These are used with five topology objects comprising* $\mathcal{T}$*, and the interface* $\mathcal{I}$ *object-set is composed of two identity and two injective interfaces.* $\mathcal{V}$ *is then built from eight variables used in the four core problems feeding into the monolithic solver.*

TABLE 3.1

Summary of available element/basis specifications in $\mathcal{C}$**Heart**. *Options are available in any combination (excluding point element types which cannot support interpolation).*

Element/basis specifications	
Element types	point, line, quadrilateral, triangle, tetrahedral, hexahedral, prism
Basis types	nodal, continuous/discontinuous, order (0–5) modal, continuous/discontinuous, order (1–15) user specified‡
Quadrature schemes	Gauss–Legendre, integrable order (0–60) Keast–Lyness, integrable order (0–10)

‡: User defined basis enables usage of alternative basis definitions, including bubble and Crouzeix–Raviart element types, which are supplied using a simple text file containing basis coefficients and powers (i.e., $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$).

as $B = \{\tau, \{\varphi_1, \ldots, \varphi_K\}, Q\}$. Here τ denotes the master (or reference) element, $\{\varphi_1, \ldots, \varphi_K\}$ is a set of K-basis functions used to construct finite-dimensional function spaces, and $Q = \{(w_p, \boldsymbol{\xi}_p),\ p = 1, \ldots, N_p\}$ defines the quadrature rule via the list of weights $w_p \in \mathbb{R}^+$ and points $\boldsymbol{\xi}_p \in \tau$. The full list of available options is detailed in Table 3.1. All bases defined for a specific problem constitute the basis object-set $\mathcal{B}$.

A wide range of basis types, including modal [43], nodal [28, 88], Crouzeix–Raviart [16], and bubble functions [28], can be written using the cardinal expansion. If M is the total number of unique polynomial terms of B, $d = \dim(\tau)$, and K is the number of functions in the basis B, then φ_i $(i = 1, \ldots, K)$ and its derivatives can be computed

as

$$\varphi_i(\boldsymbol{\xi}) = \sum_{j=1}^{M} \boldsymbol{\alpha}(i,j) \prod_{k=1}^{d} \xi_k^{\boldsymbol{\beta}(k,j)}, \tag{3.1}$$

$$\frac{\partial \varphi_i(\boldsymbol{\xi})}{\partial \xi_1^{n_1} \cdots \partial \xi_d^{n_d}} = \sum_{j=1}^{M} \boldsymbol{\alpha}(i,j) \prod_{k=1}^{d} D(\beta_{kj}, n_k)\, \xi_k^{\boldsymbol{\beta}(k,j)-n_k},$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_d)^T$ is a local coordinate in τ, $\boldsymbol{\alpha} \in \mathbb{R}^{K \times M}$ is the sparse coefficient matrix, $\boldsymbol{\beta} \in \mathbb{R}^{d \times M}$ is the corresponding powers in the cardinal expansion, and $D(p,n) = \prod_{k=0}^{n-1}(p-k)$, $D(p,0) = 1$. As shown in (3.1), the derivatives can be automatically evaluated and refactored as a new sparse cardinal expansion with adjusted coefficients. Here we note that for nodal Lagrange and modal basis functions the number of terms in the cardinal expansion is equivalent to the number of basis functions (i.e., $K = M$). However, in the case of bubble or Crouzeix–Raviart elements, this is not the case.

The coefficient matrix $\boldsymbol{\alpha}$ may either be user-supplied or constructed algorithmically. For example, for nodal Lagrange polynomials, $\boldsymbol{\alpha} = \mathbf{A}^{-T}$ is the inverse transpose of the Vandermonde matrix (i.e., $\mathbf{A}(i,j) = \prod_{k=1}^{d} (\xi_k^i)^{\boldsymbol{\beta}(k,j)}$ with $\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^K \in \tau$ being nodal coordinates).

Furthermore, tensor-based expansions can be written in a similar form,

$$\varphi_i(\boldsymbol{\xi}) = \prod_{k=1}^{d} \sum_{j=1}^{K^{1/d}} \boldsymbol{\alpha}(\boldsymbol{\gamma}(i,k), j)\, \xi_k^{\boldsymbol{\beta}(j)},$$

$$\frac{\partial \varphi_i(\boldsymbol{\xi})}{\partial \xi_1^{n_1} \cdots \partial \xi_d^{n_d}} = \prod_{k=1}^{d} \sum_{j=1}^{K^{1/d}} \boldsymbol{\alpha}(\boldsymbol{\gamma}(i,k), j) D(\boldsymbol{\beta}(j), n_k)\, \xi_k^{\boldsymbol{\beta}(j)-n_k},$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{K \times K^{1/d}}$ and $\boldsymbol{\beta} \in \mathbb{R}^{K^{1/d}}$ are the 1D sparse expansions, and $\boldsymbol{\gamma} \in \mathbb{R}^{K \times d}$ is an index array defining the corresponding 1D basis. For efficient computation the expansion for each basis function is written as a sum of the nonzero components of $\boldsymbol{\alpha}$.

Due to the arbitrary degree of the basis, a range of quadrature schemes is required. Schemes specifically for triangular and tetrahedral elements, such as those by Lyness and Jespersen [54] and Keast [46], were integrated based on the source code provided by [63]. Higher order Gauss–Legendre schemes are also available based on finding the roots of the Jacobi polynomials [43], and using tensor product expansion/collapsed expansions to integrate up to 60th order. In addition to these built-in bases, ***C*Heart** supports arbitrary user-defined bases and/or quadrature schemes to be provided via a simple text file.

3.2. Topologies. Similar to the outlined basis object-set $\mathcal{B}$, an object-set of topologies $\mathcal{T}$ is constructed. A topology object $T \in \mathcal{T}$ in ***C*Heart** is constructed using a specific basis $B \in \mathcal{B}$ and provides the mapping between mesh element indices and global basis function (node or mode) indices used to evaluate fields over elements. Specifically, the topology $T = \{\boldsymbol{E}, B, N_E, N_N\}$ is comprised of N_E and N_N, denoting the total number of elements and global basis functions, respectively, B being the basis used to perform field evaluations, and $\boldsymbol{E} \in \mathbb{N}^{N_E \times K}$ the element mapping such that $N = \boldsymbol{E}(e,k)$ for some $e \in \{1, \ldots, N_E\}$, $k \in \{1, \ldots, K\}$, $N \in \{1, \ldots, N_N\}$.

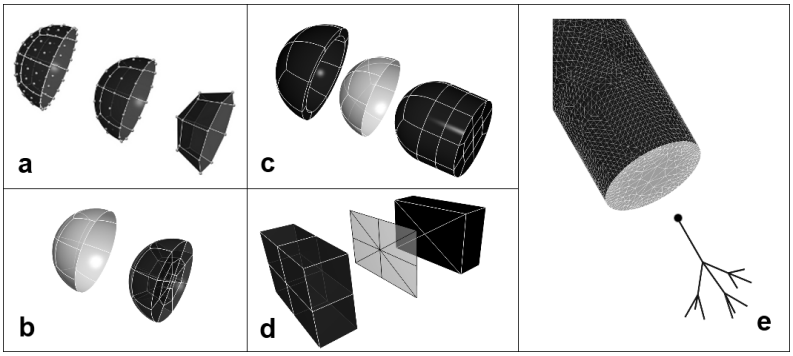


FIG. 3.2. *Types of interfaces supported in* $\mathcal{C}$**Heart**. (a) *Identity interfaces between linear, quadratic, and cubic quadrilateral meshes.* (b) *Injective interface.* (c) *Injective interface via an intermediary.* (d) *An injective interface between nonconforming meshes through a conforming intermediary.* (e) *A degenerate interface between a* 1*D vessel terminal and the boundary face of the cylinder.*

3.3. Interfaces. By default, each topology $T \in \mathcal{T}$ is defined as a singular entity with an unknown relation to others in the object-set. However, in many cases, such as mixed formulations or multiphysics interactions, it is desirable for a well-defined relationship to exist between topologies, enabling the transfer of data from one to the other. This may be accomplished by defining an interface object $I_{i\to j} \in \mathcal{I}$ (with $\mathcal{I}$ being the interface object-set) which effectively maps element indices and local element coordinates from T_i to T_j, $T_i, T_j \in \mathcal{T}$. Each interface object $I_{i\to j} = \{T_i, T_j, \boldsymbol{e}^{i,j}, \boldsymbol{I}^{i,j}\}$ contains components to construct a mapping. Letting $N_{E,i}$ and $N_{E,j}$ be the numbers of elements, and τ_i and τ_j the master elements in T_i and T_j, respectively, then $\boldsymbol{e}^{i,j} \in \mathbb{N}^{N_{E,i}}$ defines the relationship between element indices in the two topologies, and $\boldsymbol{I}^{i,j} \in \mathbb{R}^{N_{E,i}\times K}$ the relationship between local coordinates. The mapping can be fully described by the mapping $\boldsymbol{\zeta}_{i\to j} : \{1,\dots,N_{E,i}\} \times \tau_i \to \{1,\dots,N_{E,j}\} \times \tau_j$,

$$\boldsymbol{\zeta}_{i\to j}(e,\boldsymbol{\xi}) = \left(\boldsymbol{e}^{i,j}(e), \sum_{m=1}^{K} \boldsymbol{I}^{i,j}(e,m)\varphi_{i,m}(\boldsymbol{\xi})\right) = (e',\boldsymbol{\xi}'), \tag{3.2}$$

for some $(e',\boldsymbol{\xi}') \in \{1,\dots,N_{E,j}\} \times \tau_j$ and $(e,\boldsymbol{\xi}) \in \{1,\dots,N_{E,i}\} \times \tau_i$. The mapping $\boldsymbol{\zeta}_{i\to j}$ enables interpolation and evaluation of fields and field derivatives as will be discussed in section 3.4.

In $\mathcal{C}$**Heart** all interfaces are assumed to be null mappings by default, i.e., no mapping exists between topologies. Various kinds of interfaces may, subsequently, be defined. These are outlined below and illustrated in Figure 3.2.

Identity interface. The most commonly used interface in FEMs is the identity interface, given in Definition 3.1 and shown in Figure 3.2a, which simply returns the same element and local element coordinates. Here, the associated basis of each topology must be compatible in that their respective master elements must be geometrically equivalent and their number of elements identical.

DEFINITION 3.1 (identity interface). *Let* $T_i, T_j \in \mathcal{T}$ *with* $N_{E,i} = N_{E,j} = N_E$ *and* $\tau_i = \tau_j = \tau$. *An interface* $I_{i\to j} \in \mathcal{I}$ *is called an identity interface if for any* $(e,\boldsymbol{\xi}) \in \{1,\dots,N_E\} \times \tau$ *the mapping function* $\boldsymbol{\zeta}_{i\to j}(e,\boldsymbol{\xi}) = (e,\boldsymbol{\xi})$, *where* $\zeta_{i\to j}$ *is defined by* (3.2).

This is often employed for classic mixed-element formulations, such as those used in fluid mechanics or solid mechanics formulations, where velocity/displacement share the same mesh as pressure variables but use different interpolations (i.e., different topologies).

Topology groups. As topologies linked by an identity interface share many commonalities, it is often useful to define topology groups (see Definition 3.2), which link these objects together. This is particularly convenient, as more complex interfaces may then be defined for one member of the topology group and used for all other group topologies.

DEFINITION 3.2 (topology group). *Let $\mathcal{T}_g \subseteq \mathcal{T}$; then $\mathcal{T}_g$ is a topology group if there exists an identity interface $I_{i\to j} \in \mathcal{I}$ for every T_i,$T_j \in \mathcal{T}_g$. Further, $\mathcal{T}_g(i)$ is referred to as the topology group of T_i if $T_i \in \mathcal{T}_g(i)$.*

Injective interface. In multiphysics simulations the interconnectivity of topologies is often more complex. For this injective interface, mappings may be introduced which provide a continuous map between element/local coordinate data and topologies. This may be on a portion of a domain, as in FSI where exchange between the fluid and solid occurs on an interface. Definition 3.3 specifies the details of injective interfaces which are also illustrated graphically in Figure 3.2b–d.

DEFINITION 3.3 (injective interface). *Let $T_i, T_j \in \mathcal{T}$. An interface $I_{i\to j} \in \mathcal{I}$ is called injective if for every $(e, \boldsymbol{\xi}) \in \{1, \dots, N_{E,i}\} \times \tau_i$ there exists a unique $(e', \boldsymbol{\xi}') \in \{1, \dots, N_{E,j}\} \times \tau_j$ such that $\boldsymbol{\zeta}_{i\to j}(e, \boldsymbol{\xi}) = (e', \boldsymbol{\xi}')$, where $\zeta_{i\to j}$ is defined by* (3.2).

This type of interface provides a mapping for all element/local coordinate pairings from T_i into some or all pairings from T_j. As a result, the injective interface allows for the embedding of refined grids or lesser dimensional manifolds into another topology. This formulation enables a wide range of associations between topologies. Injective maps can be used to couple subsets of two topologies where only a portion of each is interfaced by introducing an intermediary topology which is injective to both over the desired subsets (see Figure 3.2c). This need arises in fluid-solid coupling, for example, where the interaction between fluid and solid domains occurs on some portion of the domain boundaries (see section 8.2). Further, injective interfaces can be easily extended to fully nonconforming grids by introducing an additional polygonal topology nested within the intersections of the interfacing topologies (see Figure 3.2d).

An important property of injective interfaces is their transferability to the associated topology groups, which facilitates data transfer between single-physics problems employing mixed interpolation spaces. As shown in Lemma 3.4, it is clear that an injective interface $I_{i\to j} = \{T_i, T_j, \boldsymbol{e}^{i,j}, \boldsymbol{I}^{i,j}\}$ can be extended to any $T_k \in \mathcal{T}_g(T_i)$ using the mapping $\boldsymbol{\zeta}_{i\to j}$.

LEMMA 3.4. *Suppose $T_i, T_k \in \mathcal{T}_g(i)$ and $I_{i\to j} \in \mathcal{I}$ is injective. Then, there is an implicit mapping $I_{k\to j} \in \mathcal{I}$ that is injective.*

Proof. Since $T_i, T_k \in \mathcal{T}_g(i)$, then $\boldsymbol{\zeta}_{k\to i}(e, \boldsymbol{\xi}) = (e, \boldsymbol{\xi})$ and $I_{k\to i}$ is an identity interface. The mapping for $I_{k\to j}$ can therefore be constructed as $\boldsymbol{\zeta}_{k\to j}(e, \boldsymbol{\xi}) = \boldsymbol{\zeta}_{i\to j} \circ \boldsymbol{\zeta}_{k\to i}(e, \boldsymbol{\xi}) = \boldsymbol{\zeta}_{i\to j}(e, \boldsymbol{\xi})$ for all $(e, \boldsymbol{\xi}) \in \{1, \dots, N_{E_k}\} \times \tau_k$. Since the interface $I_{i\to j}$ is an injective interface, $I_{k\to j}$ is also an injective interface. ∎

An important subset of identity and injective interfaces includes those which are complete (see Definition 3.5). By definition, all identity interfaces are complete. However, in general, an injective interface enables linking one topology T_i to a subset of another T_j. A complete interface ensures that this subset extends to the entirety

of T_j. This is particularly important for FEM weak form statements which must be held over a defined domain which may require interfaces to other variables.

DEFINITION 3.5 (complete interface). *An injective interface $I_{i\to j} \in \mathcal{I}$ is called complete if $\boldsymbol{\zeta}_{i\to j}$ is bijective.*

If an injective interface $I_{i\to j} \in \mathcal{I}$ is defined between two topologies $T_i, T_j \in \mathcal{T}$, then an inverse mapping can be implicitly evaluated and constructed (see Definition 3.6). However, explicitly constructing $I_{j\to i}$ in the form shown in (3.2) is not, in general, straightforward. Instead, the inverse of an injective interface is computed procedurally by $\mathcal{C}$**Heart** based on the provided injective interface.

DEFINITION 3.6 (inverse interface). *For an injective interface $I_{i\to j} \in \mathcal{I}$ we may implicitly define an inverse interface $I_{j\to i}$ that for some $(e', \boldsymbol{\xi}') \in \{1, \dots, N_{E,j}\} \times \tau_j$ there exists a unique $(e, \boldsymbol{\xi}) \in \{1, \dots, N_{E,i}\} \times \tau_i$ such that $\boldsymbol{\zeta}_{i\to j}(e, \boldsymbol{\xi}) - (e', \boldsymbol{\xi}') = (0, \mathbf{0})$.*

Degenerate interface. In certain cases, the requirements of injective interfaces or their inverses can be too restrictive, requiring the construction of *degenerate interfaces* (see Definition 3.7), where a topology may be mapped nonuniquely. This case arises when reduced-dimensional formulations are coupled to full-dimensional problems, such as 1D blood vessel flow coupled to three-dimensional (3D) fluid flow (see Figure 3.2e). Clearly, the reverse of this mapping is ill-posed and can only be obtained in simplified instances.

DEFINITION 3.7 (degenerate interface). *An interface $I_{i\to j} = \{T_i, T_j, \boldsymbol{e}^{i,j}, \boldsymbol{I}^{i,j}\}$ is called degenerate if for some $(e_1, \boldsymbol{\xi}_1), (e_2, \boldsymbol{\xi}_2) \in \{1, \dots, N_{E,i}\} \times \tau_i$ there exists a single pair $(e', \boldsymbol{\xi}') \in \{1, \dots, N_{E,j}\} \times \tau_j$ such that $\boldsymbol{\zeta}_{i\to j}(e_1, \boldsymbol{\xi}_2) = (e', \boldsymbol{\xi}')$, $\boldsymbol{\zeta}_{i\to j}(e_2, \boldsymbol{\xi}_2) = (e', \boldsymbol{\xi}')$.*

An object-set of well-defined interfaces plays a crucial role in reusability of a single-physics code within coupled problems, as it enables the abstraction of interpolation-specific implementations as well as cross-talk between different computational domains.

3.4. Variables and function spaces. Using the basis, topology, and interface structures outlined, it is possible to formally construct interpolated variables as well as discrete function spaces. As with previous structures, an object-set of variables $\mathcal{V}$ may be defined, enabling the construction or calculation of numerous quantities that are either required for solving physical systems or desired as outputs. Each variable object $V \in \mathcal{V}$ is defined as $V = \{T_V, \boldsymbol{C}_V\}$, where $T_V = \{\boldsymbol{E}_V, B_V, N_E, N_N\} \in \mathcal{T}$ is the topology, and $\boldsymbol{C}_V \in \mathbb{R}^{n\times N_N}$ the coefficient matrix, so that a finite-element variable can be computed as

$$\hat{\mathbf{v}}_h(e, \boldsymbol{\xi}) = \sum_{l=1}^{n} \sum_{k=1}^{K} \boldsymbol{C}_V(l, \boldsymbol{E}_V(e, k)) \varphi_k(\boldsymbol{\xi}) \boldsymbol{e}_l, \quad (e, \boldsymbol{\xi}) \in \{1, \dots, N_E\} \times \tau_V, \tag{3.3}$$

where $\{\varphi_1, \dots \varphi_K\} \in B_V$, $\boldsymbol{e}_1, \dots, \boldsymbol{e}_n \in \mathbb{R}^n$ are the unit base vectors for $\mathbb{R}^n$. Here we add a *hat* (ˆ), distinguishing the variable from those introduced in section 2, as it is principally a function of the element index and master element coordinate. All fields in $\mathcal{C}$**Heart**, including the mesh coordinates, are specified in this way, enabling the use of any topology defined for these fields. In the case of spatial domains, this flexibility allows for the construction of standard simplex and hexahedral domains as well as the use of curvilinear grids. We assume each domain variable $D = \{\boldsymbol{C}_D, T_D\} \in \mathcal{V}$ is well constructed (i.e., continuous, nonoverlapping, locally invertible), enabling the

construction of a computational domain,

$$\Omega_h := \left\{ \mathbf{x} \in \mathbb{R}^d \,\middle|\, \mathbf{x} = \hat{\mathbf{x}}(e, \boldsymbol{\xi}) \quad \forall (e, \boldsymbol{\xi}) \in \{1, \dots, N_E\} \times \tau_D \right\}, \tag{3.4}$$

where the definition of $\hat{\mathbf{x}}$ follows from (3.3) using the appropriate domain topology T_D (and its components) as well as the scaling array $\boldsymbol{C}_D$.

However, unlike standard variables, domains have additional properties enabling them to represent different spatial reference frames, for example, Eulerian/Lagrangian [55] or ALE [21]. This is accomplished by allowing the domain Ω_h to be considered as a function of time $\Omega_h(t)$ where the spatial positions of $\mathbf{x} \in \Omega_h$ are summed with an additional variable $V = \{T_V, \boldsymbol{C}_V\} \in \mathcal{V}$ to give the current $\mathbf{x}(t) \in \Omega_h(t)$, i.e., $\mathbf{x}(t) = \hat{\mathbf{x}}(e, \boldsymbol{\xi}, t) = \sum_{l=1}^{d} (\sum_{k=1}^{K_D} \boldsymbol{C}_D(l, \boldsymbol{E}_D(e,k)) \varphi_{D,k}(\boldsymbol{\xi}) + \sum_{k=1}^{K_V} \boldsymbol{C}_V(l, \boldsymbol{E}_V(e,k))\big|_t \varphi_{V,k}(\boldsymbol{\xi})) \boldsymbol{e}_l$, where $\{\varphi_{m,1}, \dots, \varphi_{m,K_m}\} \in B_m$ for $m = \{D, V\}$. We note that $I_{D \to V}$ is assumed to be an identity interface to enable summing.

Variable-space pairing. While (3.3) outlines the nth-dimensional vector function $\hat{\mathbf{v}}_h$ as a function of local element index and local coordinates, it is desirable in FEMs to instead define $\mathbf{v}_h : \Omega_h \to \mathbb{R}^n$ as a function of the spatial domain Ω_h. Whether this definition is possible depends on if the variable-space pairing $(\hat{\mathbf{v}}_h, \Omega_h)$ is well defined.

DEFINITION 3.8. *A variable-space pairing $(\hat{\mathbf{v}}_h, \Omega_h)$ (defined in (3.3) and (3.4)) is well defined if the spatial domain Ω_h is well constructed (i.e., continuous, non-overlapping, locally invertible), and there exists a topology $T_r \in \mathcal{T}$ and interfaces $I_{r \to D}, I_{r \to V} \in \mathcal{I}$, where $I_{r \to D}$ is complete and $I_{r \to V}$ is an identity or injective interface. In this case, T_r is called a root topology.*

The stipulations of Definition 3.8 ensure that for each $\mathbf{x} \in \Omega_h$ there exists a unique corresponding pair $(e, \boldsymbol{\xi}) \in \{1, \dots, N_{E_r}\} \times \tau_r$. The existence of a root topology T_r as outlined implies that the composition

$$\mathbf{v}_h(\mathbf{x}) = \hat{\boldsymbol{v}}_h \circ \boldsymbol{\zeta}_{r \to V}(e, \boldsymbol{\xi}), \quad \mathbf{x} = \hat{\mathbf{x}} \circ \boldsymbol{\zeta}_{r \to D}(e, \boldsymbol{\xi}) \tag{3.5}$$

is well defined for all $\mathbf{x} \in \Omega_h$. As a result, T_r provides a topology for which $\mathbf{v}_h$ and Ω_h can be related. The completeness requirement for $I_{r \to \Omega}$ ensures that each $\mathbf{x}$ has a corresponding $\mathbf{v}_h$; however, $I_{r \to V}$ need not be complete, allowing for conditions or weak equations on subsets of a variable field (i.e., use of constraints along boundaries or interior regions).

Discrete function spaces. Using the definition of discrete variables as shown in (3.3), and the concept of variable-space pairings introduced in Definition 3.8 and (3.5), it is natural to define the discrete function spaces introduced in (2.1) as

$$V^h(\Omega_h) = \Big\{ \mathbf{v}_h \in [L^2(\Omega_h)]^n \,\Big|\, \mathbf{v}_h(\mathbf{x}) = \hat{\boldsymbol{v}}_h \circ \boldsymbol{\zeta}_{r \to V}(e, \boldsymbol{\xi}),\ \mathbf{x} = \hat{\mathbf{x}} \circ \boldsymbol{\zeta}_{r \to D}(e, \boldsymbol{\xi}) \\ \forall (e, \boldsymbol{\xi}) \in \{1, \dots, N_{E_r}\} \times \tau_r,\ \forall \boldsymbol{C}_V \in \mathbb{R}^{n \times N_N} \Big\}.$$

The properties of the discrete space $V^h(\Omega_h)$, such as the smoothness and order, depend on the underlying connectivity of T_V given by $\boldsymbol{E}_V$, the choice of basis, as well as the spatial domain over which the space is defined.

3.5. Expressions. Often during a modeling process, it is desirable to conduct algebraic computations with the variables defined over the finite-element meshes at

runtime. This functionality is implemented in $\mathcal{C}$**Heart** through an object-set of plaintext mathematical expressions provided at problem specification time. The expressions can take variable pointers, time, and other expressions as inputs and produce outputs that can be assigned to set boundary conditions, material and source fields, and other variables as part of partitioned solver strategies and postprocessing. The parsing of expressions is implemented using an internal bytecode-compiler.[2] Based on an algebraic expression provided by the user, the internal compiler decomposes the string into its component operators which are then stored as a sequence of simple bytecode instructions. This conversion occurs only once, when the problem is set up, and so in this way the expressions can be evaluated at key points in the computation phase of execution with a minimal amount of runtime overhead. The user can specify both scalar and vector expressions that contain standard algebraic and trigonometric functions as well as logical and conditional operators, modulo max/min and data interpolation.

3.6. General weak form systems. Multiphysics systems are comprised of multiple weak form operators which act on different variables. These operators, outlined for specific physics in section 2, are constructed as needed and built into an object-set $\mathcal{F}$, where each operator $f \in \mathcal{F}$ is defined as in (2.1).

Remark 3.9. In the case of Lagrangian or arbitrary Lagrangian–Eulerian (ALE) domains, functionals can be defined as $f(\mathbf{v}_h, \mathbf{y}_h; \Omega_h(t)) = \int_{\Omega_h(t)} a(\mathbf{v}_h) \cdot \mathbf{y}_h + c(\mathbf{v}_h) : \nabla \mathbf{y}_h \; d\Omega_h$, enabling definition of the weak form on a specific reference frame. In the case of conservative ALE formulations [25, 70],

$$f(\mathbf{v}_h, \mathbf{y}_h; \Omega_h) = \int_{\Omega_h(t+\delta_t)} a_0(\mathbf{v}_h) \cdot \mathbf{y}_h \; d\Omega_h \; - \; \int_{\Omega_h(t)} a_1(\mathbf{v}_h) \cdot \mathbf{y}_h \; d\Omega_h + \int_t^{t+\delta_t} \int_{\Omega_h(s)} a_2(\mathbf{v}_h) \cdot \mathbf{y}_h + c(\mathbf{v}_h) : \nabla \mathbf{y}_h \; d\Omega_h \; ds$$

is used with functionals a_0, a_1, and a_2.

Remark 3.10. Boundary conditions may be introduced to operators $f \in \mathcal{F}$ as discussed in section 4.2.

Each operator $f \in \mathcal{F}$ may be combined with others to form multiphysics systems. This is done by defining the summed operator $\boldsymbol{f} \subseteq \mathcal{F}$,

$$\boldsymbol{f}(\mathbf{v}_h, \mathbf{y}_h) := \sum_k f_k(\mathbf{v}_{h,k}, \mathbf{y}_{h,k}; \Omega_{h,k}), \quad f_k \in \mathcal{F}, \tag{3.6}$$

where f_k and $\Omega_{h,k}$ are the kth operator/domain and $(\mathbf{v}_{h,k}, \mathbf{y}_{h,k})$ its respective variables (where $\mathbf{v}_{h,k} \in \mathbf{v}_h$ and $\mathbf{y}_{h,k} \in \mathbf{y}_h$). Importantly, the operator $\boldsymbol{f}$ and its variables, $\mathbf{v}_h, \mathbf{y}_h$, do not reside on a single domain. This enables the merger of operators which span different $\Omega_{h,k}$, as often occurs in multiphysics systems. Each operator $\boldsymbol{f}$ denotes a specific system for which we look to find $\mathbf{v}_h \in \boldsymbol{V}^h$ so that

$$\boldsymbol{f}(\mathbf{v}_h, \mathbf{y}_h) = 0 \quad \forall \, \mathbf{y}_h \in \boldsymbol{Y}^h, \tag{3.7}$$

where $\boldsymbol{V}^h, \boldsymbol{Y}^h$ denote the complete trial/test spaces.

[2] http://fparser.sourceforge.net/

4. Physics coupling. In sections 2 and 3 we introduced the formulation of single-physics operators as well as their construction through a series of object-sets. To further enable flexible multiphysics integration, this section discusses a series of coupling strategies developed to formally link operators (as shown in (3.7)) which represent multiphysics weak form systems. Here we define a coupling as a functional relationship between single-physics systems within a problem, the examples of which can be found in Table 4.1. While this may occur in numerous instances, here we identify three principal mechanisms:

(1) functional dependence of existing operators on evolving variables from another problem,
(2) functional dependence of boundary conditions on evolving variables from another problem, and
(3) new coupling operators which effectively link problems together.

These approaches are all made possible using different couplings methods.

4.1. Coupling via existing operators. As full access to all variables in $\mathcal{V}$ is available to any problem, the inclusion of variables through existing operators is straightforward. For example, the diffusion coefficient in a scalar transport problem may be linked to any variable in $\mathcal{V}$, including one which is solved for within a different system (e.g., as is the case in strongly coupled electromechanics). This enables the automatic coupling of systems. Further, as different solve strategies may be applied as discussed in section 5.5, ***CHeart*** enables this coupling to be either explicit or handled through a fixed point iteration.

4.2. Coupling via boundary conditions. Each operator (see section 3.6) may be linked to any number of boundary conditions. Steady or time-dependent Dirichlet or Neumann boundary conditions may be applied to different portions of the domain boundary (typically a series of user-defined nodes or element faces).

CHeart boundary conditions may be specified using variables as well as numeric expressions, so that the solution of one problem may be provided as the boundary data for another problem. This is often required in partitioned ALE Navier–Stokes where the domain motion can be computed independently from the flow while the two problems are coupled along portions of the domain boundary. This type of coupling is also common in FSI, where the motion of the interface between fluid and solid domains is set as a Dirichlet condition and the traction on fluid is applied as a Neumann-type condition to the solid.

4.3. Coupling via expressions. While the integration of multiple physics using variables provides a powerful tool for coupling systems, this is not always sufficient as the coupling may require algebraic or differential manipulation of a variable. This feature is enabled in ***CHeart*** via expression strings (see section 3.4) which may be provided by users to define the specific mathematical form of coupling.

In many cases, such as with boundary conditions, expressions and variables may be used interchangeably. In addition, ***CHeart*** allows the casting of expressions to variables, which enables the coupling of certain operations which would otherwise need to be hardcoded.

4.4. Monolithic integration via coupling operators or Lagrange multipliers. Multiphysics integration often necessitates the inclusion of additional terms or constraints in the system. In some cases, as in partitioned approaches, this merely involves adding a source or augmenting boundary conditions. In other approaches, the coupling may be imposed via *coupling operators*, or by the introduction of *La-*

TABLE 4.1
Summary of available coupling options for different styles of multiphysics.

Physics coupling operators/options	
Simple coupling	Coupling via shared variables
	Coupling via algebraic expressions
	Coupling via boundary conditions
Lagrange coupling	Coupling one or more variables to a constraint
	Coupling variable or variable rate, i.e., $D\mathbf{u} = \mathbf{u}$ or $D\mathbf{u} = \partial\mathbf{u}/\partial t$
	Coupling all components or normal components, i.e., $L(\mathbf{u}) = \mathbf{u}$ or $L(\mathbf{u}) = \mathbf{u} \cdot \mathbf{n}$
	User defined constraint expressions available
	Support for constraints on 1D, 2D, and 3D manifolds or volumes
Volume coupling	Linear coupling through reaction terms
	Support for 2D, 3D

grange multipliers. In these cases, the additional elements are considered as separate physical problems which augment traditional single-physics examples. The modular implementation of these *coupling problems* in $\mathcal{C}$**Heart** promotes code reuse in various types of multiphysics problems.

Coupling operators. In $\mathcal{C}$**Heart** coupling operators provide a link between variables from distinct physics problems through the addition of new terms or operators into the weak form. Considering a set of variables $(\mathbf{u}_1, \ldots, \mathbf{u}_m) \in U_1^h \times \cdots \times U_m^h$ which have been introduced for a set of physics problems, the general form of a in equation (2.1) for coupling operators is given by $a(\mathbf{u}_1, \ldots, \mathbf{u}_m) := (\sum_{k=1}^m \gamma_{1k} L_{1k} \mathbf{u}_k, \ldots, \sum_{k=1}^m \gamma_{mk} L_{mk} \mathbf{u}_k)$, where γ_{ik} is a scalar coefficient detailing the coupling between $\mathbf{u}_i$ and $\mathbf{u}_k$ and L_{ik} the corresponding linear operator (see Table 4.1). As a result, the weak form defined for $\mathbf{u}_i$ is augmented with $\sum_{k=1}^m \gamma_{ik} L_{ik} \mathbf{u}_k$, which may be tailored to address a specific type of coupling. One classical example of the use of coupling operators is in Boussinesq convection, where the Navier–Stokes equations are coupled to advection-diffusion through the advective velocity field and a temperature-based forcing term.

Lagrange multipliers. Lagrange multipliers are typically used for the addition of new constraint equations which may augment a single-physics system or link multiple systems. Specifically, an additional variable $\boldsymbol{\lambda} \in M^h$ is added to the set of variables $(\mathbf{u}_1, \ldots, \mathbf{u}_m)$ in order to weakly impose over some domain or manifold, Ω_h, that the variables satisfy $\sum_k w_k L_k(D_k \mathbf{u}_k) - \boldsymbol{s} = \mathbf{0}$, where L_k and D_k are the kth linear algebraic and differential operators, respectively (see Table 4.1), w_k is the kth scalar weight, and $\boldsymbol{s}$ is some provided source. The additional constraint terms can be included using the operator a of (2.1) defined as $a(\mathbf{u}_1, \ldots, \mathbf{u}_m, \boldsymbol{\lambda}) := (w_1 L_1(\boldsymbol{\lambda}), \ldots, w_m L_m(\boldsymbol{\lambda}), \sum_k w_k L_k(D_k \mathbf{u}_k) - \boldsymbol{s})$, where $(\mathbf{u}_1, \ldots, \mathbf{u}_m, \boldsymbol{\lambda}) \in U_1^h \times \cdots \times U_m^h \times M^h$. This enables a broad range of coupling using multipliers and may also be used to construct projection operators for iterative fixed point schemes. Examples of this approach are found in the imposition of complex physical constraints as well as fluid-solid coupling.

5. Assembly, matrix operations, and building. Efficient and flexible matrix assembly operations are a principal enabler of multiphysics integration in finite-element modeling and a key engine driving its computational performance. The core $\mathcal{C}$**Heart** library provides automated functionalities and standardized interfaces to enable rapid and extensible developments. The process by which the topology group, equation map, and variable are used to build and populate sparse matrices is described below.

5.1. FEM element assembly. FEM element assembly of residuals, matrices, and other objects (see section 3) is usually handled by an element loop over a mesh. In the code, ***C*Heart** contains a single element loop routine, using a series of procedure pointers—`Setup`, `LocalElementCalculate`, `LocalElementAssemble`, and `CleanUp` procedures—to control the tasks executed. These procedure pointers may be linked (or nullified) before execution of the element loop and altered according to the desired assembly task (i.e., residual evaluation, Jacobian construction, norm calculations, etc.). `Setup` and `CleanUp` procedures are commonly used to extract tasks which need not be repeated during the element loop—such as dynamically allocating/deallocating memory or constructing/destroying access shortcuts to data required for FEM assembly—to optimize computational performance. The `Calculation` and `Set` routines are established to do tasks which may vary by element, with the `Calculation` step involving the computation of a physics-dependent element-level matrix or residual, for example. The `Set` step is a low-level, abstracted task used to incorporate element-level data into global matrix structures.

Beyond the standard steps followed in FEM assembly, the presence of merged functionals and interface mappings necessitates important extensions. As the weak form functional for a physical system may be a sum of multiple physical systems (see (3.6)), the general assembly occurs over each individual functional $f_k \in \boldsymbol{f}$, for which we define a symbolic variable dependence using equation maps, $\mathbf{E}(f_k)$. Crucially, this equation map contains a root topology T_r, over which element-level procedures may be computed.

Since integral evaluations of weak form equations (see (2.1)) are computed over T_r, all other topologies used as part of the problem governed by f_k must be evaluated on T_r. This list may be automatically compiled by recording the usage of various topologies as problems are defined by the user. The completeness of T_r ensures the looping process covers the indicated spatial domain.

In addition, the corresponding basis functions and their gradients of all involved terms must be evaluated on T_r. If $I_{r\to k}$ for $T_k \in \mathcal{T}$ is an identity interface (see Definition 3.1), no mapping is required, and pre-evaluations of the basis functions (and their derivatives) can be used directly. However, when $I_{r\to k}$ is a more complex mapping, such as an injective interface, then the basis on T_k must be mapped onto T_r (see (3.6)). Users can elect to either execute these computations *on the fly* or have the program precompute these quantities into fast access cached lists.

Algorithm (FEM LOOP)

```
for all f_k ∈ f
    call SetupProcedure
    for all e ∈ T_r ⇐ E(f_k)
        for all T_i ∈ {T_k}
            Compute needed φ, ∇_r φ, ... at Q_r for all φ ∈ B_i
        end for
        call LocalElementCalculateProcedure
        call LocalElementAssembleProcedure
    end for
    call CleanUpProcedure
end for
```

5.2. Equation mappings. The translation of weak form operators f into discrete finite-element equivalents may be achieved via the use of equation maps (Definition 5.1). An equation map is an abstraction of the dependence of weak form operators

on their spaces, which completely encapsulates the row-column variable mappings necessary to construct the local element tensors. Considering (2.2), the equation mapping defines whether $f(\mathbf{v}_h', \mathbf{y}_h'; \Omega_h) \neq 0$ for some $\mathbf{v}_h' = (\mathbf{0}, \dots, \mathbf{v}_{h,i}, \dots, \mathbf{0})$ and $\mathbf{y}_h' = (\mathbf{0}, \dots, \mathbf{y}_{h,j}, \dots, \mathbf{0})$ with $\mathbf{v}_{h,i} \in V_i^h$ and $\mathbf{y}_{h,j} \in Y_j^h$, that is, whether there are operators in f which pair the ith and jth trial and test spaces.

DEFINITION 5.1 (equation map). *Let $\{V_1^h, \dots, V_m^h\}$ and $\{Y_1^h, \dots, Y_n^h\}$ be well-defined function spaces on Ω_h. Then, if f is a functional form as shown in* (2.2) *on the discrete finite-element spaces, the equation map $\mathbf{E}(f)$ of f is defined for each row variable $i \in \{1, \dots, n\}$ by $E_i(f)$ where $E_i(f) := \{ j \in \{1, \dots, m\} \,|\, f(\mathbf{v}_h, \mathbf{y}_h; \Omega_h) \neq 0$ for some $(\mathbf{v}_h, \mathbf{y}_h) \in V_j^h \times Y_i^h \}$, denoting the symbolic nonzero dependence of terms in f on $\{V_1^h, \dots, V_m^h\} \times \{Y_1^h, \dots, Y_n^h\}$.*

Remark 5.2. In Definition 5.1, the equation map $\mathbf{E}(f)$ implies that the operator, f, operates on well-defined spaces on Ω_h (with topology $T_D \in \mathcal{T}$). Hence, there is a root topology, T_r, which links all topologies of variables in $\mathbf{E}(f)$. That is, $I_{r \to D}$ is complete and $I_{r \to K}$ is nonnull for each $T_k \Leftarrow \mathbf{E}(f)$.

Encoding the operators of f, the equation map $\mathbf{E}(f)$ enables the generalization of many lower-level routines and operations which require knowledge of the weak form structure, such as linearized system assembly, merger, etc. In addition to the symbolic dependence outlined, the equation map $\mathbf{E}(f)$ of f also labels the structure of each term as *identity*, *trace*, or *full*. This relates to the pairing of vector variables, which, due to operators, may share components in different ways (see Definition 5.3). For example, the standard inner-product of two vector variables may be classified as having *trace* structure, while the standard inner-product of two scalar variables is a *full* structure operator.

DEFINITION 5.3 (equation map structure). *Let $\mathbf{E}(f)$ be the equation map of the operator f given in Definition* 5.1. *Considering the variable pairing $V_j \times Y_i$, for some $j \in E_i(f)$, suppose each $\mathbf{v}_h \in V_j^h$ and $\mathbf{y}_h \in Y_i^h$ takes $\mathbf{v}_h : \Omega_h \to \mathbb{R}^s$ and $\mathbf{y}_h : \Omega_h \to \mathbb{R}^r$. Letting $\mathbf{v}_h^{n,k} \in V_j^h$ denote the nth basis function in the kth component and, similarly, letting $\mathbf{y}_h^{m,p} \in Y_i^h$ denote the mth basis function in the pth component, the structure of the variable pairing is defined as*

$$\begin{aligned} \textit{(Identity) if } & f(\mathbf{v}_h^{n,k}, \mathbf{y}_h^{m,p}) \neq 0 \textit{ for some } n = m,\ k = p, \\ \textit{(Trace) if } & f(\mathbf{v}_h^{n,k}, \mathbf{y}_h^{m,p}) \neq 0 \textit{ for some } n, m, k = p, \\ \textit{(Full) if } & f(\mathbf{v}_h^{n,k}, \mathbf{y}_h^{m,p}) \neq 0 \textit{ for some } n, k, m, p, \end{aligned}$$

where the least generic structure applicable is selected for the specific functional f.

While the equation mapping of an operator enables the construction of a generic template for FEM calculations, its dependence on a single domain, Ω_h, is limiting, particularly in multiphysics contexts where multiple domains are linked together. As a result, the multiphysics operator $\boldsymbol{f}$ is enabled through the formation of equation map sets where the final structure is the union of each equation map automatically evaluated from the individual maps provided, i.e., $\mathbf{E}(\boldsymbol{f}) := \bigcup_k \mathbf{E}(f_k)$. An example of this process is illustrated in Figure 5.1. In these cases, operations are done on each individual equation map, $\mathbf{E}(f_k)$, and their results are combined to emulate the final mapping, $\mathbf{E}(\boldsymbol{f})$.

5.3. Matrix assembly and monolithic integration. The final weak form system shown in (3.7) may be solved in a variety of ways. This is achieved, in part,

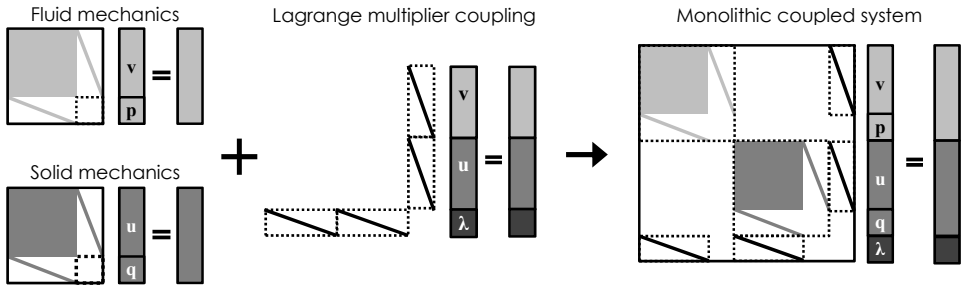


FIG. 5.1. *Illustration of three separate equation maps—incompressible Navier–Stokes with velocity and pressure variables* $(\mathbf{v}, p)$*, incompressible solid mechanics with displacement and pressure variables* $(\mathbf{u}, q)$*, and Lagrange multiplier coupling,* λ*, between* $(\mathbf{v}, \mathbf{u})$*—combined into a single equation map to form the monolithically coupled system used in FSI; see section* 8.2.

by allowing single-physics or other operators to be integrated into a single monolithic problem which may be solved implicitly. Importantly, not all operators of the total problem $\boldsymbol{f}$ need be monolithic, thereby allowing partitioned solving of some/all physics systems.

For problems with monolithic integration, their implicit solution requires the formulation of a Jacobian matrix (or some approximation to it) which may be inverted or iteratively solved to give the solution or a solution update (as in Newton–Raphson methods [57, 32]). Similar to other structures, we construct an object-set $\mathcal{M}$ of matrices. Each matrix, $M \in \mathcal{M}$, is assembled on an underlying equation mapping $\mathbf{E}(\boldsymbol{f})$ which may be composed of a simple physical system or a more complex multiphysics system spanning multiple domains and variables. From the equation map definition, we seek to assemble M in compressed sparse row (CSR) format [27] by determining the connectivity between a series of rows and columns. In this sense, the matrix,

$$M = \{\mathscr{V}_r, \mathscr{V}_c, \mathscr{E}\}, \tag{5.1}$$

is a graph with row/column vertices ($\mathscr{V}_r$ and $\mathscr{V}_c$) and edges ($\mathscr{E}$) which provide their connectivity. In the case of structurally simple equation maps—such as that which arises from scalar Laplace—the matrix construction process is straightforward. In this case, if the solution variable is $U = \{\mathbf{U}, T\} \in \mathcal{V}$ with N_N nodes/modes, then by computing the inverse mapping of the topology array $\boldsymbol{E} \in T$ (the node-to-element map), $\boldsymbol{E}^{-1}(n) = \{e \in \{1, \ldots, N_E\} \mid n = \boldsymbol{E}(e, k),$ for some $k \in \{1, \ldots, K\}\}$, $n \in \{1, \ldots, N_N\}$ one can effectively determine the interdependence of nodes/modes on elements. Subsequently, the inverse map can be used to directly compute the matrix graph $M = \{\mathscr{V}_r, \mathscr{V}_c, \mathscr{E}\}$. Here $\mathscr{V}_r = \mathscr{V}_c$ is defined so that, for $n \in \{1, \ldots, N_N\}$, $\mathscr{V}_r(U, n) = n$ and

$$\mathscr{E}(n) = \{m \in \{1, \ldots, N_N\} \mid m = \boldsymbol{E}(e, k) \text{ for some } k \in \{1, \ldots, K\} \text{ and } e \in \boldsymbol{E}^{-1}(n)\}. \tag{5.2}$$

Remark 5.4. In the case of discontinuous Galerkin assembly, $\tilde{\boldsymbol{E}}^{-1}$ is defined and used in place of $\boldsymbol{E}^{-1}$ in (5.2) with $\tilde{\boldsymbol{E}}^{-1}(n) = \{e \in \{1, \ldots, N_E\} \mid n = \boldsymbol{E}(e_l, k)$ for some $k \in \{1, \ldots, K\},$ and $e_l \in \mathcal{N}(e)\}$, $n \in \{1, \ldots, N_N\}$,and $\mathcal{N}(e)$ denoting the set of neighboring element indices of e and e itself.

In a multiphysics environment, this approach must be adapted to deal with additional (potentially multidimensional) variables, multiple equation maps (i.e., $\boldsymbol{f} =$

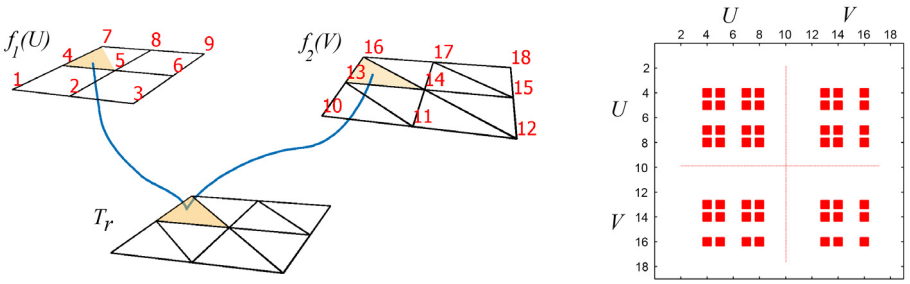


FIG. 5.2. *An illustration of monolithic matrix assembly procedure. Top left: Two topologies (quadrilateral and triangle types) with node numbers indicated. The equation maps for the individual systems are $E(f_1(U))$ and $E(f_2(V))$. When the two systems are coupled, the coupling operator provides the additional functionals, f_3, and equation maps, $E(f_3)$, linking the variables. Following the formation of the monolithic $\mathbf{E}(\boldsymbol{f})$, the matrix connectivity $\mathscr{E}$ can be determined through their common mapping to the root topology T_r. Assuming full sparsity in all mappings, for the element indicated in yellow in the above example rows and columns* 4, 5, 7, 8, 13, 14, *and* 16 *become populated, as shown on the right.*

$\{f_k\}$), and complex interfaces. However, the construction of more complex linear systems can be efficiently managed using both the equation mapping $\mathbf{E}(\boldsymbol{f})$ as well as the underlying interface mappings, $\mathcal{I}$. $\mathbf{E}(\boldsymbol{f})$ provides details of how the set of i-row variables, $\mathbf{Y} = \{Y_1, \dots, Y_i\} \subseteq \mathcal{V}$, as well as j-column variables, $\mathbf{V} = \{V_1, \dots, V_j\} \subseteq \mathcal{V}$, is interrelated. Using this data, we may construct a variable-matrix mapping for both rows and colums. That is, for any $Y = \{\boldsymbol{C}, T\} \in \mathbf{Y}$ and $n \in \{1, \dots N_N\}$, we define the row mapping $\mathscr{V}_r(Y, n) = R$, $R \in \{1, \dots, N_{\mathscr{V}_r}\}$, where $N_{\mathscr{V}_r}$ is the total number of rows (and similarly for the column mapping). We note that, in this case, R is uniquely paired to Y and n. This definition effectively orders how nodes/modes associated with specific variable spaces are mapped to the linear system.

Remark 5.5. Construction of $\mathscr{V}_r$ and $\mathscr{V}_c$ mappings follows element clustering, where all variables are ordered according to elements, or variable clustering, where each variable is ordered sequentially, depending on the parallelization strategy.

Subsequently, assembly is managed by constructing the matrix connectivity, $\mathscr{E}$, as seen in Definition 5.6. Here, the connection between row and column variables is extracted by first operating on each equation mapping $\mathbf{E}(f_k)$ resulting from functionals $f_k \in \boldsymbol{f}$. This provides a link to the root topology T_{r^k} over which all row and column variables of $\mathbf{E}(f_k)$ are related. Each element of the root topology T_{r^k} then acts as the common point, enabling mapping through the defined interfaces to determine the corresponding elements in the topology used by all variables in the mapping. By collating these nodes/modes, it is then possible to construct the collective connectivity, $\mathscr{E}$, as given in Definition 5.6 and illustrated in Figure 5.2.

DEFINITION 5.6 (matrix connectivity). *Let $\mathscr{E}$ denote the connectivity of a matrix $M \in \mathcal{M}$ based on the equation mapping $\mathbf{E}(\boldsymbol{f})$ with preconstructed matrix-topology mappings for both rows and columns ($\mathscr{V}_r$ and $\mathscr{V}_c$, respectively). Suppose $\mathbf{E}(\boldsymbol{f})$ contains row variable $\mathbf{Y} = \{Y_1, \dots, Y_I\}$ and column variable $\mathbf{V}\{V_1, \dots, V_J\}$ and the functional $\boldsymbol{f}$ is comprised of multiple functionals $f_k \in \boldsymbol{f}$ for which $\mathbf{E}(f_k)$ is a proper equation mapping with root topology $T_{r^k} = \{\boldsymbol{E}_r^k, B_r^k, N_{E,r}^k, N_{N,r}^k\}$ (see section* 5.2*). Then the connectivity $\mathscr{E}(R)$ of any row $R = \mathscr{V}_r(Y_i, n)$ corresponding to the variable $Y_i \in \mathbf{Y}$ and*

node/mode $n \in \{1, \dots, N_{N,i}\}$ *is defined as*

$$\mathscr{E}(R) = \Big\{ C \in \{1, \dots, N_{\mathscr{V}_c}\} \mid C = \mathscr{V}_c(V_j, m) \text{ for some } m \in \{\boldsymbol{E}_j(\boldsymbol{e}^{r^k,j}(e), p), p = 1, \dots, K_j\} \\ \boldsymbol{e}^{r^k,i}(e) \in \boldsymbol{E}_i^{-1}(n),\ j \in E_i(f_k),\ \text{and } f_k \in \boldsymbol{f} \Big\},$$

where $\boldsymbol{e}^{r^k,j} \in I_{r^k \to j}$, $\boldsymbol{e}^{r^k,i} \in I_{r^k \to i}$, *and* $\boldsymbol{E}_i \in T_i \in Y_i$ *and* $\boldsymbol{E}_j \in T_j \in V_j$.

This process is automatically handled using the internal FEM LOOP routine, thereby removing any burden on the developer's part in the assembly procedure. Looping over each functional $f_k \in \boldsymbol{f}$ and the resulting root topology T_{r^k}, we may construct local node/mode lists for each element (done through a generic `LocalElementCalculate` procedure). These values may then subsequently be used to update the connectivity structure $\mathscr{E}$ (done through a general `LocalElementAssemble` procedure).

With both row/column vertices and edges defined, the matrix seen in (5.1) can be constructed. When variable spaces are comprised of vectors, the structure of the equation mapping $\mathbf{E}(\boldsymbol{f})$ (see Definition 5.3) is used to evaluate the sparsity. Here, which components of a vector variable appear in the functional is outlined as either *full*, *trace,* or *identity*. This can be efficiently used to decipher which nodes/modes to include in $\mathscr{E}$.

5.4. Matrix and residual evaluations. Construction of different matrix/residual evaluations is achieved by selecting an appropriate `LocalElementCalculate` procedure for each functional $f_k \in \boldsymbol{f}$. This procedure is set locally by physics-specific code which manages the calculation of element-level matrix/residual entries. As variables may stem from different topologies, use different interpolation schemes, etc., each variable required for f_k is implicitly *prebuilt* at the lower level, efficiently linking required element-level data needed on the root topology T_{r^k}. In cases where no complex interfaces are present, this step involves assigning pointers to prebuilt data structures significantly limiting the overhead incurred by this generalized formulation. As a result, element-level construction within different physics-based modules involves standard quadrature loops and term evaluation procedures seen in all FEM codes.

Integration of element-level data through `LocalElementAssemble` procedures is managed by lower-level routines. Using the equation mapping structure $\mathbf{E}(f_k)$, entries of element-level data may be seamlessly integrated into the global matrix/residual structures. Adding additional terms within a single-physics routine is straightforward, requiring only adaptation of the equation map for the adapted functional f_k.

5.5. Solution strategies. While the summed operator, $\boldsymbol{f}$, introduced in (3.7) implies the monolithic integration of the full set of operators $f_k \in \boldsymbol{f}$ in a given multiphysics problem, it is often undesirable (and, indeed, limiting) to rely purely on monolithic forms. As detailed below, ***C*Heart** allows a broader range of solution strategies.

To solve the problems constructed, we enable the decomposition of the set of operators $\mathcal{F}$ into a solve group $\mathcal{S}$. The object-set $\mathcal{S}$ is constructed of a series of mutually exclusive subgroup objects, $S_g \in \mathcal{S}$. While $\mathcal{F}$ provides all operators and problems, $\mathcal{S}$ provides order and structure for carrying out the numerical solution. $\mathcal{S}$ is related to $\mathcal{F}$ by

(5.3)
$$S_g = \{\boldsymbol{f}_g\} \subseteq \mathcal{F}, \quad \mathcal{F} = \bigcup_g S_g, \quad S_g \in \mathcal{S}, \quad \text{and} \quad S_i \cap S_j = \emptyset, \quad i \neq j, \quad S_i, S_j \in \mathcal{S}.$$

Each subgroup is used to update the solutions for each of its problems, $\boldsymbol{f} \subseteq \mathcal{F}$. This is done by first assigning to each problem an update method, which may be a Newton–Raphson scheme, a linearized explicit scheme, or some combination, within a time step. With an update method defined, $\mathcal{C}$**Heart** updates the subgroup using either a *synchronous* or a *sequential* manner—analogous to the difference between Jacobi and Gauss–Seidel methods—which is based on previous data (synchronous) or the most current updated data (sequential).

The application of updates to the subgroup either may be done once or may be repeated using a fixed point criterion. The criteria used to terminate the fixed point iteration may be defined to depend on the size of computed updates, residual error, or some combination of the two (for more details, see [57]). In addition, when linked with a fixed point procedure, the subgroup may also use line search [57], which effectively scales all updates by a single scalar $\alpha \in (0, 1]$ to minimize residual error. These procedures defined for each S_g are applied to each member of $\mathcal{S}$ sequentially.

6. Graph partitioning and parallelization. Efficient parallelization of FEM-based procedures relies on an optimal mesh decomposition that minimizes communication overhead and achieves a good load balance amongst the processors. Often the decomposition utilizes graph partitioning [12, 45], whereby the computational domain is decomposed into a number of regions appropriate for the number of processors used. This requires the construction of a graph $G = \{V, E\}$ which is composed of a set of vertices V and edges E, and we seek to minimize the number of edge cuts required to split G into N-partitions (with N being the number of parallel processors). Then, a set of $\{V_1, \ldots, V_P\}$ vertices is returned to the P-processes and is used to define a final partitioning, $\{\tilde{V}_1, \ldots, \tilde{V}_P\}$.

Most FEMs use elements within a mesh (or, in our case, topology) to denote the vertices V of G, and define the edges E as the dual of the mesh,

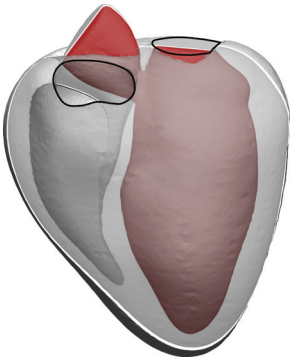
$$D(T)(e) = \{e_k \in \{1, \ldots, N_E\} \mid e \text{ and } e_k \text{ are neighbors } \}, \quad e \in \{1, \ldots, N_E\}, \tag{6.1}$$

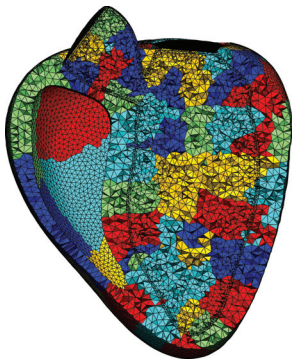
where $D(T)$ defines the neighboring elements for each element in the topology. The final graph $G = \{\{1, \ldots, e_T\}, D(T)\}$ can then be broken into N-partitions using graph partitioning libraries, determining the set of vertices (or elements) $\{V_1, \ldots, V_P\}$ assigned to each processor. While this approach is standard, there are numerous strategies for determining the final partitioning strategy $\{\tilde{V}_1, \ldots, \tilde{V}_P\}$.

A common approach is so-called *ghosting*, where each set of vertices V_k on rank k is used to construct the final element partition, $\tilde{V}_k$, by requiring that

$$\tilde{V}_k = \{e \in \{1, \ldots, N_E\} | e \text{ and } e_i \text{ share a corner, edge, or face for some } e_i \in V_k\}. \tag{6.2}$$

As a result, all coefficients which map into an element-level computation on V_k may have their contribution entirely computed on the kth processor. However, as FEM-based computations are, in general, heavily dependent on element-level computations for which the time is dependent on factors such as interpolations/quadrature schemes elected, this strategy has limited scalability potential. This is due to the facts that $V_k \subset \tilde{V}_k$ and that $\dim(V_k)/\dim(\tilde{V}_k) >> 1$ as $N >> 1$. That is, the number of surrounding elements grows, while the number of interior elements shrinks. In addition, the introduction of *ghost* elements requires communication of all coefficients affiliated with these elements—a burden which also increases as the number of processors increases. An alternative strategy, followed here, involves defining the element-level graph partitioning $\tilde{V} = V$ and adopting optimized communication strategies to relate data across ranks (as outlined in [87]).





Color	Rank
dark blue	1–25
light blue	26–50
green	51–75
yellow	76–100
red	100–125

FIG. 6.1. *Left: Tetrahedral mesh of the heart consisting of* $582K$ *elements. Right: Half-section of the heart with partitions labeled illustrating global graph partitioning across fluid-solid interfaces.*

In our case, a global graph is constructed over the set of topologies $\mathcal{T}_p \subseteq \mathcal{T}$, making the vertices $V = \{1, \ldots, N_{\mathcal{T}_p}\}$ of G equivalent to the total number of elements $N_{\mathcal{T}_p}$ across all topologies in $\mathcal{T}_p$. Here $\mathcal{T}_p$ uses a single representative topology for each topology group to ensure that all one-to-one topologies are equivalently partitioned. Moreover, we define a local to global element index map $g(T, e) = E$ mapping each element e of $T \in \mathcal{T}_p$ into a global element index $E \in V$. In our case, we define edges E as the dual of the mesh/interface constructs. That is,

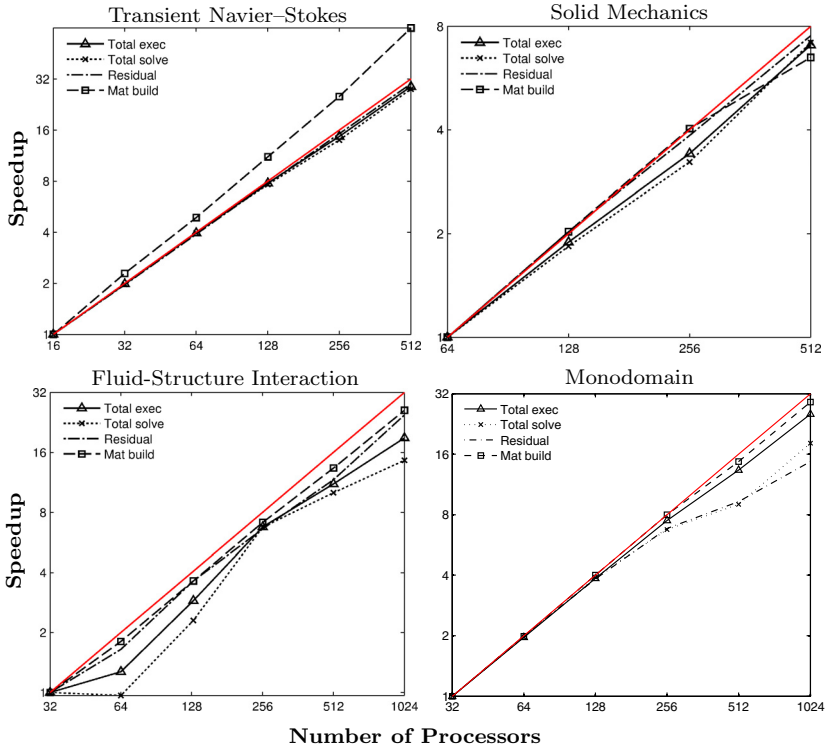
$$D(\mathcal{T}_p)(T_i, e) = \left\{ \begin{array}{ll} g(T_i, e_k), & \text{where } g(T_i, e) \text{ and } g(T_i, e_k) \text{ are neighbors} \\ g(T_j, e_k), & \text{where } e_k = \boldsymbol{e}^{i,j}(e) \text{ and } T_j \in \mathcal{T}_p \backslash T \end{array} \right\}. \tag{6.3}$$

The final graph $G = \{V, D(\mathcal{T}_p)\}$ is then broken into N-partitions and the elements $\{\tilde{V}_1, \ldots, \tilde{V}_P\} = \{V_1, \ldots, V_P\}$. By construction, the global graph enables the incorporation of multiphysics coupling across different topologies (see Figure 6.1) to be built into the partitioning strategy. To further improve balancing, different element-level weights may be assigned to each topology (by default, this is based on the calculated cost of computing a mass matrix), allowing a priori balancing based on different demands stemming from the physics.

7. Software implementation. ***C*Heart** is implemented in FORTRAN 2003 [42] as a distributed parallel application using Message Passing Interface (MPI) [29] for interprocess communication. Further detail regarding the implementation can be found in the online supplement (M101409_01.pdf [local/web 4.23MB]).

8. Results. To illustrate the efficiency of the base infrastructure presented, a series of parallel scalability tests is presented in section 8.1. To demonstrate the generality of the coupling infrastructure outlined, examples of major multiphysics problems in cardiac modeling research are presented in the online supplement (M101409_01.pdf [local/web 4.23MB]) along with a specific example for FSI given in section 8.2.

8.1. Parallel scalability performance. Parallel scaling tests were conducted for fluid mechanics, solid mechanics, monodomain, and FSI problems. The details of each simulation setup are summarized in Figure 8.1. These results were obtained on HECToR, the National Supercomputing Service provided by UK research councils. The Cray XE6 system features nodes with two 16-core AMD Opteron 2.3GHz Interlagos processors, each with access to 16GB RAM, and were coupled with a Cray Gemini

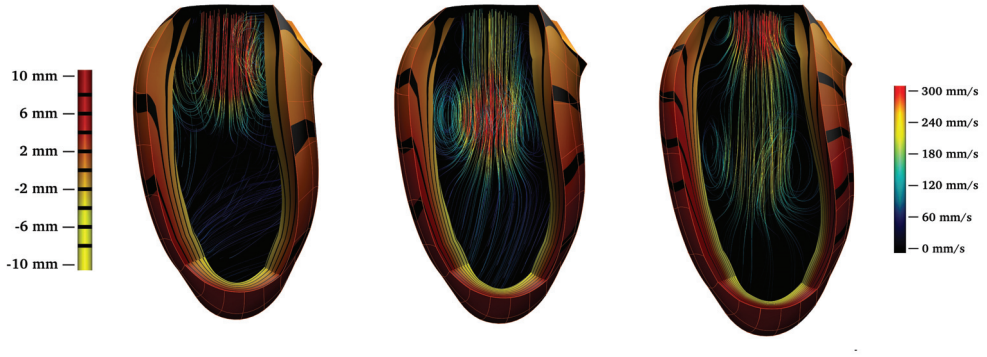


Physics Problem	Solver + Preconditioner	Mesh Size + DOFs	Element Type
Transient Navier–Stokes	GMRES + BFBT	262K elements 2.4M DOFs	Quadratic/Linear quadrilateral
Solid mechanics	GMRES + ASM	122K elements 3M DOFS	Quadratic/Linear hexahedra
Fluid-structure interaction (FSI)	GMRES + ASM	582K elements 7.4M DOFS	Quadratic/Linear Crouzeix–Raviart
Monodomain with ten Tusscher (2006)	GMRES + Boomer-AMG	63M elements 214M DOFS	Linear tetrahedra

FIG. 8.1. *Parallel scaling results for fluid and solid mechanics, FSI, and monodomain problems. The solid red line indicates linear scaling. Total execution time includes residual evaluation, matrix build, and solve steps.*

communications chip achieving node-to-node latency of 1–1.5μs. All simulations were repeated at least three times, and the optimal results were used for the analysis. The total execution time is defined as the sum of residual evaluation, matrix build, and solve times but excludes initial setup, partitioning, and I/O times. In Navier–Stokes, monodomain, and solid mechanics problems, linear scaling was observed in the total execution time. In the FSI problem, a deviation from linear scaling was observed, caused by the increased time spent in the linear solve steps. Residual and matrix build times did not exhibit a significant loss in scalability.

8.2. Passive filling of left ventricle: FSI. In this example, we consider the simulation of fluid-solid interaction in the passive left ventricle [68]. Due to an increased pressure (preload), the left ventricular chamber of the heart fills with blood.



(a) t = 0.15 s (b) t = 0.30 s (c) t = 0.45 s

FIG. 8.2. *Simulation of left ventricular filling under passive inflation, showing fluid streamlines in the chamber and displacement in the fiber direction at different time points through diastole.*

The passive tissue response of the heart wall is modeled using the nonlinear hyperelastic Costa law [14], which relies on the definition of local tissue microstructure using an additional fiber field [69]. The blood flow was modeled using a conservative ALE formulation [25] of the Navier–Stokes equations. Interpolation of both fluid and solid phases used inf-sup stable pairings, with the fluid interpolated by $\mathbb{P}^2 - \mathbb{P}^1$ and the solid by $\mathbb{Q}^3 - \mathbb{Q}^2$ elements.

Integration of this multiphysics system was achieved using monolithic coupling (see Figure 5.1), where an additional Lagrange multiplier allows coupling between the tetrahedral fluid and hexahedral solid grids (as detailed in [67]). The coupling is enforced weakly with respect to the trace of the velocity space on the fluid-solid boundary (i.e., using quadratic triangular elements). A coupling problem, introduced in section 4.4, was used to enforce coupling, and variables were equated on the common boundary using an injective interface onto the volume surface of the fluid and solid.

In addition to the monolithic fluid-solid-Lagrange multiplier system, the domain motion during filling requires the solution of a grid motion problem to appropriately adapt the fluid mesh. In this case, the ALE grid motion was solved using a simple Laplacian problem, where the boundary velocity was given by the solid motion (although other options are available; see Table 2.1). This problem was seamlessly integrated using a sequential fixed point solver strategy (see section 5.5), where the monolithic system is solved, the ALE system is solved, and the updates are applied simultaneously via a line search algorithm [57].

The results in Figure 8.2 show the passive filling cycle, marked by an initial pulse of blood flow, the formation of a ring vortex, and a relaxation of inflow, followed by a second pulse and second ring vortex [68].

9. Conclusion. In this work we have developed a generic and extensible computational modeling platform which successfully addresses the existing major challenges in multiphysics coupling. ***CHeart*** overcomes the implementational challenges of coupling different physics and mesh topologies through formal definitions of interfaces and operator-, boundary condition– or Lagrange multiplier–based coupling procedures in a generic manner. The matrix assembly procedure for the coupled system is handled through the use of equation maps, which, through its modular design, gives users full algorithmic configuration access without the need to engage in the lower-level details.

The example provided for the FSI problem in the heart demonstrates the flexible utility of partitioned and monolithic algorithms for multiphysics coupling combined with efficient parallel scalability.

REFERENCES

[1] T. Arts, T. Delhaas, P. Bovendeerd, X. Verbeek, et al., *Adaptation to mechanical load determines shape and properties of heart and circulation: The circadapt model*, Am. J. Physiol., 288 (2005), pp. H1943–H1954.

[2] L. Asner, M. Hadjicharalambous, R. Chabiniok, D. Peresutti, E. Sammut, J. Wong, G. Carr-White, P. Chowienczyk, J. Lee, A. King, N. Smith, R. Razavi, and D. Nordsletten, *Estimation of passive and active properties in the human heart using 3D tagged MRI*, Biomech. Model. Mechan., (2015), http://dx.doi.org/10.1007/s10237-015-0748-z.

[3] L. Asner, M. Hadjicharalambous, J. Lee, and D. Nordsletten, *Stacom challenge: Simulating left ventricular mechanics in the canine heart*, in Statistical Atlases and Computational Models of the Heart-Imaging and Modelling Challenges, Springer, New York, 2015, pp. 123–134.

[4] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. Curfman McInnes, B. Smith, and H. Zhang, *PETSc Users Manual*, Tech. report ANL-95/11 - Revision 3.3, Argonne National Laboratory, Lemont, IL, 2012.

[5] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc Web Page*, http://www.mcs.anl.gov/petsc (2012).

[6] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser, Basel, 1997, pp. 163–202.

[7] K. Bathe, *Finite Element Procedures*, Prentice–Hall, Englewood Cliffs, NJ, 1996.

[8] J. Boland and R. Nicolaides, *Stability of finite elements under divergence constraints*, SIAM J. Numer. Anal., 20 (1983), pp. 722–731, http://dx.doi.org/10.1137/0720048.

[9] J. Bonet and R. Wood, *Nonlinear Continuum Mechanics for Finite Element Analysis*, Cambridge University Press, Cambridge, UK, 1997.

[10] C. Bradley, A. Bowery, R. Britten, V. Budelmann, O. Camara, R. Christie, A. Cookson, A. F. Frangi, T. B. Gamage, T. Heidlauf, S. Krittian, D. Ladd, C. Little, K. Mithraratne, M. Nash, D. Nickerson, P. Nielsen, Ø. Nordbø, S. Omholt, A. Pashaei, D. Paterson, V. Rajagopal, A. Reeve, O. Röhrle, S. Safaei, R. Sebastián, M. Steghöfer, T. Wu, T. Yu, H. Zhang, and P. Hunter, *OpenCMISS: A multi-physics and multi-scale computational infrastructure for the VPH/physiome project*, Prog. Biophys. Mol. Biol., 107 (2011), pp. 32–47.

[11] F. Brezzi and R. Falk, *Stability of higher-order Hood–Taylor methods*, SIAM J. Numer. Anal., 28 (1991), pp. 581–590, http://dx.doi.org/10.1137/0728032.

[12] C. Chevalier and F. Pellegrini, *PT-scotch: A tool for efficient parallel graph ordering*, Parallel Comput., 34 (2008), pp. 318–331.

[13] A. N. Cookson, J. Lee, C. Michler, R. Chabiniok, E. Hyde, D. A. Nordsletten, M. Sinclair, M. Siebes, and N. P. Smith, *A novel porous mechanical framework for modelling the interaction between coronary perfusion and myocardial mechanics*, J. Biomech., 45 (2012), pp. 850–855.

[14] K. Costa, J. Holmes, and A. McCulloch, *Modelling cardiac mechanical properties in three dimensions*, Phil. Trans. R. Soc. Lond. A, 359 (2001), pp. 1233–1250.

[15] O. Coussy, *Poromechanics*, Wiley, New York, 2004.

[16] M. Crouzeix and P. Raviart, *Conforming and nonconforming finite element methods for solving the stationary Stokes equations* I, Rev. Française Automat. Informat. Recherche Opérationnelle Ser. Rouge, 7 (1973), pp. 33–75.

[17] A. de Vecchi, A. Gomez, K. Pushparajah, T. Schaeffter, D. A. Nordsletten, J. M. Simpson, G. P. Penney, and N. P. Smith, *Towards a fast and efficient approach for modelling the patient-specific ventricular haemodynamics*, Prog. Biophys. Mol. Biol., 116 (2014), pp. 3–10.

[18] A. de Vecchi, D. Nordsletten, E. Remme, H. Bellsham-Revell, G. Greil, J. M. Simpson, R. Razavi, and N. P. Smith, *Inflow typology and ventricular geometry determine efficiency of filling in the hypoplastic left heart*, Ann. Thorac. Surg., 94 (2012), pp. 1562–1569.

[19] K. Devine, E. Boman, R. Heaphy, B. Hendrickson, and C. Vaughan, *Zoltan data management services for parallel dynamic applications*, Comput. Sci. Eng., 4 (2002), pp. 90–97.
[20] F. Donati, C. A. Figueroa, N. P. Smith, P. Lamata, and D. A. Nordsletten, *Non-invasive pressure difference estimation from PC-MRI using the work-energy equation*, Medical Image Analysis, 26 (2015), pp. 159–172.
[21] J. Donea, *An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions*, Comput. Methods Appl. Mech. Engrg., 33 (1982), pp. 689–723.
[22] D. Endy and R. Brent, *Modelling cellular behaviour*, Nature, 409 (2001), pp. 391–395.
[23] M. Fink, S. A. Niederer, E. M. Cherry, F. H. Fenton, J. T. Koivumäki, G. Seemann, R. Thul, H. Zhang, F. B. Sachse, D. Beard, E. J. Crampin, and N. P. Smith, *Cardiac cell modelling: Observations from the heart of the cardiac physiome project*, Prog. Biophys. Mol. Biol., 104 (2011), pp. 2–21.
[24] L. Formaggia, J. F. Gerbeau, and C. Prud'homme, *LifeV Developer Manual*, The LifeV Project, EPFL, INRIA, Polytechnico Di Milano, 2001–2010; available online from https://github.com/lifev/lifev/blob/master/doc/manual/lifev-manual.pdf.
[25] L. Formaggia and F. Nobile, *Stability analysis of second-order time accurate schemes for ALE-FEM*, Comput. Methods Appl. Mech. Engrg., 193 (2004), pp. 4097–4116.
[26] L. Formaggia, F. Nobile, A. Quarteroni, and A. Veneziani, *Multiscale modelling of the circulatory system: A preliminary analysis*, Comput. Vis. Sci., 2 (1999), pp. 75–83.
[27] G. H. Golub and C. F. Van Loan, *Matrix Computations, Vol.* 3, Johns Hopkins University Press, Baltimore, MD, 2012.
[28] P. Gresho and R. Sani, *Incompressible Flow and the Finite Element Method* I: *Advection-Diffusion*, John Wiley and Sons, New York, 1998.
[29] W. Gropp, E. Lusk, and R. Thakur, *Using MPI-2: Advanced Features of the Message-Passing Interface*, MIT Press, Cambridge, MA, 1999.
[30] J. Guccione, K. Costa, and A. McCulloch, *Finite element stress analysis of left ventricular mechanics in the beating dog heart*, J. Biomech., 28 (1995), pp. 1167–1177.
[31] M. Hadjicharalambous, R. Chabiniok, L. Asner, E. Sammut, J. Wong, G. Carr-White, J. Lee, R. Razavi, N. Smith, and D. Nordsletten, *Analysis of passive cardiac constitutive laws for parameter estimation using 3D tagged MRI*, Biomech. Model. Mechan., 14 (2015), pp. 807–828.
[32] M. Hadjicharalambous, J. Lee, N. P. Smith, and D. A. Nordsletten, *A displacement-based finite element formulation for incompressible and nearly-incompressible cardiac mechanics*, Comput. Methods Appl. Mech. Engrg., 274 (2014), pp. 213–236.
[33] M. Heil and A. Hazel, *oomph-lib—an object-oriented multi-physics finite-element library*, in Fluid-Structure Interaction, Lect. Notes Comput. Sci. Eng. 53, Springer, Berlin, 2006, pp. 19–49.
[34] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, *An overview of the Trilinos project*, ACM Trans. Math. Softw., 31 (2005), pp. 397–423.
[35] C. W. Hirt, A. A. Amsden, and J. L. Cook, *An arbitrary Lagrangian-Eulerian computing method for all flow speeds*, J. Comput. Phys., 14 (1974), pp. 227–253.
[36] G. A. Holzapfel and R. W. Ogden, *Constitutive modelling of passive myocardium: A structurally based framework for material characterization*, Philos. Trans. A, 367 (2009), pp. 3445–3475.
[37] A. Huerta and W. Liu, *Viscous flow with large free surface motion*, Comput. Methods Appl. Mech. Engrg., 69 (1988), pp. 277–324.
[38] T. J. R. Hughes and A. N. Brooks, *A theoretical framework for Petrov-Galerkin methods, with discontinuous weighting functions: Application to the streamline upwind procedure*, in Finite Elements in Fluids, Vol. 4, John Wiley and Sons, Chichester, UK, 1982, pp. 47–65.
[39] T. J. R. Hughes and J. Lubliner, *On the one-dimensional theory of blood flow in the larger vessels*, Math. Biosci., 18 (1973), pp. 161–170.
[40] T. J. R. Hughes, W. Liu, and T. K. Zimmermann, *Lagrangian-Eulerian finite element formulation for incompressible viscous flows*, Comput. Methods Appl. Mech. Engrg., 29 (1981), pp. 329–349.
[41] E. R. Hyde, A. N. Cookson, J. Lee, C. Michler, A. Goyal, T. Sochi, R. Chabiniok, M. Sinclair, D. A. Nordsletten, J. Spaan, J. P. H. M. van den Wijngaard, M. Siebes, and N. P. Smith, *Multi-scale parameterisation of a myocardial perfusion model using whole-organ arterial networks*, Ann. Thorac. Surg., 42 (2014), pp. 797–811.
[42] ISO/IEC 1539-1:2004, *Fortran – Part* 1: *Base Language*, ISO, Geneva, Switzerland, 2004.
[43] G. Karniadakis and S. Sherwin, *Spectral/hp Element Methods for Computational Fluid Dy-*

namics, Oxford University Press, Oxford, UK, 2005.

[44] G. Karypis and V. Kumar, *A parallel algorithm for multilevel graph partitioning and sparse matrix ordering*, J. Parallel Distrib. Comput., 48 (1998), pp. 71–95.

[45] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392, http://dx.doi.org/10.1137/S1064827595287997.

[46] P. Keast, *Moderate degree tetrahedral quadrature formulas*, Comput. Methods Appl. Mech. Engrg., 55 (1986), pp. 339–348.

[47] R. C. P. Kerckhoffs, S. N. Healy, T. P. Usyk, and A. D. McCulloch, *Computational methods for cardiac electromechanics*, Proc. IEEE, 94 (2006), pp. 769–783.

[48] S. Krittian, U. Janoske, H. Oertel, and T. Bhlke, *Partitioned fluid–solid coupling for cardiovascular blood flow: Validation study of pressure driven fluid–domain deformation*, Ann. Biomed. Engrg., 38 (2010), pp. 2676–2689.

[49] S. A. Lambert, S. P. Näsholm, D. A. Nordsletten, C. Michler, L. Juge, J.-M. Serfaty, L. Bilston, B. Guzina, S. Holm, and R. Sinkus, *Bridging three orders of magnitude: Multiple scattered waves sense fractal microscopic structures via dispersion*, Phys. Rev. Lett., 115 (2015), 094301.

[50] J. Lee, A. Cookson, R. Chabiniok, S. Rivolo, E. Hyde, M. Sinclair, C. Michler, T. Sochi, and N. Smith, *Multiscale modelling of cardiac perfusion*, in Modeling the Heart and the Circulatory System, Springer, New York, 2015, pp. 51–96.

[51] J. Lee, D. Nordsletten, A. Cookson, S. Rivolo, and N. P. Smith, *In silico coronary wave intensity analysis: Application of an integrated one-dimensional and poromechanical model of cardiac perfusion*, Biomech. Model. Mechanobiol., (2016), pp. 1–21.

[52] A. Logg, K.-A. Mardal, and G. N. Wells, *Automated Solution of Differential Equations by the Finite Element Method*, Springer, New York, 2012.

[53] C.-H. Luo and Y. Rudy, *A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interaction*, Circ. Res., 68 (1991), pp. 1501–1526.

[54] J. Lyness and D. Jespersen, *Moderate degree symmetric quadrature rules for the triangle*, IMA J. Appl. Math., 15 (1975), pp. 19–32.

[55] L. E. Malvern, *Introduction to the Mechanics of Continuous Medium*, Prentice–Hall, Englewood Cliffs, NJ, 1969.

[56] M. McCormick, D. Nordsletten, D. Kay, and N. Smith, *Modelling left ventricular function under assist device support*, Internat. J. Numer. Methods Biomed. Engrg., 27 (2011), pp. 1073–1095.

[57] M. McCormick, D. Nordsletten, D. Kay, and N. Smith, *Fluid–solid coupling methods for simulating LV function through the full cardiac cycle*, J. Comput. Phys., 244 (2013), pp. 80–96.

[58] M. McCormick, D. Nordsletten, P. Lamata, and N. Smith, *Computational analysis of the importance of flow synchrony for cardiac ventricular assist devices*, Comput. Biol. Med., 49 (2014), pp. 83–94.

[59] R. McFarlane and I. V. Biktasheva, *Beatbox - a computer simulation environment for computational biology of the heart*, in Proceedings of the BCS Int. Acad. Conf., 2008, pp. 98–110.

[60] C. Michler, A. N. Cookson, R. Chabiniok, E. Hyde, J. Lee, M. Sinclair, T. Sochi, A. Goyal, G. Vigueras, D. A. Nordsletten, and N. P. Smith, *A computationally efficient framework for the simulation of cardiac perfusion using a multi-compartment Darcy porous-media flow model*, Internat. J. Numer. Methods Biomed. Engrg., 29 (2013), pp. 217–232.

[61] M. Nash and P. Hunter, *Computational mechanics of the heart*, J. Elasticity, 61 (2000), pp. 113–141.

[62] S. A. Niederer, E. Kerfoot, A. P. Benson, M. O. Bernabeu, O. Bernus, C. Bradley, E. M. Cherry, R. Clayton, F. H. Fenton, A. Garny, E. Heidenreich, S. Land, M. Maleckar, P. Pathmanathan, G. Plank, J. F. Rodríguez, I. Roy, F. B. Sachse, G. Seemann, O. Skavhaug, and N. P. Smith, *Verification of cardiac tissue electrophysiology simulators using an N-version benchmark*, Philos. Trans. A, 369 (2011), pp. 4331–4351.

[63] A. Nijenhuis and H. Wilf, *Combinatorial Algorithms for Computers and Calculators*, Academic Press, New York, 1978.

[64] F. Nobile, *Numerical Approximation of Fluid-Structure Interaction Problems with Application to Haemodynamics*, Ph.D. thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2001.

[65] D. Noble, *Modeling the heart—from genes to cells to the whole organ*, Science, 295 (2002), pp. 1678–1682.

[66] D. Nordsletten, P. Hunter, and N. Smith, *Conservative and non-conservative arbitrary Lagrangian-Eulerian forms for ventricular flows*, Internat. J. Numer. Methods Biomed. Engrg., 56 (2008), pp. 1457–1463.

[67] D. Nordsletten, D. Kay, and N. Smith, *A non-conforming monolithic finite element method for problems of coupled mechanics*, J. Comput. Phys., 20 (2010), pp. 7571–7593.

[68] D. Nordsletten, M. McCormick, P. Kilner, D. Kay, and N. P. Smith, *Fluid-solid coupling for the investigation of diastolic and systolic human left ventricular function*, Internat. J. Numer. Methods Biomed. Engrg., 27 (2011), pp. 1017–1039.

[69] D. Nordsletten, S. Niederer, M. Nash, P. Hunter, and N. P. Smith, *Coupling multi-physics models to cardiac mechanics*, Prog. Biophys. Mol. Biol., 104 (2011), pp. 77–88.

[70] D. Nordsletten, N. Smith, and D. Kay, *A preconditioner for the finite element approximation to the arbitrary Lagrangian–Eulerian Navier–Stokes equations*, SIAM J. Sci. Comput., 32 (2010), pp. 521–543, http://dx.doi.org/10.1137/08072958X.

[71] The OpenFOAM Foundation, *OpenFOAM - The Open Source CFD Toolbox - User Guide*, OpenFOAM Foundation, Bracknell, UK, available online from http://www.openfoam.com/documentation/user-guide/ (2016).

[72] F. Pellegrini and J. Roman, *Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs*, in High-Performance Computing and Networking, Lecture Notes in Comput. Sci. 1067, H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot, eds., Springer, Berlin, Heidelberg, 1996, pp. 493–498.

[73] J. Pitt-Francis, P. Pathmanathan, M. O. Bernabeu, R. Bordas, J. Cooper, A. G. Fletcher, G. R. Mirams, P. Murray, J. M. Osborne, A. Walter, S. J. Chapman, A. Garny, I. M. M. van Leeuwen, P. K. Maini, B. Rodríguez, S. L. Waters, J. P. Whiteley, H. M. Byrne, and D. J. Gavaghan, *Chaste: A test-driven approach to software development for biological modelling*, Comput. Phys. Commun., 180 (2009), pp. 2452–2471.

[74] J. P. Schmidt, S. L. Delp, M. A. Sherman, C. A. Taylor, V. S. Pande, and R. B. Altman, *The Simbios national center: Systems biology in motion*, Proc. IEEE, 96 (2008), pp. 1266–1280.

[75] G. Seemann, F. B. Sachse, M. Karl, D. L. Weiss, V. Heuveline, and O. Dössel, *Framework for modular, flexible and efficient solving the cardiac bidomain equations using PETSc*, in Progress in Industrial Mathematics at ECMI 2008, Mathematics in Industry, A. D. Fitt et al., eds., Springer, Berlin, Heidelberg, 2010, pp. 363–369.

[76] S. J. Sherwin, V. Franke, J. Peiró, and K. Parker, *One-dimensional modelling of a vascular network in space-time variables*, J. Engrg. Math., 47 (2003), pp. 217–250.

[77] Y. Shi and T. Korakianitis, *Numerical simulation of cardiovascular dynamics with left heart failure and in-series pulsatile ventricular assist device*, Art. Organs, 30 (2006), pp. 929–948.

[78] Y. Shi, P. Lawford, and R. Hose, *Review of zero-D and 1-D models of blood flow in the cardiovascular system*, Biomed. Eng. Online, 10 (2011), 33.

[79] N. P. Smith, A. J. Pullan, and P. J. Hunter, *An anatomically based model of transient coronary blood flow in the heart*, SIAM J. Appl. Math., 62 (2002), pp. 990–1018, http://dx.doi.org/10.1137/S0036139999355199.

[80] K. H. ten Tusscher and A. V. Panfilov, *Alternans and spiral breakup in a human ventricular tissue model*, Am. J. Physiol. Heart Circ. Physiol., 291 (2006), pp. H1088–H1100.

[81] N. A. Trayanova, *Whole-heart modeling applications to cardiac electrophysiology and electromechanics*, Circ. Res., 108 (2011), pp. 113–128.

[82] E. J. Vigmond, C. Clements, D. McQueen, and C. Peskin, *Effect of bundle branch block on cardiac output: A whole heart simulation study*, Prog. Biophys. Mol. Biol., 97 (2008), pp. 520–542.

[83] E. J. Vigmond, M. Hughes, G. Plank, and L. J. Leon, *Computational tools for modeling electrical activity in cardiac tissue*, J. Electrocardiol., 36 (2003), pp. 69–74.

[84] G. Vigueras, I. Roy, A. Cookson, J. Lee, N. Smith, and D. Nordsletten, *Toward GPGPU accelerated human electromechanical cardiac simulations*, Int. J. Numer. Methods Biomed. Eng., 30 (2014), pp. 117–134.

[85] N. Westerhof, J.-W. Lankhaar, and B. E. Westerhof, *The arterial Windkessel*, Med. Biol. Eng. Comput., 47 (2009), pp. 131–141.

[86] J. H. Yang and J. J. Saucerman, *Computational models reduce complexity and accelerate insight into cardiac signaling networks*, Circ. Res., 108 (2011), pp. 85–97.

[87] M. Zhou, O. Sahni, H. Kim, C. Figueroa, C. A. Taylor, M. S. Shephard, and K. E. Jansen, *Cardiovascular flow simulation at extreme scale*, Comput. Mech., 46 (2009), pp. 71–82.

[88] O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method*, Elsevier, Amsterdam, 2000.