IMPLEMENTATION DETAILS OF THE LEVEL SET TWO-PHASE NAVIER–STOKES EQUATIONS IN PROTEUS*

ALISTAIR BENTLEY†, NIALL BOOTLAND‡, ANDREW WATHEN†, AND CHRISTOPHER KEES§

Abstract. In this Technical Report, we outline the implementation of the two-phase Navier–Stokes equations in the Reynolds Averaged Two Phase Navier–Stokes (RANS2P) module of the Proteus toolkit. Specifically, we present the weak formulation used in Proteus, as well as the toolkit's techniques for stabilization and weak enforcement of Dirichlet boundary conditions. We end with an overview of the non-linear and linear solver methods used to solve the resulting system of equations.

Key words. finite elements, two-phase flow, level set, Navier–Stokes, preconditioner

AMS subject classifications. 65M60, 65M22, 65F10, 76D05, 76T10

1. Introduction. This Technical Report outlines the implementation of the two-phase Navier–Stokes equations in the Reynolds Averaged Two Phase Navier–Stokes (RANS2P) module of the Proteus toolkit (`http://proteustoolkit.org`). Section 2 offers a brief recap of the continuous conservative level set method used to solve dynamic two-phase flow problems in the RANS2P module. Next, Section 3 highlights the discrete non-linear Navier–Stokes equations that arise at each time-step of the discrete level set approach. Finally, Section 4 explores the methods used to solve the resulting non-linear and linearized systems of equations.

2. Overview of the Continuous Level set Method. In this section, we briefly overview the continuous level set method employed in Proteus. Most details unrelated to the Navier–Stokes problem are omitted in this report. Interested readers can find a complete description of the method in [11].

2.1. Level set. Consider a domain $\Omega \subset \mathbb{R}^n$ $(n = 2, 3)$ occupied by two immiscible fluids — air and water. For ease of exposition, all the methods discussed in this paper will be described in 2D, but extend naturally to 3D. Let Ω_w denote the segment of the domain containing water and Ω_a the segment of the domain containing air such that $\overline{\Omega} = \overline{\Omega}_w \cup \overline{\Omega}_a$ and $\Omega_w \cap \Omega_a = \emptyset$.
Next we define a continuous function ϕ such that

$$\Omega_w = \{\mathbf{x} : \phi(\mathbf{x}, t) < 0\} \quad \text{and} \quad \Omega_a = \{\mathbf{x} : \phi(\mathbf{x}, t) > 0\}. \tag{1}$$

The zero level set of ϕ defines the interface, $\Gamma = \overline{\Omega}_w \cap \overline{\Omega}_a$, between the two fluids implicitly. That is, $\Gamma = \{\mathbf{x} : \phi(\mathbf{x}) = 0\}$. Coupling the fluids' velocity, $\mathbf{u}$, with this level set function via a transport equation,

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0 \text{ in } \Omega, \tag{2}$$

*This publication is based on work supported by the EPSRC Centre for Doctoral Training in Industrially Focused Mathematical Modelling (EP/L015803/1) in collaboration with the US Army Coastal and Hydraulics Laboratory and HR Wallingford. Permission was granted by the Chief of Engineers to publish this information.

†Department of Mathematical Sciences, Clemson University, Martin Hall, Clemson, SC 29634, USA (abentle@clemson.edu).

‡Mathematical Institute, University of Oxford, Radcliffe Observatory Quarter, Woodstock Road, Oxford, OX2 6GG, UK (bootland@maths.ox.ac.uk, wathen@maths.ox.ac.uk).

§Coastal and Hydraulics Laboratory, US Army Engineer Research and Development Center, 3909 Halls Ferry Road, Vicksburg, MS 39180-6133, USA (christopher.e.kees@usace.army.mil).

determines the evolution in time of the fluids in Ω. Note that, throughout this report, bold letters will denote vector quantities.

2.2. Navier–Stokes. Next we consider a variable density and viscosity Navier–Stokes equation for incompressible fluids in $\Omega \times [0 \times T]$. Proteus allows two different formulations for the two phase problem. The first uses the kinematic viscosity

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \frac{\nabla p}{\rho} - \nabla \cdot (2\nu \nabla^s \mathbf{u}) = \mathbf{g} \text{ in } \Omega, \tag{3}$$

$$\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega, \tag{4}$$

where $\mathbf{u} = \begin{pmatrix} u & v \end{pmatrix}^T$ is the velocity, p is the pressure, ν is the kinematic viscosity, ρ is the density, $\mathbf{g}$ is the gravitational acceleration, and $\nabla^s \mathbf{u} = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T)$ is the symmetric gradient tensor. The second uses the dynamic viscosity

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho\, \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \nabla \cdot (2\mu \nabla^s \mathbf{u}) = \rho\, \mathbf{g} \text{ in } \Omega, \tag{5}$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega, \tag{6}$$

where μ is the dynamic viscosity.
Let ρ_a, ρ_w and ν_a, ν_w denote the density and viscosity of the air and water respectively. To describe the variable density and viscosity, we define ρ, ν and μ as

$$\rho = \rho_a H(\phi) + \rho_w (1 - H(\phi)) \ , \ \nu = \nu_a H(\phi) + \nu_w (1 - H(\phi)), \tag{7}$$

$$\text{and } \mu = \rho_a \nu_a H(\phi) + \rho_w \nu_w (1 - H(\phi)), \tag{8}$$

where H is a Heaviside function

$$H(\phi) = \begin{cases} 1 & \text{if } \phi > 0 \\ \frac{1}{2} & \text{if } \phi = 0 \\ 0 & \text{if } \phi < 0 \end{cases}. \tag{9}$$

Boundary conditions for (3)–(4) or (5)–(6), can be either Dirichlet ($\partial\Omega_D$) or Neumann ($\partial\Omega_N$), where

$$\mathbf{u} = \mathbf{u}_D \text{ on } \partial\Omega_D, \tag{10}$$

$$\mathbf{n} \cdot \overline{\boldsymbol{\sigma}} = \mathbf{h} \quad \text{on } \partial\Omega_N, \tag{11}$$

and $\overline{\boldsymbol{\sigma}} = -\frac{p}{\rho}\mathbf{I} + 2\nu \nabla^s \mathbf{u}$ (kinematic formulation) or $\overline{\boldsymbol{\sigma}} = -p\mathbf{I} + 2\mu \nabla^s \mathbf{u}$ (dynamic formulation) is the stress tensor.
Along the dynamic interface between the fluids, $\Gamma(t) = \overline{\Omega}_w \cap \overline{\Omega}_a$, we have

$$\mathbf{u}_w - \mathbf{u}_a = 0, \tag{12}$$

$$(\overline{\boldsymbol{\sigma}}_w - \overline{\boldsymbol{\sigma}}_a) \cdot \mathbf{n} = \mathbf{f}, \tag{13}$$

where $\mathbf{n}$ is the outward normal vector for the water phase. Moreover, $\mathbf{f}$ is a force from the surface tension between the fluids such that

$$\mathbf{f} = \mathbf{n} \frac{\gamma \kappa}{\rho} \quad \text{or} \quad \mathbf{f} = \mathbf{n}\, \gamma \kappa, \tag{14}$$

for the kinematic and dynamic formulations respectively. Here, γ is the air-water surface tension coefficient and κ is the mean curvature of Γ.

To define the variational form, let $\mathbf{V}_T(0,T;\mathbf{V}(\Omega))$ denote an appropriate function space for the vector-quantity velocity across time and space and $M_T(0,T;M(\Omega))$ denote an appropriate function space for the pressure across time and space.
The weak formulation of equations (3)–(4) requires finding $\mathbf{u} \in \mathbf{V}_T(0,T;\mathbf{V}(\Omega))$ and $p \in M_T(0,T;M(\Omega))$ such that

$$
\begin{aligned}
\int_\Omega \frac{\partial \mathbf{u}}{\partial t}\mathbf{v}\, d\Omega - \int_\Omega (\mathbf{u}\otimes\mathbf{u}) : \nabla \mathbf{v}\, d\Omega = &- \int_\Omega \frac{1}{\rho}\nabla p \cdot \mathbf{v}\, d\Omega \\
&- \int_\Omega (2\nu \nabla^s \mathbf{u}) \cdot \nabla \mathbf{v}\, d\Omega \\
&+ \int_\Omega \mathbf{g}\cdot\mathbf{v}\, d\Omega \\
&- \int_{\partial\Omega} ((\mathbf{u}\otimes\mathbf{u})\cdot\mathbf{n})\cdot\mathbf{v}\, \partial\Omega \\
&+ \int_{\partial\Omega} (2\nu\nabla^s\mathbf{u}\cdot\mathbf{n})\cdot\mathbf{v}\, \partial\Omega \\
&+ \tau \int_\Gamma \frac{\kappa}{\rho}\,\mathbf{n}_\Gamma\cdot\mathbf{v}\, d\Gamma,
\end{aligned} \tag{15}
$$

$$
\int_\Omega \mathbf{u}\cdot\nabla q\, d\Omega = -\int_{\partial\Omega} (\mathbf{u}\cdot\mathbf{n})\, q\, \partial\Omega, \tag{16}
$$

for all $\mathbf{v} \in \mathbf{V_T}(0,T;\mathbf{V}(\Omega))$ and $q \in M_T(0,T;M(\Omega))$.
For the formulation (5)–(6), the weak form requires finding $\mathbf{u} \in \mathbf{V}_T(0,T;\mathbf{V}(\Omega))$ and $p \in M_T(0,T;M(\Omega))$ such that

$$
\begin{aligned}
\int_\Omega \rho\,\frac{\partial \mathbf{u}}{\partial t}\mathbf{v}\, d\Omega + \int_\Omega \rho\,(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v}\, d\Omega = &- \int_\Omega \nabla p\cdot\mathbf{v}\, d\Omega \\
&- \int_\Omega (2\mu\nabla^s\mathbf{u})\cdot\nabla\mathbf{v}\, d\Omega \\
&+ \int_\Omega \rho\,\mathbf{g}\cdot\mathbf{v}\, d\Omega \\
&+ \int_{\partial\Omega} (2\mu\nabla^s\mathbf{u}\cdot\mathbf{n})\cdot\mathbf{v}\, \partial\Omega \\
&+ \tau\int_\Gamma \kappa\,\mathbf{n}_\Gamma\cdot\mathbf{v}\, d\Gamma,
\end{aligned} \tag{17}
$$

$$
\int_\Omega \mathbf{u}\cdot\nabla q\, d\Omega = -\int_{\partial\Omega}(\mathbf{u}\cdot\mathbf{n})\, q\, \partial\Omega, \tag{18}
$$

for all $\mathbf{v} \in \mathbf{V_T}(0,T;\mathbf{V}(\Omega))$ and $q \in M_T(0,T;M(\Omega))$.
The difference between (15)–(16) and (17)–(18) includes the inner-product density scaling and the momentum flux boundary condition in (15) that arises from integrating the advection term by parts.
Also, these weak forms are different from the standard variational form of Navier–Stokes because equations (16) and (18) have been integrated by parts instead of the pressure gradient ∇p term in the momentum equation.

2.3. Redistancing. Once the Navier–Stokes equations have been solved, the Eikonal equation is used to redistance the level set function ϕ to ϕ_d such that

$$\|\nabla \phi_d\| = 1 \text{ in } \Omega, \tag{19}$$

$$\phi_d = 0 \text{ on } \Gamma. \tag{20}$$

2.4. Volume of fluid. Next, to determine the volume fraction, the linear scalar conservation law

$$\frac{\partial \widehat{H}}{\partial t} + \nabla \cdot (\widehat{H}\mathbf{u}) = 0, \tag{21}$$

is solved for $\widehat{H}$.

2.5. Mass correction. Finally, the volume fraction and signed distance functions are linked through the non-linear equation

$$\kappa \Delta \phi' = H(\phi_d + \phi') - \widehat{H} \text{ in } \Omega, \tag{22}$$

$$\nabla \phi' \cdot \mathbf{n} = 0 \qquad \text{on } \partial\Omega, \tag{23}$$

and solved for ϕ' to find a mass conserving adjustment.

3. Numerical Methods for Navier–Stokes. In this section we outline the discrete non-linear Navier–Stokes equation that arises in the discrete level set method. To establish the fluid phases, we assume there is a continuous level set phase function ϕ (recall (1)) from the previous time-step.

3.1. Discrete Navier–Stokes. First, assume that the physical domain Ω and its boundary $\partial\Omega$ are fixed during $[0, T]$. Next, assume the time interval $[0, T]$ is partitioned into a sequence of steps $0 = t_0 < \cdots < t_n < t_{n+1} < \cdots < t_N = T$, and time intervals $\Delta t_n = t_n - t_{n-1}$. In practice, this partition is done dynamically during the simulation, but since our focus is on the discrete Navier–Stokes problem we assume these values are set in advance.

At each time-step, the domain Ω is partitioned into an unstructured mesh $\mathcal{T}_h$ of N_e simplex elements K_i. Each element has diameter h_e and $\Omega = \cup_{i=1}^{N_e} K_i$. The mesh $\mathcal{T}_h$ also partitions $\partial\Omega$ into n_e segments $\partial\Omega_i$. To avoid confusion, ∂K_i will denote the boundary of an element K_i, while $\partial\Omega_i$ is used to denote the boundary of the partitioning $\mathcal{T}_h$. When calculating element or boundary integrals over the entire domain, we use the notation

$$\int_{\Omega'} = \sum_{i=1}^{N_e} \int_{K_i} \quad \text{and} \quad \int_{\partial\Omega'} = \sum_{i=1}^{n_e} \int_{\partial\Omega_i}. \tag{24}$$

For the mesh partitioning $\mathcal{T}_h$, we define our finite element spaces as continuous degree k polynomials of equal order

$$\mathbf{V}^h = \{\mathbf{u}^h \in \mathbf{V}(\Omega) \cap C^0 : \mathbf{u}^h|_{\Omega_e} \in P^k(\Omega_e) \text{ for all } \Omega_e \in \mathcal{T}_h\} \tag{25}$$

$$M^h = \{w^h \in M(\Omega) \cap C^0 : w^h|_{\Omega_e} \in P^k(\Omega_e) \text{ for all } \Omega_e \in \mathcal{T}_h\}. \tag{26}$$

Let $\{\boldsymbol{\varphi}_i\}_{i=1}^{n_v}$ and $\{\psi_i\}_{i=1}^{n_p}$ denote bases for $\mathbf{V}^h$ and M^h respectively. When referring to the components of an element of $\{\boldsymbol{\varphi}_i\}_{i=1}^{n_v}$, we use the notation $\boldsymbol{\varphi}_i = \begin{pmatrix} \varphi_i^u & \varphi_i^v \end{pmatrix}^T$. The RANS2P module allows the user to specify the polynomial order, k, for the

velocity and pressure spaces. A frequently used element pairing uses $k = 1$ for the velocity and pressure (commonly referred to as P1–P1). This choice requires pressure stabilization terms, which are discussed in Subsection 3.2. Furthermore, since the Dirichlet boundary conditions are enforced weakly (see Subsection 3.3 for details), these spaces do not satisfy any particular boundary conditions.
At each time-step t_n, we use the discrete solution from the previous m time-steps to approximate the temporal derivative $\mathbf{D}_t\mathbf{v}_n$ with a backward differentiation formula (BDF) given by

$$\mathbf{D}_t\mathbf{v}_n = \alpha\mathbf{v}_n + \sum_{i=1}^{m} \beta^i \mathbf{v}_{n-i}, \tag{27}$$

where the subscript n indicates the time of the solution variable. That is, $\mathbf{u}_n$ indicates the discrete solution at time t_n. Also, note that $m = 1$, $\alpha = \frac{1}{\Delta t_n}$, and $\beta^1 = -\frac{1}{\Delta t_n}$ gives the backward Euler formula.
Using $\mathbf{V}^h$ and M^h, we can define the discrete approximation to the continuous problem (15)–(16) at time t_n as: find $\mathbf{u}_n^h \in \mathbf{V}^h$ and $p_n^h \in M^h$ such that

$$\begin{aligned}
\int_\Omega \mathbf{D}_t\mathbf{u}_n^h\mathbf{v}^h \, d\Omega - \int_\Omega (\mathbf{u}_n^h \otimes \mathbf{u}_n^h) : \nabla\mathbf{v}^h \, d\Omega = &- \int_\Omega \frac{1}{\rho}\nabla p_n^h \cdot \mathbf{v}^h \, d\Omega \\
&- \int_\Omega (\nu\nabla^s\mathbf{u}_n^h) \cdot \nabla\mathbf{v}^h \, d\Omega \\
&+ \int_\Omega \mathbf{g}\cdot\mathbf{v}^h \, d\Omega \\
&- \int_{\partial\Omega} ((\mathbf{u}_n^h \otimes \mathbf{u}_n^h)\cdot\mathbf{n})\cdot\mathbf{v}^h \, \partial\Omega \\
&+ \int_{\partial\Omega} (\nu\nabla^s\mathbf{u}_n^h\cdot\mathbf{n})\cdot\mathbf{v}^h \, \partial\Omega
\end{aligned} \tag{28}$$

$$\int_\Omega \mathbf{u}_n^h \cdot \nabla q^h \, d\Omega = -\int_{\partial\Omega} (\mathbf{u}_n^h\cdot\mathbf{n})\, q^h \, \partial\Omega \tag{29}$$

for all $\mathbf{v}^h \in \mathbf{V}_h$ and $q^h \in M_h$. The discrete form of (17)–(18) takes a similar form. Implementation of the boundary conditions is discussed in Subsection 3.3.

3.2. Stabilization Methods. It is well known that the Galerkin finite element method is unstable for advection dominated (e.g. high Reynolds number) flows. Moreover, pressure stabilization is needed when finite element pairs are not inf-sup stable (e.g. P1–P1). In this section, we outline the stabilization terms used to address these issues in Proteus' RANS2P module. For more information on the stabilization methods discussed herein, see [9], [5] and [15].
RANS2P uses a variation of algebraic sub-grid scale (ASGS) stabilization. ASGS stabilization uses weighted element integrals of the strong residual tested against an adjoint differential operator to stabilize (28)–(29). To illustrate, consider the strong residuals of equations (3)–(4). For (4) the residual is

$$\mathbf{r}_p = \frac{\partial u_{n-1}^h}{\partial x} + \frac{\partial v_{n-1}^h}{\partial y} = \nabla\cdot\mathbf{u}_{n-1}^h. \tag{30}$$

If there was a mass source, it would be part of this term. For the two components of the vector equation (3), the residuals are

$$\mathbf{r}_u = \mathbf{D}_t u^h_{n-1} + \mathbf{u}^h_{n-1} \cdot \nabla u^h_n + \frac{1}{\rho}\frac{\partial p^h_n}{\partial x} - \nabla \cdot (\nu \nabla u^h_n) - g_1, \tag{31}$$

$$\mathbf{r}_v = \mathbf{D}_t v^h_{n-1} + \mathbf{u}^h_{n-1} \cdot \nabla v^h_n + \frac{1}{\rho}\frac{\partial p^h_n}{\partial y} - \nabla \cdot (\nu \nabla v^h_n) - g_2. \tag{32}$$

For linear finite elements, the diffusion terms in (31)–(32) are omitted because second derivatives vanish. Also, (31)–(32) use different forms of the advection and diffusion operators from (3). Finally, the solution from the previous time-step is used to calculate the divergence term in (30) and the advective velocity in (31)–(32). Analogous residuals can be computed for the dynamic formulation (5)–(6).

Next we define the adjoint operator $\mathcal{L}^*$. Rather than acting on the solution functions $\mathbf{u}^h$ and p^h, $\mathcal{L}^*$ acts on the test functions $\mathbf{v}^h$ and q^h. The components of $\mathcal{L}^*$ are

$$\begin{aligned}
\mathcal{L}^*_{uu}\mathbf{v}^h &= \mathbf{u}^h_{n-1} \cdot \nabla v^h_1 - g_1 v^h_1, & \mathcal{L}^*_{vv}\mathbf{v}^h &= \mathbf{u}^h_{n-1} \cdot \nabla v^h_2 - g_2 v^h_2, \\
\mathcal{L}^*_{up} q^h &= \frac{\partial q^h}{\partial x}, & \mathcal{L}^*_{vp} q^h &= \frac{\partial q^h}{\partial y}, \\
\mathcal{L}^*_{pu}\mathbf{v}^h &= \frac{1}{\rho}\frac{\partial v^h_1}{\partial x}, & \mathcal{L}^*_{pv}\mathbf{v}^h &= \frac{1}{\rho}\frac{\partial v^h_2}{\partial y}.
\end{aligned} \tag{33}$$

Each element $e \in \mathcal{T}_h$ also has the stabilization weighting terms

$$\tau_v(e) = \left(\frac{4\nu}{h^2} + \frac{2\|\mathbf{u}^h_{n-1}\|_2}{h_e} + |\alpha| \right)^{-1}, \tag{34}$$

$$\tau_p(e) = \rho \left(4\nu + 2\|\mathbf{u}^h_{n-1}\|_2 h_e + |\alpha| h_e^2 \right). \tag{35}$$

Recall that α is the coefficient from the BDF formula (27) and h_e is element e's diameter. In the dynamic formulation, μ replaces ν.

Finally, the stabilization terms are added to the global system (28)–(29) (or analogous terms for the dynamic representation). Specifically,

$$\int_{\Omega'} \tau_v(K_i)\ (\mathbf{r}_u\ \mathcal{L}^*_{uu} + \mathbf{r}_v\ \mathcal{L}^*_{vv})\mathbf{v}_h + \int_{\Omega'} \tau_p(K_i)\ \mathbf{r}_p\ (\mathcal{L}^*_{pu} + \mathcal{L}^*_{pv})\mathbf{v}_h, \tag{36}$$

is added to the right hand side of (28) and

$$\int_{\Omega'} \tau_v(K_i)\ (\mathbf{r}_u\ \mathcal{L}^*_{up} + \mathbf{r}_v\ \mathcal{L}^*_{vp})q_h, \tag{37}$$

is added to the right hand side of (29).

A separate numerical diffusion term is also used to account for discontinuity capturing. Specifically, to the conservation of momentum equation (28) we add,

$$\int_{\Omega'} q^* \nabla \mathbf{u}^h : \nabla \mathbf{v}^h\ d\Omega, \quad \text{where} \quad q^* = C_{dc}\sqrt{(\mathbf{r}^e_u)^2 + (\mathbf{r}^e_v)^2}\ h_e^2, \tag{38}$$

and $C_{dc} > 0$ is a constant.

The formulas for τ_v, τ_p and q^* above are the simplest versions used in RANS2P. While the details are omitted, Proteus has more sophisticated methods for calculating these quantities that account for elements with wide aspect ratios. For more information on metric based approaches to calculating the τ and q^* quantities see [10] and [14].

3.3. Boundary Conditions. Proteus allows users to specify Dirichlet conditions, advective flux conditions for the mass ($\mathbf{u}^h \cdot \mathbf{n}$) and momentum ($(\mathbf{u}^h \otimes \mathbf{u}^h) \cdot \mathbf{n}$), as well as the diffusive flux condition ($\nu \nabla^s \mathbf{u}^h \cdot \mathbf{n}$).
Care must be taken to ensure consistency across these different boundary condition specifications. Consider, for example, a channel flow in which a fluid enters an inflow boundary ($\mathbf{u} \cdot \mathbf{n} < 0$), travels through a characteristic boundary region ($\mathbf{u} \cdot \mathbf{n} = 0$) and leaves through an outflow boundary ($\mathbf{u} \cdot \mathbf{n} > 0$). On the inflow boundary, a Dirichlet condition for the velocity and an appropriate advective mass flux condition are specified. On the characteristic boundary, Dirichlet boundary conditions should be specified with zero advective mass flux. Finally, on the outflow region, Neumann conditions are prescribed, which require specifying the diffusive flux and a Dirichlet pressure condition.
Proteus typically enforces the Dirichlet boundary conditions weakly (see [3] and [1] for more details). On the Dirichlet boundary (recall $\mathbf{u} = \mathbf{u}_D$ on $\partial\Omega_D$), with no prescribed flux conditions, the following integral terms are added

$$\int_{\partial\Omega'} (-2\nu \nabla \mathbf{u}^h \cdot \mathbf{n}) \cdot \mathbf{v}^h \, ds \qquad \text{(diffusive flux condition)} \tag{39}$$

$$+ \int_{\partial\Omega'} (-2\nu \nabla \mathbf{v}^h \cdot \mathbf{n}) \cdot (\mathbf{u}^h - \mathbf{u}) \, ds \qquad \text{(adjoint diffusive flux penalty)} \tag{40}$$

$$+ \int_{\partial\Omega'} \gamma (\mathbf{u}^h - \mathbf{u}_D) \cdot \mathbf{v}^h \, ds, \qquad \text{(penalty term)} \tag{41}$$

where $\gamma = \dfrac{C_b |2\nu|}{h}$ for a penalty constant C_b.

4. Solvers. The discrete two-phase Navier–Stokes problem at time t_n was outlined in Section 3. Next we overview the non-linear and linear techniques used to solve this discrete problem. Note that we will focus on the kinematic viscosity formulation, but these ideas also apply to the dynamic formulation.
Applying standard finite element techniques to the combined equations of (28)–(29), along with the additional terms described in Subsections 3.2 and 3.3, gives a non-linear vector function $\mathbf{F}(\mathbf{x})$ where $\mathbf{x} = \begin{pmatrix} \mathbf{u} & \mathbf{p} \end{pmatrix}^T$. The solution to this problem is an $\mathbf{x}^*$ such that

$$\mathbf{F}(\mathbf{x}^*) = 0, \tag{42}$$

and can be solved using a fixed point Newton iteration.
Before discussing the implementation details, we first recap Newton's method. Recall that Newton's method is a fixed point iteration, that creates a sequence $\{\mathbf{x}_0, \mathbf{x}_1, \ldots\}$ of approximations to the true solution $\mathbf{x}^*$. Provided the initial guess $\mathbf{x}_0$ is sufficiently close to $\mathbf{x}^*$ and $\mathbf{F}$ is sufficiently regular, this sequence converges to $\mathbf{x}^*$ quadratically.
Note also that earlier in this report, the subscript was used to identify the time-step. In this section, subscripts are used to indicate the Newton iteration. If it is necessary to refer to the solution from the previous time-step, we will us the notation $\widetilde{\mathbf{x}}^*$.
Newton's method is derived from the linear approximation of the vector function $\mathbf{F}(\mathbf{x})$ centered at a point $\mathbf{x}_k$

$$\mathbf{F}(\mathbf{x}) \approx \mathbf{F}(\mathbf{x}_k) + D\mathbf{F}(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) \tag{43}$$

where $D\mathbf{F}(\mathbf{x}_k)$ is the Jacobian matrix with entries

$$[D\mathbf{F}(\mathbf{x}_k)]_{i,j} = \left[\frac{\partial F_i(\mathbf{x}_k)}{\partial x_j}\right]_{i,j}. \tag{44}$$

It follows that if $\mathbf{F}(\mathbf{x}_k) + D\mathbf{F}(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = 0$, then $\mathbf{F}(\mathbf{x}_{k+1}) \approx 0$ where

$$\mathbf{x}_{k+1} = \mathbf{x}_k - D\mathbf{F}(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{x}_k). \tag{45}$$

That is, each Newton iteration generates an updated solution approximation $\mathbf{x}_{k+1}$ using the previous iterate $\mathbf{x}_k$ and the solution update $-D\mathbf{F}(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{x}_k) = \Delta\mathbf{x}_k$. As we discuss in more detail below, the bulk of the work in applying Newton's iteration rests in finding $\Delta\mathbf{x}_k$.

Next we discuss how Newton's method is implemented in RANS2P. The initial guess for the Newton iteration $\mathbf{x}_0$, comes from the solution at the previous time-step $\widetilde{\mathbf{x}}^*$, except for the initial time-step $t_0 = 0$ where $\mathbf{x}_0 = \mathbf{0}$. Provided the time-steps are sufficiently close together, these initial guesses will be in the Newton solver's radius of convergence. After the first iteration, the updated solution approximation $\mathbf{x}_k$ is used as input to find $\mathbf{x}_{k+1}$.

Once $\mathbf{x}_k$ has been selected, next we build the Jacobian matrix (44). To illustrate how the various terms are calculated, consider the conservative form of the non-linear advection operator where the test function is $\mathbf{w} = \begin{pmatrix} w & z \end{pmatrix}^T$

$$\begin{aligned} \int_\Omega (\mathbf{u}\otimes\mathbf{u}) : \nabla\mathbf{w}\, d\Omega &= \int_\Omega \begin{pmatrix} u^2 & uv \\ vu & v^2 \end{pmatrix} : \begin{pmatrix} \partial w_x & \partial w_y \\ \partial z_x & \partial z_y \end{pmatrix} d\Omega \\ &= \int_\Omega u^2\partial w_x + uv\partial w_y + uv\partial z_x + v^2\partial z_y\, d\Omega. \end{aligned} \tag{46}$$

Taking the Fréchet derivative of (46) with respect to u yields

$$\begin{aligned} \frac{\partial \int_\Omega (\mathbf{u}\otimes\mathbf{u}) : \nabla\mathbf{w}\, d\Omega}{\partial u} &= \int_\Omega 2\, u\, \partial w_x\, \delta u + v\, \partial w_y\, \delta u + v\, \partial z_x\, \delta u\, d\Omega \\ &= \int_\Omega \begin{pmatrix} 2u & v \end{pmatrix}\, \delta u\, \nabla w + \begin{pmatrix} v & 0 \end{pmatrix}\, \delta u\, \nabla z\, d\Omega \\ &= \sum_{i=1}^{n_u} a_i \int_\Omega \begin{pmatrix} 2u & v \end{pmatrix}\, \varphi_i^u\, \nabla w + \begin{pmatrix} v & 0 \end{pmatrix}\, \varphi_i^u\, \nabla z\, d\Omega. \end{aligned} \tag{47}$$

In the final step, we've used the fact that δu is an element of the discrete finite element space, and can be represented as $\delta u = \sum_{i=1}^{n_u} a_i\varphi_i^u$. Moreover, values for u and v are approximated using $\mathbf{x}_k$. This approach is also used to calculate the diffusion, divergence and gradient terms, but these terms are easier to handle because they are linear.

The Jacobian also requires derivatives of the strong residuals (30)–(32). For instance, the derivatives of $\mathbf{r}_p$ are

$$\frac{\partial \mathbf{r}_p}{\partial p} = 0, \qquad \frac{\partial \mathbf{r}_p}{\partial u} = \frac{\partial \delta u}{\partial u}, \quad \text{and} \quad \frac{\partial \mathbf{r}_p}{\partial v} = \frac{\partial \delta v}{\partial v}. \tag{48}$$

Similarly, the derivatives of $\mathbf{r}_u$ are

$$\frac{\partial \mathbf{r}_u}{\partial p} = \frac{1}{\rho}\frac{\partial \delta p}{\partial x}, \qquad \frac{\partial \mathbf{r}_u}{\partial u} = \alpha\delta u + \widetilde{\mathbf{x}}^* \cdot \nabla\delta u, \quad \text{and} \quad \frac{\partial \mathbf{r}_u}{\partial v} = 0, \tag{49}$$

where $\widetilde{\mathbf{x}}^*$ is the solution from the previous time-step and α is defined in (27). Further, note that the constant terms (e.g. those with β coefficients) drop out of the expressions involving $\mathbf{D}_t$ and that the constant body source $\mathbf{g}$ vanishes.

Once $D\mathbf{F}(\mathbf{x}_k)$ is built, our attention turns to finding $\Delta\mathbf{x}_k = -D\mathbf{F}(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{x}_k)$. For most problems of interest, computing $D\mathbf{F}(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{x}_k)$ directly is impractical. Instead, we solve the sparse linear system

$$D\mathbf{F}(\mathbf{x}_k)\Delta\mathbf{x}_k = -\mathbf{F}(\mathbf{x}_k), \tag{50}$$

to provide the solution update $\Delta\mathbf{x}_k = \begin{pmatrix}\delta\mathbf{u} & \delta\mathbf{p}\end{pmatrix}^T$, with an iterative Krylov technique such as GMRES.

As mentioned before, solving the linear system (50) takes up most of the work for each Newton iteration. While the values of the Jacobian matrix change at each Newton iteration, the linear system (50) always has the block structure

$$\begin{pmatrix}\mathbf{A} & B^T \\ B & C\end{pmatrix}\begin{pmatrix}\delta\mathbf{u} \\ \delta\mathbf{p}\end{pmatrix} = -\mathbf{F}(\mathbf{x}_k), \tag{51}$$

with the system matrix $D\mathbf{F}(\mathbf{x}_k)$ being a sparse matrix.

In RANS2P, the linear system (51) is solved using PETSc and PETSc for Python ([2], [6]). Typically GMRES is used with Gram–Schmidt orthogonalization and restarts after every 500 iterations.

The system (51), is often preconditioned using a Schur complement approach. In this case, the pressure and velocity unknowns are identified, and PETSc's fieldsplit settings are used to specify the saddle point structure of $D\mathbf{F}(\mathbf{x}_k)$. This allows us to create the upper triangular Schur complement preconditioner

$$P^{-1} = \begin{pmatrix}\widehat{\mathbf{A}}^{-1} & -\widehat{\mathbf{A}}^{-1}B^T\widehat{S}^{-1} \\ 0 & \widehat{S}^{-1}\end{pmatrix}. \tag{52}$$

The operators $\widehat{\mathbf{A}}$ and $\widehat{S}$ are approximations to the operators $\mathbf{A}$ and $S = C - B\mathbf{A}^{-1}B^T$. Depending on the problem's size and nature, a variety of different options are available for approximating the action $\widehat{\mathbf{A}}^{-1}$. The reader is referred to [7] for more details. If the problem size permits, we often take $\widehat{\mathbf{A}}^{-1} = \mathbf{A}^{-1}$ and use a direct solver package like SuperLU or SuperLU_DIST [12]. For larger problems, an additive Schwarz or multigrid preconditioner is used.

For (52) to be an effective preconditioner, one must have an effective approximation for $\widehat{S}^{-1}$. For single-phase Navier–Stokes problems, a number of good approximations for $\widehat{S}^{-1}$ have been developed (see [7] and [13]). These methods do not, however, translate well to two-phase problems.

One method that has shown itself to be effective for dynamic two-phase Navier–Stokes problems (provided short time-steps are taken) is the SIMPLE [16] approximation

$$\widehat{S}_p^{-1} = \left(C - B\ \mathrm{diag}(\mathbf{A})^{-1}B^T\right)^{-1}. \tag{53}$$

This approximation is available in PETSc where it is called selfp.

Recently, some novel approximations of the inverse Schur complement, S^{-1}, suited for the two-phase Navier–Stokes problem were developed and presented in [4]. Since they offer the possibility to improve the effectiveness of the preconditioner (52), these methods are of particular interest to us. Due to the fact that RANS2P uses simplices

and stabilized finite element pairs, the two-phase pressure convection–diffusion (PCD) operator is more relevant for RANS2P than the least-squares commutator (LSC) approach.

The PCD inverse Schur complement operator takes the form

$$\widehat{S}_{\mathrm{PCD}}^{-1} = \left(Q^{(1/\mu)}\right)^{-1} + \left(A_p^{(1/\rho)}\right)^{-1} \left(N_p^{(1)} + \frac{1}{\Delta t} Q^{(1)}\right) \left(Q^{(1)}\right)^{-1}. \tag{54}$$

While two different scaling options for the PCD operator are presented in [4], our experience suggests (54) is the most effective version of the operator for both the dynamic and kinematic formulations.

All the components used in the PCD operator approximation are calculated on the pressure space and are defined, for a general scaling function α, as

$$Q_{i,j}^{(\alpha)} = \int_\Omega \alpha\ \psi_j\ \psi_i\ d\Omega, \tag{55}$$

$$A_{p;i,j}^{(\alpha)} = \int_\Omega \alpha\ \nabla\psi_j \cdot \nabla\psi_i\ d\Omega, \tag{56}$$

$$N_{p;i,j}^{(\alpha)} = \int_\Omega \alpha\ \left(\mathbf{w}_h \cdot \nabla\psi_j\right) \cdot \psi_i\ d\Omega, \tag{57}$$

where $\mathbf{w}_h$ is the velocity approximation from the previous non-linear iteration, and $\{\psi_i\}_{i=1}^{n_p}$ are the pressure basis functions.

In RANS2P, the PCD approximation to the Schur complement works best when the mass matrices $Q^{(\alpha)}$ are lumped or approximated with diagonal terms. Recall that a lumped mass-matrix is the diagonal matrix formed when all the mass terms in a row are added together. In other words,

$$Q_{\mathrm{lumped}}^{(\alpha)} = \mathrm{diag}(\mathbf{v}), \quad \text{where} \quad \mathbf{v}_i = \sum_{j=1}^{n_p} Q_{ij}^{(\alpha)}. \tag{58}$$

Since the fluid phases typically change at every time-step, the $\widehat{S}_{\mathrm{PCD}}^{-1}$ operator must be frequently rebuilt. This suggests an advantage for $\widehat{S}_p^{-1}$, since it is built using sub-blocks of the global linear system. Especially for dynamic simulations with many time-steps, $\widehat{S}_{\mathrm{PCD}}^{-1}$ must exhibit superior performance to compensate for its higher overhead.

Once the operators $\widehat{S}_p^{-1}$ and $\widehat{S}_{\mathrm{PCD}}^{-1}$ have been assembled, the work required to apply them is similar. To approximate $\widehat{S}_p^{-1}$, we use the HYPRE library [8] and apply one BoomerAMG V-cycle. This is an effective approach because $\widehat{S}_p$ is similar to a Laplace operator.

To apply $\widehat{S}_{\mathrm{PCD}}^{-1}$ (recall (54)), we attach a Python matrix shell to the PETSc Schur complement preconditioner (52). As with $\widehat{S}_p^{-1}$, we only apply the action of $\widehat{S}_{\mathrm{PCD}}^{-1}$ once per GMRES iteration. Since the mass matrices in $\widehat{S}_{\mathrm{PCD}}^{-1}$ are approximated with diagonal matrices, their action requires rescaling the appropriate vector. Meanwhile, the action of $(A_p^{(1/\rho)})^{-1}$ is approximated with one BoomerAMG V-cycle.

Once the linear system (50) is solved, $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$ and the residual $\mathbf{F}(\mathbf{x}_{k+1})$ are calculated. If $\|\mathbf{F}(\mathbf{x}_{k+1})\|_2$ is larger than a desired tolerance, the solution approximation $\mathbf{x}_{k+1}$ is updated with another Newton iteration. This continues until the desired tolerance is reached or the non-linear solver is determined to have failed.

5. Discussion. In this report we have outlined the methods used to set up and solve the two-phase Navier–Stokes equations in Proteus' RANS2P module. Section 2 gave a brief overview of RANS2P's continuous, conservative level set methodology for solving dynamic two-phase flow problems. We have focused on the Navier–Stokes equations, but also noted that the Navier–Stokes problem is just one part of a broader numerical method in which the fluid phases are advected dynamically over time. Then, in Section 3, we discussed the discrete form of the Navier–Stokes problem at a given time-step in the level set method. Also discussed in this section was the problem's weak form, the addition of stabilization terms and how Dirichlet boundary conditions are enforced weakly. Finally, Section 4 explored the linear and non-linear techniques used to solve the discrete Navier–Stokes system for a given time-step.

REFERENCES

[1] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, *Unified analysis of discontinous Galerkin methods for elliptic problems*, SIAM J. Numer. Anal., 39 (2002), pp. 1749–1779.

[2] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. Gropp, D. Karpeyev, D. Kaushik, M. Knepley, L. McInnes, K. Rupp, B. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc users manual*. http://www.mcs.anl.gov/petsc/petsc-current/docs/manual.pdf, 2016.

[3] Y. Bazilevs and T. J. R. Hughes, *Weak imposition of Dirichlet boundary conditions in fluid mechanics*, Comput. Fluids, 36 (2007), pp. 12–26.

[4] N. Bootland, A. Bentley, C. Kees, and A. Wathen, *Preconditioners for two-phase incompressible Navier–Stokes flow*, under review, (2017).

[5] R. Codina, *A stabilized finite element method for generalized stationary incompressible flows*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 2681–2706.

[6] L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo, *Parallel distributed computing using Python*, Adv. Water Resour., 34 (2011), pp. 1124–1139.

[7] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Oxford University Press, second ed., 2014.

[8] R. D. Falgout and U. M. Yang, *hypre: a library of high performance preconditioners*, in Computational Science — ICCS 2002: International Conference Amsterdam, The Netherlands, April 21–24, 2002 Proceedings, Part III, P. M. A. Sloot, A. G. Hoekstra, C. J. K. Tan, and J. J. Dongarra, eds., Springer Berlin Heidelberg, 2002, pp. 632–641.

[9] T. J. R. Hughes, *Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods*, Comput. Methods Appl. Mech. Engrg., 127 (1995), pp. 387–401.

[10] T. J. R. Hughes, M. Mallet, and A. Mizukami, *A new finite element formulation for computational fluid dynamics II. Beyond SUPG*, Comput. Methods Appl. Mech. Engrg., 54 (1986), pp. 341–355.

[11] C. E. Kees, I. Akkerman, M. W. Farthing, and Y. Bazilevs, *A conservative level set method suitable for variable-order approximations and unstructured meshes*, J. Comput. Phys., 230 (2011), pp. 4536–4558.

[12] X. S. Li, *An overview of SuperLU: algorithms, implementation, and user interface*, ACM Trans. Math. Software, 31 (2005), pp. 302–325.

[13] M. A. Olshanskii and Y. V. Vassilevski, *Pressure Schur complement preconditioners for the discrete Oseen problem*, SIAM J. Sci. Comput., 29 (2007), pp. 2686–2704.

[14] F. Shakib, T. J. R. Hughes, and Z. Johan, *A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier–Stokes equations*, Comput. Methods Appl. Mech. Engrg., 89 (1991), pp. 141–219.

[15] T. E. Tezduyar, *Stabilized finite element formulations for incompressible flow computations*, Adv. Appl. Mech., 28 (1992), pp. 1–44.

[16] M. ur Rehamn, C. Vuik, and G. Segal, *Simple-type preconditioners for the Oseen problem*, Int. J. Numer. Meth. Fluids, 61 (2009), pp. 432–452.