Streamline-Upwind Petrov-Galerkin Stabilization in PRONGHORN

April Novak

December 11, 2016

1 Introduction

The creation of high-fidelity multiphysics simulation tools for nuclear engineering applications is of high priority to support the development of advanced reactors, where traditionally-used software may be no longer be sufficiently accurate to characterize new reactor operating regimes. PRONGHORN is a Finite Element (FE) Computational Fluid Dynamics (CFD) code that is currently being developed on the Multiphysics Object-Oriented Simulation Environment (MOOSE) framework that solves the compressible Euler equations for the density, velocity, energy, and temperature of a fluid flowing in a porous media [1, 2]. Porous media models are of interest for thermal-hydraulic codes because no solid is explicitly meshed - all quantities, such as temperature, are averaged over the liquid and solid phases, reducing the number of finite elements needed from about 10^{10} to 10^6. While any reactor can be described as a porous media, the approach is especially desirable for modeling pebble bed reactors due to the natural analog between fuel pebbles and the pebbles of sand or rocks for which porous media theory was initially developed. All symbols used are defined in the Appendix.

PRONGHORN is under development to provide a thermal-hydraulics solver for advanced reactors that also easily couples to the other MOOSE codes to provide a framework under which multiphysics simulations are substantially simplified over the traditional approach of coupling two legacy codes. The majority of the physics in PRONGHORN have been encoded, but a remaining challenge is to stabilize the nonlinear equations governing fluid flow. Stability analysis of the Euler equations shows that there are nodal oscillations in density, pressure, and velocity when $Pe = Vh\rho C_p/2k > 1$. The Peclet number Pe characterizes the importance of advection to diffusion, and for reactor applications is typically of order $10^3 - 10^4$, and hence the Euler equations are highly unstable. This project applies the Streamline-Upwind Petrov-Galerkin (SUPG) method to stabilize the governing equations such that the Peclet number has no impact on stability, thus allowing simulation without the need for extremely fine spatial meshes. This report will discuss the governing equations, the finite element spatial discretization, and results showing the success of the implementation of the SUPG stabilization in PRONGHORN.

2 Governing Equations

This section provides the continuous and discretized governing equations solved in PRONGHORN. All fluids satisfy conservation of mass, momentum, and energy, and different variations on equations used to actually model fluid flow simply use different constitutive relationships for the stress tensor. The Navier-Stokes equations are the most commonly-used model for fluid flow, and assume a Newtonian constitutive relationship. The derivation of the Navier-Stokes equations is a very lengthy process that requires knowledge of continuum mechanics, and hence the bulk of the derivation has been given in Section 8.2 [5]. Instead of repeating this lengthy derivation, this section provides a high-level overview of these equations and how the porous media forms of the Navier-Stokes equations are obtained.

2.1 Continuous Equations

The equations governing fluid flow are conservation equations - namely, conservation of mass, momentum, and energy. By balancing the forces on an arbitrary continuum, and using a constitutive relationship for the stress tensor for a Newtonian fluid, the Navier-Stokes equations are:

$$
\begin{aligned}
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) &= 0 \\
\frac{\partial (\rho \vec{V})}{\partial t} + \nabla \cdot (\rho \vec{V} \otimes \vec{V}) &= \rho \vec{g} - \nabla P + \nabla \cdot \left[\mu \left(\nabla \vec{V} + (\nabla \vec{V})^T \right) - \frac{2\mu}{3} \nabla \cdot \vec{V} \mathbf{I} \right] \\
\frac{\partial (\rho E)}{\partial t} + \nabla \cdot ((\rho E + P) \vec{V}) &= \rho \vec{g} \cdot \vec{V} + \nabla \cdot \left\{ \left[\mu \left(\nabla \vec{V} + (\nabla \vec{V})^T \right) - \frac{2\mu}{3} \nabla \cdot \vec{V} \mathbf{I} \right] \vec{V} \right\} - \nabla \cdot \vec{q}
\end{aligned}
\tag{2.1}
$$

The first equation represents conservation of mass, the second conservation of momentum, and the third conservation of energy (all for the fluid). Because viscous effects are negligible in reactor applications, terms representing viscous forces can be neglected, giving a form of the Navier-Stokes equations called the Euler equations. To then obtain the porous-media form of the Euler equations, each occurrence of density is replaced by $\epsilon\rho$, where ϵ is the porosity, which denotes the fraction of the volume that is fluid. Then, special models are added to the momentum equation to account for friction and form drag in a porous media. The friction drag is related to the friction the fluid feels as it moves around pebbles, while form drag captures the "drafting" effect a fluid particle feels as it moves past a solid obstacle in a flow field. With these additions, the porous-media forms of the Euler equations become:

$$
\begin{aligned}
\epsilon \frac{\partial \rho_f}{\partial t} + \nabla \cdot (\epsilon \rho_f \vec{V}) &= 0 \\
\epsilon \frac{\partial (\rho_f \vec{V})}{\partial t} + \nabla \cdot (\epsilon \rho_f \vec{V} \vec{V}) + \nabla \cdot (\epsilon P \mathbf{I}) - \epsilon \rho_f \vec{g} + \mathcal{D} \mu \epsilon \vec{V} + \mathcal{F} \epsilon^2 \rho_f |\vec{V}| \vec{V} &= 0 \\
\epsilon \frac{\partial (\rho_f E)}{\partial t} + \nabla \cdot (\epsilon h_f \rho_f \vec{V}) - \epsilon \rho_f \vec{g} \cdot \vec{V} + \alpha (T_f - T_s) - q_f &= 0 \\
(1 - \epsilon) \rho_s C_{p,s} \frac{\partial T_s}{\partial t} - \nabla \cdot (\kappa_s \nabla T_s) + \alpha (T_s - T_f) - q_s &= 0
\end{aligned}
\tag{2.2}
$$

Another equation has been included to provide the solid temperature that results in heat conduction from the fuel to the fluid. Refer to Section 8.2 for the full derivation of these governing equations.

2.2 Discretized Equations

PRONGHORN uses the Finite Element Method (FEM) to solve the coupled set of equations given in Eq. (2.2). This report does not permit a lengthy discussion of the FEM, and instead the basis of this numerical method is deferred to Section 8.3. The FEM is a weighted residual method that requires the use of a weak form of the governing equations. This weak form is obtained by multiplying each term in the equation by an arbitrary weight function ψ and integrating over all space, applying the divergence rule when possible. Using the convention that angled brackets denote an inner product over area (with a unit normal implicitly included) and curved brackets an inner product over volume, the weak form of the governing equations is:

$$\left(\epsilon\frac{\partial\rho_f}{\partial t},\psi\right)-\left(\epsilon\rho_f\vec{V},\nabla\psi\right)+<\epsilon\rho_f\vec{V},\psi>=0$$

$$\left(\epsilon\frac{\partial\eta_x}{\partial t},\psi\right)-\left(\epsilon\eta_x\vec{V},\nabla\psi\right)+<\epsilon\eta_x\vec{V}+\epsilon P\vec{I},\psi>-\left(\epsilon P\vec{I},\nabla\psi\right)+$$

$$(\mathcal{D}\mu\epsilon V_x,\psi)+\left(\mathcal{F}\epsilon^2\rho_f|\vec{V}|V_x,\psi\right)-\left(\epsilon\rho_f g_x,\psi\right)=0 \tag{2.3}$$

$$\left((1-\epsilon)\frac{\partial(\rho_s C_{p_s}T_s)}{\partial t},\psi\right)+(\kappa_s\nabla T_s,\nabla\psi)-<\kappa_s\nabla T_s,\psi>+\left(\alpha(T_s-T_f),\psi\right)-(q_s,\psi)=0$$

$$\left(\epsilon\frac{\partial\gamma_f}{\partial t},\psi\right)-\left(\epsilon h_f\vec{\eta},\nabla\psi\right)+<\epsilon h_f\vec{\eta},\psi>-(\epsilon\vec{g}\cdot\vec{\eta},\psi)+\left(\alpha(T_f-T_s),\psi\right)-\left(q_f,\psi\right)=0$$

Only the x-direction momentum equation is shown for conciseness and momentum and energy have been introduced as separate variables. The equations can be solved on an unstructured mesh, and MOOSE has capabilities for any type of mesh that can be generated by a commercial meshing software, and hence is highly general.

3 Algorithms

This section discusses the method used to stabilize the governing equations in PRONGHORN. A detailed discussion of the Jacobian-Free Newton Krylov (JFNK) method used to solve the discretized equations is out of the scope of this project, and is deferred to Section 8.4. Stabilization is needed for advection-dominated flows because the Galerkin method is equivalent to a central difference method, which is plagued by oscillation problems. A flow is advection-dominated when it has a high Re or Pe number. The magnitude of Pe is influenced by the mesh size due to the presence of h_e in its definition. As long as some diffusion is present, then in theory, the mesh could be refined to a point at which $Pe < 1$, but at the great cost of computational time when usually only engineering-scale results are desired. For hyperbolic systems, for which $k = 0$, then $Pe = \infty$, and stabilization methods are required.

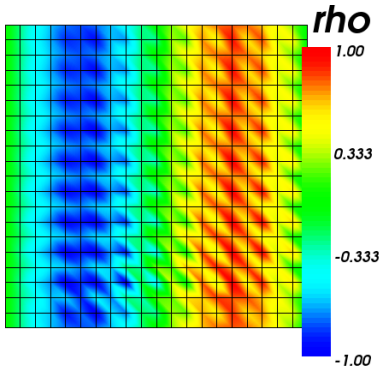


Figure 1. Solution to the `continuity_eqn_full` test without stabilization. The solution should be $\rho = \sin(2\pi x)\sin(2\pi t)$. Results are shown on $0 \le x \le 1,\ 0 \le y \le 1$ at $t = 1.15s$.

A dramatic example of the need for stabilization is obtained from a solution to the continuity equation, for which advection terms are present in the governing equations. As can be seen in Fig. 1, instabilities due to

the advection terms is generally characterized by nodal oscillations.

There are many ways to stabilize the Euler equations, but the most commonly-used method is the SUPG method [3,4]. The SUPG method is a form of upwinding that generalizes optimal 1-D approaches to higher dimensions while not being subject to the artificial diffusion of earlier upwind methods. The SUPG method essentially multiplies each term in the governing equations by $\tau\vec{V}$, where τ is a tuning parameter, and adds this to the original weak forms. In simplest terms, this adds a component to the weight function:

$$\psi \rightarrow \psi + \tau\vec{V} \cdot \nabla\psi \tag{3.1}$$

The motivation behind this can be seen by investigating the weak form of the diffusion kernel $\nabla^2 T$:

$$(\nabla T, \nabla\psi) \tag{3.2}$$

The weak form of the advection kernel $\vec{V} \cdot \nabla T$ is:

$$(\vec{V} \cdot \nabla T, \psi) \tag{3.3}$$

By inserting the stabilized shape function $\psi + \tau\vec{V} \cdot \nabla\psi$, the advective weak form becomes:

$$(\vec{V} \cdot \nabla T, \psi) + (\tau\vec{V} \cdot (\vec{V} \cdot \nabla T), \nabla\psi) \tag{3.4}$$

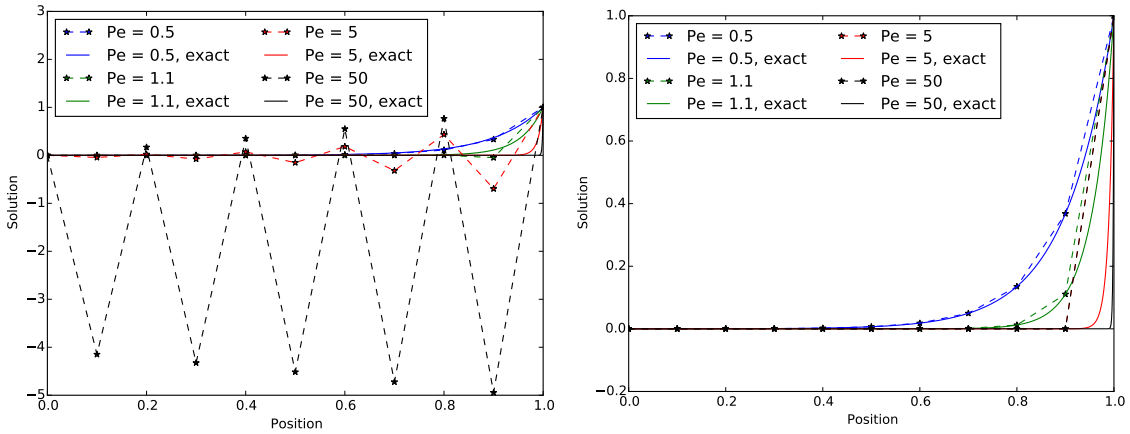
The presence of the two gradients, ∇T and $\nabla\psi$ in the diffusive weak form in Eq. (3.2) is, from a numerical standpoint, what gives the diffusion term nice numerical properties. By choosing the weight functions as in Eq. (3.1), the $\nabla T \nabla\psi$ term is obtained, thereby numerically adding "nice" properties that are very similar to the diffusion numerical properties. It is this similarity to solving a diffusion kernel that is allowed by this choice of shape functions that results in the SUPG stabilization being successful. The SUPG weak forms are given in Section 8.5.

4 Code Use

A full description of a PRONGHORN input file is beyond the scope of this report, but a brief summary can be given. Each input file specifies conservation of mass, momentum, and energy by including all physics kernels in those equations. An equation of state is specified in order to determine pressure in terms of other solution variables. Boundary conditions are then specified at inflow locations (no boundary conditions at outflow locations). The user selects the family and order of the finite element shape functions, and can either upload a mesh generated with commercial software or use the mesh-generation capability of MOOSE. Finally, the executioner is specified, which by default is preconditioned-JFNK. Post-processed results, such as averages or integrals of the solution, can also be specified to assess simulation results. Running a PRONGHORN input file gives results for the fluid density, velocity, and energy, and for the solid temperature.
It was requested that the code be submitted, but for export control reasons that can't be done (but I think you have access to PRONGHORN?). PRONGHORN is currently on the Idaho National Lab (INL) GitLab page. Likewise, there are about 55 tests in PRONGHORN, and submitting all of these is not necessary, but attached in Section 8.7 is the input to solve the 1-D convection-diffusion equation, with the results shown in Fig. 2.

5 Test Problems and Results

This section presents the test problems and results used to verify and validate the implementation of SUPG stabilization in PRONGHORN. In 1-D cases, SUPG stabilization is exact. Fig. 2 shows the difference between (a) no stabilization and (b) SUPG stabilization in the 1-D convection diffusion equation for various Pe. As can be seen, without stabilization, oscillations result for $Pe > 1$, with the amplitude of the oscillations increasing with the magnitude of Pe. With stabilization, however, nodally-exact results are obtained. This demonstrates the successful implementation of SUPG stabilization in 1-D.

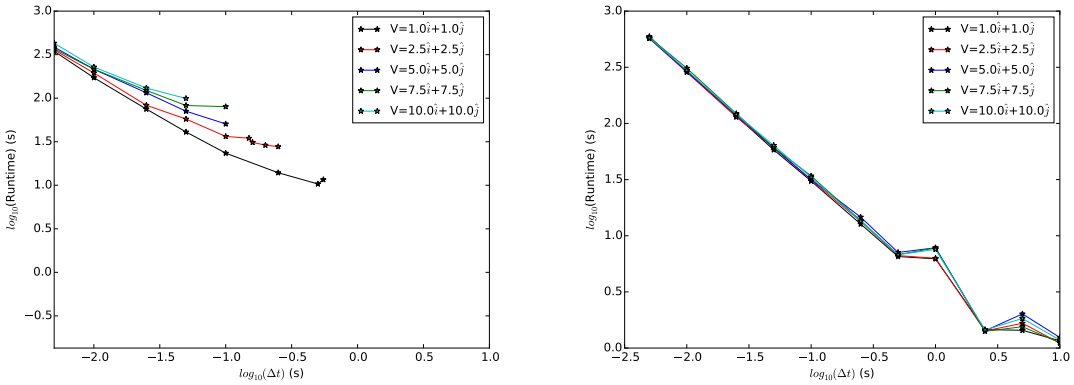


(a) Unstabilized **(b)** Stabilized

Figure 2. Solution with (a) no stabilization and (b) SUPG stabilization to $V du/dx - k d^2u/dx^2 = 0$ over the domain $0 \leq x \leq 1$ with boundary conditions $x(0) = 0, x(1) = 1$. The analytical solutions are shown as solid lines.

In higher dimensions, SUPG will not always give nodally-exact results due to the fact that velocities are often skew to the mesh and obtaining the representative element dimension is non-exact. However, all SUPG kernels developed displayed theoretical or better-than-theoretical convergence rates when measured in the L^2 norm. The details of this verification process are very lengthy, and a table indicating the kernel, the test name, and the convergence rates for all kernels is contained in Section 8.6. Two tests are still running, but based on the success of the 2-D tests, are very likely to give correct convergence rates once extended to 3-D.

With SUPG stabilization, two times as many residual and Jacobian evaluations are required, but stabilization is known to improve iterative solution convergence, so a runtime study was performed to determine how the runtime varies as a function of velocity both with and without stabilization. A test was created that solves the mass conservation equation in a transient case with an end time of 60 seconds. The test was run both with and without stabilization for a variety of time step sizes Δt and for several velocity profiles. As shown in Fig. 3, for the unstabilized cases, at very small Δt, have lower runtimes than the stabilized cases since there are a lower number of residual evaluations and the small time step makes those cases sufficiently stable that no appreciable difference is observed in runtime. However, as Δt is increased, the unstabilized cases begins to suffer from convergence problems, and at a sufficiently large Δt, the runtime actually begins to increase (even though there are a fewer number of total time steps) because the time instability limit is approached.



(a) Unstabilized, continuity **(b)** Stabilized, continuity

Figure 3. Difference in runtime between the (a) unstabilized and (b) stabilized mass conservation equation as a function of Δt for various velocity profiles for an end time of 60 seconds. Data for the unstabilized cases is cut-off once reaching a Δt beyond which larger Δt lead to longer run times.

By comparing the smallest possible runtime for the stabilized and unstabilized (defined as the runtime beyond which the runtime begins to increase again due to instabilities) cases, the ratio in the runtime without stabilization to with stabilization can be estimated. This is shown in Fig. 4 for the test of the mass conservation equation results given in Fig. 3 and for a similar test of the momentum equation. With stabilization, the runtime is decreased by anywhere between a factor of 10 and 80 for a runtime of 60 seconds. The larger the velocity magnitude, the more significant the stabilizing effect in the iterative solution, which is to be expected based on the de-stabilizing properties of convection. Typical runtimes for thermal-hydraulic transients can be of the order of hours or even days for certain transients in reactors with large amounts of thermal inertia, and so the runtime in those cases, provided a larger Δt than 60 seconds is acceptable to the user, the runtime could be decreased by a factor of 10-1000. Hence, not only is stabilization imperative in obtaining physically-meaningful results, but it also greatly reduces the runtime by decreasing the number of iterations in the iterative solution.

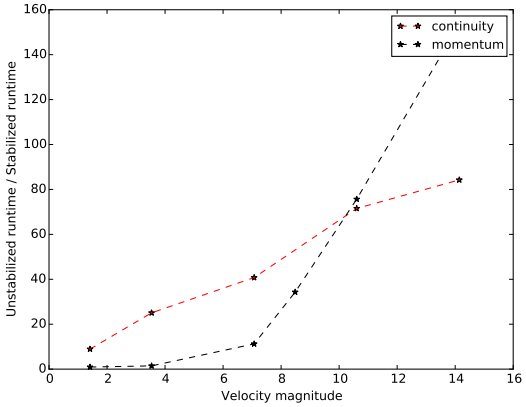


Figure 4. Ratio between the runtime without and with stabilization as a function of velocity magnitude for tests of the conservation of mass (continuity) equation and momentum equation for an end time of 60 seconds.

6 Conclusion

This report discussed the continuous and weak-form discretized compressible Euler equations and the theory behind SUPG stabilization. SUPG stabilization has been successfully implemented in PRONGHORN, and 1-D, 2-D, and 3-D verifications tests have been successfully performed, and nodally-exact results, along with theoretical convergence rates, verify that the method has been implemented correctly. The benefits of stabilization, with regard to obtaining physically-meaningful results with a significantly reduced number of iterations in the iterative solution scheme, have been demonstrated with a variety of tests. Future work will include the stabilization-like-implementation of a shock-capturing term to allow validation tests to be performed using shock-wave tests developed by the aerospace industry, the largest user of the Euler equations solved in PRONGHORN.

7 References

[1] Novak, A.J., Peterson, J.W., Andrs, D., and R.C. Martineau. "PRONGHORN Theory Manual." *Idaho National Laboratory.*

[2] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandie. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. *Nuclear Engineering and Design*, 239(10):1768-1778, 2009.

[3] Brooks, A.N. *A Petrov-Galerkin finite element formulation for convection dominated flows.* PhD thesis, California Institute of Technology, 1981.

[4] Fries, T.P. and H.G. Matthies. A Review of Petrov-Galerkin Stabilization Approaches and an Extension to Meshfree methods. Technical Report Informatikbericht-Nr. 2004-01, Technical University of Braunschweig, 2004.

[5] S. Morris. Advanced Fluid Mechanics. Technical report, University of California-Berkeley, 2016.

[6] Nield, D.A. and A. Bejan. *Convection in Porous Media.* Springer, 2013.

8 Appendix

8.1 Symbols and Notation

All notation is defined in this section, along with the equation numbers in which the symbol is either defined or first used. Precedence is given to the equation in which the symbol is defined. Symbols with units of "1" are unitless. Symbols that are multiply-defined should be inferred from the context. Material properties and common fluids terms are not referenced with an equation number, since their symbolic notation is well-known and their definition is not defined by the equations in which they are used (unless they are explicitly defined within this manual).

a_w	specific area	$1/m$	Eq. (8.48)
C_p	specific heat at constant pressure	$J/kg \cdot K$	Eq. (8.63)
C_ν	specific heat at constant volume	$J/kg \cdot K$	Eq. (8.64)
c_{ijkl}	Newtonian fluid proportionality tensor	—	Eq. (8.22)
$d(.)/dt$	material derivative of (.)	—	—
d_{pebble}	pebble diameter	m	—
D	hydraulic diameter	m	—
$\mathcal{D}$	Darcy coefficient	$kg/m^3 \cdot s$	Eq. (8.32)
e_{ij}	(symmetric) deformation tensor	$1/s$	Eq. (8.13)
$\tilde{e_{ij}}$	deviatoric strain tensor	$1/s$	Eq. (8.60)
e	internal energy per unit mass	J/kg	—
$\vec{e}_i$	vector aligned along axis i	1	—
E	total energy per unit mass	J/kg	Eq. (8.38)
$\vec{f}$	stress vector with components f_i	Pa	—
$\vec{f_s}$	resultant surface tension force vector with components $f_{s,i}$	N/m^2	—
$\mathcal{F}$	Forcheimer coefficient	$1/m$	Eq. (8.33)
$\vec{g}$	gravity vector with components g_i	m/s^2	—
h	enthalpy per unit mass	J/kg	Eq. (8.44)
h_e	finite element size	m	—
$\mathbf{I}$	identity matrix	1	—
k	thermal conductivity	$J/m \cdot s \cdot K$	—
L	a length	m	—
$\hat{n}$	unit outward normal with components n_i	—	—
P	pressure	Pa	—
Pe	Peclet number	1	—
S	a surface	m^2	—
s	specific entropy	$J/kg \cdot K$	—
$\vec{q}$	heat flux vector with components q_i	$J/m^2 \cdot s$	—
q	volumetric heat source	$J/m^3 \cdot s$	—

$\mathrm{Tr}(.)$	trace of (.)	—	—
$\mathbf{U}$	vector of PRONGHORN unknowns	several	—
$\vec{v}$	Darcy velocity with components v_i	m/s	Eq. (8.2)
$\vec{V}$	intrinsic velocity with components V_i	m/s	Eq. (8.2)
V	volume	m^3	—
$\vec{x}$	coordinate axes with components x_i	m	—
α	heat transfer coefficient	$J/m^2 \cdot s \cdot k$	Eq. (8.47)
β	expansivity	$1/K$	Eq. (8.66)
δ_{ij}	Kronecker delta	1	—
$\vec{\delta r}$	differential length with components δr_i	m	Eq. (8.11)
ϵ	porosity	1	Eq. (8.1)
ε	permutation	1	—
ξ_{ij}	(antisymmetric) deformation tensor	$1/s$	Eq. (8.13)
κ	bulk viscosity	$Pa \cdot s$	Eq. (8.27)
κ_s	effective solid thermal conductivity	$J/m \cdot s \cdot K$	Eq. (8.71)
λ_e	eigenvalue	1	—
λ	second Lamé parameter	$Pa \cdot s$	Eq. (8.24)
μ	dynamic viscosity	$Pa \cdot s$	Eq. (8.24)
$\vec{\eta}$	momentum	$kg/m^2 \cdot s$	—
σ	stress tensor with components σ_{ij}	Pa	Eq. (8.7)
θ	angle	1	—
ρ	density	kg/m^3	—
$\bar{\bar{\tau}}$	deviatoric stress tensor	Pa	Eq. (8.29)
ν	specific volume	m^3/kg	—
γ	total energy per unit mass	J/kg	—

8.1.1 Subscripts

f	fluid
s	solid
x, y, z	x, y, z components of Cartesian space
∞	infinite medium value
$\|(.)\|_i$	L^i norm of (.)

8.1.2 Superscripts

$(.)^T$	transpose

8.2 Governing Equations

This section presents the full derivation of the conservation of mass, momentum, and energy equations for a porous media, and explains the friction and form drag terms to account for porous media effects. Porous media are characterized by the porosity ϵ, or the ratio of void volume to total volume.

$$\epsilon \equiv \frac{\text{(connected) void volume}}{\text{total volume}} \tag{8.1}$$

Two different velocities can be defined for a porous media. $\vec{v}$ represents the fluid velocity averaged over the entire medium (over solid and fluid). This velocity is often referred to as the Darcy velocity. The alternative definition, $\vec{V}$, represents the fluid velocity averaged over only the fluid, and is referred to as the intrinsic phase velocity. These two representations of velocity are related to each other by the Dupuit-Forchiemer relationship:

$$\vec{v} = \epsilon \vec{V} \tag{8.2}$$

8.2.1 The Mass Equation

Because mass is conserved, the rate of change of the mass within an arbitrary volume $V(t)$ enclosing a system must be zero, because a system is defined such that its boundaries are closed to mass flow.

$$\frac{d}{dt} \int_{V(t)} \rho d\mathcal{V} = 0 \tag{8.3}$$

where $d(.)/dt$ represents the material derivative of (.). By the Reynolds Transport Theorem, the system perspective is converted to a control volume perspective:

$$\int_{V(t)} \frac{\partial \rho}{\partial t} d\mathcal{V} + \oint_{S(t)} \rho \vec{V} \cdot \hat{n} dS = 0 \tag{8.4}$$

where S is the surface bounding the system. By applying the divergence rule, the surface integral is converted to a volume integral. Then, because the selection of $V(t)$ is arbitrary, the integrand must be zero, giving the conservation of mass equation, often referred to as the "continuity equation:"

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0 \tag{8.5}$$

The porous media version of this equation simply replaces ρ by $\epsilon \rho_f$ to reflect the portion of the volume that is fluid:.

$$\frac{\partial (\epsilon \rho_f)}{\partial t} + \nabla \cdot (\epsilon \rho_f \vec{V}) = 0 \tag{8.6}$$

8.2.2 The Momentum Equation

The momentum equation is derived in a similar manner as the continuity equation, except that it is in this equation that particular porous media models are implemented. After deriving the Navier-Stokes equations for porous media, these porous media models will be individually discussed.

8.2.2.1 The Navier-Stokes Equations

The conservation of momentum equation is derived beginning from a general treatment of a balance of linear momentum for an arbitrary continuum. In order to relate the stress tensor σ_{ij} to the surface force experienced by the particle, the principle of local stress equilibrium is used. Applying the principle of local stress equilibrium to a generic shape in a frame with mutually orthogonal axes $\vec{e_i}$ will show that the surface force acting on the fluid particle is a linear, homogeneous function of the component of $\hat{n}$. The coefficient of proportionality, which is independent of the surface normal, is defined to be the stress tensor σ_{ij}.

$$f_i(\hat{n}) = f_i(\vec{e_j})n_j \rightarrow \sigma_{ij}n_j \tag{8.7}$$

A general balance of linear momentum for a system is:

$$\underbrace{\frac{d}{dt}\int_{V(t)} \rho V_i dV}_{\text{rate of change of momentum}} = \underbrace{\int_{V(t)} \rho g_i dV}_{\text{body force}} + \underbrace{\oint_{S(t)} \sigma_{ij} n_j dS}_{\text{surface force}} \tag{8.8}$$

where the surface integral is converted to a volume integral using the divergence theorem:

$$\frac{d}{dt}\int_{V(t)} \rho V_i dV = \int_{V(t)} \rho g_i dV + \int_{V(t)} \frac{\partial \sigma_{ij}}{\partial x_j} dV \tag{8.9}$$

It is assumed that the only body force acting on the fluid is gravity. In order to differentiate under the integral sign, the continuum is assumed to satisfy conservation of mass such that the time rate of change of ρV_i does not change. Then, the time differentiation can be moved inside the summation to act only on the velocity. Extending this argument to the continuous form in Eq. (8.9), and since Eq. (8.9) must apply for any arbitrary volume:

$$\rho \frac{dV_i}{dt} = \rho g_i + \frac{\partial \sigma_{ij}}{\partial x_j} \tag{8.10}$$

Eq. (8.10), often called the Cauchy equation, represents a balance of linear momentum for any mass-conserving continuum. In its most general form the stress tensor contains nine unknowns ($i = 1, 2, 3$ and $j = 1, 2, 3$). The stress tensor is symmetric if the only moments on the fluid particle are due to the resultant body and surface forces. It is rare that a fluid experience torques in excess of the body and surface force torques, and symmetry of the stress tensor is assumed in virtually all fluids codes, reducing nine unknowns to six.

The Navier-Stokes equations are derived by inserting a constitutive relation for σ_{ij} into the balance of momentum in Eq. (8.10). To understand this constitutive relation, the kinematics of deformation must be understood. The motion of a material element can be decomposed into three contributions:

1. Translation - due to the magnitude of the velocity $\vec{V}$
2. Rigid rotation (with angular velocity $\omega/2$) - due to the antisymmetric component ξ_{ij}
3. Stretching (with rate λ_e) along the eigenvectors of symmetric component e_{ij}

To understand the form of each of these components, consider two points in a fluid that are sufficiently close such that their separation $\vec{\delta r}$ can be approximated by a one-term Taylor series:

$$V_i(\vec{r} + \vec{\delta r}, t) - V_i(\vec{r}, t) = \frac{d}{dt}(\delta r_i) \approx \delta r_j \frac{\partial V_i}{\partial x_j} \tag{8.11}$$

Any tensor can be written as the sum of symmetric and antisymmetric parts:

$$\frac{\partial V_i}{\partial x_j} = \underbrace{\frac{1}{2}\left(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i}\right)}_{\text{symmetric}} + \underbrace{\frac{1}{2}\left(\frac{\partial V_i}{\partial x_j} - \frac{\partial V_j}{\partial x_i}\right)}_{\text{antisymmetric}} = e_{ij} + \xi_{ij} \tag{8.12}$$

where the symmetric and antisymmetric components are denoted as:

$$e_{ij} = \frac{1}{2}\left(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i}\right) \qquad \xi_{ij} = \frac{1}{2}\left(\frac{\partial V_i}{\partial x_j} - \frac{\partial V_j}{\partial x_i}\right) \tag{8.13}$$

e_{ij} is referred to as the "deformation tensor." To understand the antisymmetric component ξ_{ij}, assume for the time being that $e_{ij} = 0$. Combining Eqs. (8.11) and (8.13) and taking the scalar product with δr_i:

$$\delta r_i \frac{d}{dt}(\delta r_i) = \delta r_i \delta r_j \xi_{ij} \tag{8.14}$$

$\delta r_i \delta r_j$ is symmetric, while ξ_{ij} is antisymmetric. The product of any symmetric tensor with an asymmetric tensor must be zero, so the above reduces to:

$$\delta r_i \frac{d}{dt}(\delta r_i) \rightarrow \frac{1}{2}\frac{d}{dt}\left(\delta r_i\right)^2 = 0 \tag{8.15}$$

Hence, if only the antisymmetric component ξ_{ij} is present, because δr_i is constant in time, ξ_{ij} cannot represent stretching, since stretching would occur along δr_i. Because translation is accounted for simply by the magnitude of the velocity V_i, ξ_{ij} therefore represents a rigid rotation. ξ_{ij} does not deform the fluid, within the limit of applicability of the one term Taylor series assumed in Eq. (8.11), but rather produces a rigid rotation. To interpret the symmetric component of $\partial v_i/\partial x_j$, e_{ij}, consider for the time being that $\xi_{ij} = 0$. Then, Eq. (8.11) reduces to:

$$\frac{d}{dt}(\delta r_i) \approx \delta r_j e_{ij} \tag{8.16}$$

e_{ij} satisfies the eigenvalue problem:

$$\begin{aligned} e_{ij}\delta r_j &= \lambda_e \delta r_i \\ \left(e_{ij}\delta r_j - \lambda_e \delta r_i \delta_{ij}\right) &= 0 \end{aligned} \tag{8.17}$$

where λ_e are the eigenvalues and δr_j are the eigenvectors. Because e_{ij} is a 3 x 3 symmetric, real, matrix, there will be three unique, real, and mutually orthogonal eigenvectors δr_j. These eigenvectors are referred to as the "principal axes" of e_{ij}, and the eigenvalues as the "principal strain rates" of e_{ij}. Inserting Eq. (8.17) into Eq. (8.16), an interpretation of e_{ij} results for line elements that are parallel to one of the principal axes:

$$\frac{d}{dt}(\delta r_i) = \lambda_e \delta r_i \tag{8.18}$$

Because the relative velocity is along δr_i, e_{ij} represents stretching with rate λ_e in three mutually orthogonal directions (along the eigenvectors of e_{ij}. If a line element is parallel to one of these principal axes, then e_{ij} only contributes stretching. However, a line not parallel to one of the principal axes is both stretched and rotated due to e_{ij}, since the components of the line element stretch at different rates. From Eq. (8.13), the trace of the deformation tensor is equal to the divergence of the velocity.

$$\mathrm{Tr}(e_{ij}) = \nabla \cdot \vec{V} \tag{8.19}$$

As long as the volume is sufficiently small such that the divergence of the velocity can be taken as constant over the volume, the divergence of the velocity represents the fractional rate of increase of an infinitesimal volume.

$$\frac{1}{V}\frac{dV}{dt} \approx \nabla \cdot \vec{V} \tag{8.20}$$

Knowing that $\nabla \cdot \vec{V}$ represents a volume expansion, the motion at a point described by Eq. (8.11) can be separated into an isotropic radial expansion, a volume preserving motion obtained by subtracting off the volume change, and a rigid rotation. A factor of 1/3 appears to cancel the 3 that results from summation over i and j in δ_{ij}.

$$\begin{aligned} V_i(\vec{r}+\vec{\delta r},t) - V_i(\vec{r},t) &= \delta r_j \frac{\partial V_i}{\partial x_j} \\ &= \delta r_j e_{ij} - \frac{1}{2}\delta r_j \varepsilon_{ijk}\omega_k \\ &= \underbrace{\frac{1}{3}\delta r_i \nabla\cdot\vec{V}}_{\text{radial expansion}} + \underbrace{\delta r_j\left(e_{ij} - \frac{1}{3}\nabla\cdot\vec{V}\delta_{ij}\right)}_{\text{constant-volume motion}} - \underbrace{\frac{1}{2}\delta r_j \varepsilon_{ijk}\omega_k}_{\text{rigid rotation}} \end{aligned} \tag{8.21}$$

To formulate the Navier-Stokes equations requires a constitutive relation for the stress tensor σ_{ij}. The Navier-Stokes equations assume a Newtonian fluid constitutive relationship. A fluid is defined to be Newtonian if σ_{ij} is a linear, isotropic, function of the deformation tensor e_{ij}. Because ξ_{ij} causes a rigid rotation, but no deformation, there is no contribution in σ_{ij} due to ξ_{ij}. For simple fluids, the Newtonian approximation is an excellent assumption over a wide range of strain rates, but for molecules consisting of long chains of atoms, or with suspended solid particles, shearing the fluid breaks down an internal structure that can align the molecules such that the shear stress is not a linear function of the velocity gradient. For example, paint on a brush has high enough viscosity to not drip off the paint brush, but once sheared on a wall, viscous forces decrease such that it can be easily applied. The constitutive relationship for a Newtonian fluid is, in its most general form:

$$\sigma_{ij} = -P\delta_{ij} + c_{ijkl}e_{kl} \tag{8.22}$$

where P is the thermodynamic pressure determined from an equation of state and c_{ijkl} is a fourth-order tensor. The first term on the right-hand-side (RHS) reflects shear stresses due to pressure forces, a requirement to match hydrostatic observations. c_{ijkl} has 81 components that in general depend on pressure and temperature. In order to reduce 81 components to two unique parameters, the isotropic assumption is used. If the axes of a rectangular prism of fluid are aligned with the eigenvectors of e_{ij}, then from the discussion surrounding Eq. (8.18), the fluid element will not rotate, but will experience a stretching along the eigenvectors of e_{ij}. For an isotropic fluid, the stresses on the faces of that prism should be purely normal - if this were not so, then the fluid would have a preferred direction of deformation, a characteristic that is anisotropic. In order to show that selecting axes along the principal axes of e_{ij} for an isotropic fluid produces a diagonal e_{ij}, it must be required that when the principal axes of σ_{ij} coincide with the principal axes of e_{ij}, σ_{ij} is diagonal (no shear stresses).

This requirement gives $c_{12kl} = 0$ for all k and l. Physically, this means that the same stress rate in different coordinate frames does not give rise to different shear stresses. This reduces the 81 components of c_{ijkl} to nine. Isotropy also requires that $c_{1122} = c_{1133}$, since rotation of axes by 90° should not change the response in the fluid. σ_{11} then becomes:

$$\begin{aligned} \sigma_{11} &= -P + c_{1111}e_{11} + c_{1122}(e_{22} + e_{33}) \\ \sigma_{11} &= -P + (c_{1111} - c_{1122})e_{11} + c_{1122}(e_{11} + e_{22} + e_{33}) \end{aligned} \tag{8.23}$$

The dynamic viscosity μ and the parameter λ are material-specific parameters defined as:

$$\begin{aligned} 2\mu &= c_{1111} - c_{1122} \\ \lambda &= c_{1122} \end{aligned} \tag{8.24}$$

Expanding Eq. (8.23) to axes with arbitrary orientation, the constitutive relationship for a Newtonian fluid is:

$$\sigma_{ij} = -P\delta_{ij} + 2\mu e_{ij} + \lambda \nabla \cdot \vec{V}\delta_{ij} \tag{8.25}$$

Interpreting the significance of each of these terms is difficult unless Eq. (8.25) is reformulated in a manner similar to Eq. (8.21) by subtracting out the component associated with a volume change:

$$\sigma_{ij} = -P\delta_{ij} + \underbrace{2\mu\left(e_{ij} - \frac{1}{3}\nabla \cdot \vec{V}\delta_{ij}\right)}_{\text{deviatoric stress tensor}} + \underbrace{\left(\lambda + \frac{2\mu}{3}\right)\nabla \cdot \vec{V}\delta_{ij}}_{\text{volume dissipative term}} \tag{8.26}$$

where the bulk, or volume, viscosity κ can be introduced:

$$\kappa = \lambda + \frac{2\mu}{3} \tag{8.27}$$

The deviatoric stress term represents volume-preserving deformation. The dissipative term, which represents viscous dissipation effects due to volume changes, is only observed if $\nabla \cdot \vec{V} \neq 0$. This term is often negligible, unless $\nabla \cdot \vec{V}$ changes slowly over a long period of time (attenuation) or if $\nabla \cdot \vec{V}$ is large, such as in shock waves. However, in situations with nonzero $\nabla \cdot \vec{V}$, the physics are often so different that an entirely different constitutive equation should be used. Substituting Eq. (8.26) into Eq. (8.10) equation gives the Navier-Stokes equations, where Eq. (8.13) is used to insert the definition of e_{ij} and Eq. (8.27) for the definition of κ:

$$\begin{aligned} \rho\frac{dV_i}{dt} &= \rho g_i + \frac{\partial}{\partial x_j}\left\{-P\delta_{ij} + 2\mu\left[\frac{1}{2}\left(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i}\right) - \frac{1}{3}\nabla \cdot \vec{V}\delta_{ij}\right] + \kappa\nabla \cdot \vec{V}\delta_{ij}\right\} \\ \rho\frac{dV_i}{dt} &= \rho g_i - \nabla P + \nabla \cdot \left[\mu\left(\nabla V_i + (\nabla V_i)^T\right) - \frac{2\mu}{3}\nabla \cdot \vec{V}\mathbf{I}\right] + \nabla \cdot \left(\kappa\nabla \cdot \vec{V}\right) \end{aligned} \tag{8.28}$$

The deviatoric stress tensor introduced in Eq. (8.26) is:

$$\bar{\bar{\tau}} \equiv 2\mu\left(e_{ij} - \frac{1}{3}\nabla \cdot \vec{V}\delta_{ij}\right) = \mu\left(\nabla V_i + (\nabla V_i)^T\right) - \frac{2\mu}{3}\nabla \cdot \vec{V}\mathbf{I} \tag{8.29}$$

Using this notation, and neglecting the volume dissipative effect captured by κ, the non-porous form of the Navier-Stokes equations used in PRONGHORN is:

$$\frac{\partial(\rho\vec{V})}{\partial t} + \nabla \cdot (\rho\vec{V}\vec{V}) = -\nabla P + \nabla \cdot \bar{\bar{\tau}} + \rho\vec{g} \tag{8.30}$$

where conservation of mass has been used to express the left-hand-side (LHS) in conservative form. The

volume dissipative term with the κ coefficient is commonly neglected by setting $\lambda = -2\mu/3$ ([**?**], page 144). From here forward, κ is set to zero in the momentum equation based on the argument that $\lambda = -2\mu/3$. The porous-media form of Eq. (8.30) is obtained by replacing ρ by $\epsilon\rho_f$ to reflect the fraction of medium that is fluid. In addition, the viscous stress is also scaled by ϵ to reflect the fraction of the medium that is subject to viscous forces. This means that $\nabla \cdot \tau \rightarrow \nabla \cdot (\epsilon\tau)$, which when expanded in terms of pressure and the deviatoric stress, translates to $-\nabla(\epsilon P) + \nabla \cdot (\epsilon\bar{\bar{\tau}})$ [**?**].

$$\frac{\partial(\epsilon\rho_f \vec{V})}{\partial t} + \nabla \cdot (\epsilon\rho_f \vec{V}\vec{V}) + \nabla(\epsilon P) - \nabla \cdot (\epsilon\bar{\bar{\tau}}) - \epsilon\rho_f \vec{g} = 0 \tag{8.31}$$

Porous media models essentially then add an additional momentum sink in the momentum equations. These terms are described next. Some codes retain the viscous loss term $\nabla \cdot \epsilon\bar{\bar{\tau}}$, though this term is dropped in PRONGHORN because viscous effects are negligible in reactor applications.

8.2.2.2 The Darcy Equation

Darcy's law relates the Darcy velocity from Eq. (8.2) to the pressure drop through a relationship determined experimentally by Henry Darcy during his studies of the water supply to Dijon, France. It has since been derived from the Stokes equation, a simplification of the Navier-Stokes equation for stationary, creeping, and incompressible flow. The non-constant porosity form of Darcy's law is:

$$\nabla(\epsilon P) = -\mathcal{D}\mu\vec{v} \tag{8.32}$$

where $\mathcal{D}$ will be referred to as the "Darcy coefficient." Values for this coefficient are given in Section **??**. This relationship is linear in velocity, and hence represents friction drag.

8.2.2.3 The Forcheimer Equation

In the range $1 \leq Re_D \leq 10$, nonlinear drag effects begin to become significant. These nonlinear drag effects scale as the square of velocity, and hence represent form drag due to movement in a field of solid obstacles. Use of Darcy's law, a linear drag effect, gives inaccurate results at higher Re unless another term is included. The Forcheimer equation is a modification to a concept developed by Dupuit that adds a quadratic drag term to the conventional form of Darcy's law. The total drag on the fluid is taken as the sum of the viscous drag (Darcy term) and the form drag (Forcheimer term), an approach first suggest by Reynolds [**?**]. While most forms assume constant porosity, the non-constant-porosity form is:

$$\nabla(\epsilon P) = -\mathcal{D}\mu\vec{v} - \mathcal{F}\rho_f|\vec{v}|\vec{v} \tag{8.33}$$

where $\mathcal{F}$ will be referred to as the "Forcheimer coefficient." Values for the Forcheimer and Darcy coefficients are provided by KTA empirically as:

$$\begin{aligned} \mathcal{D}_{KTA} &= \frac{160(1-\epsilon)^2}{d_{\text{pebble}}^2\epsilon^2} \\ \mathcal{F}_{KTA} &= 3\left(\frac{1-\epsilon}{Re}\right)^{0.1}\frac{(1-\epsilon)}{d_{\text{pebble}}\epsilon^2} \end{aligned} \tag{8.34}$$

8.2.2.4 The Momentum Equation

This section combines the discussion in Section 8.2.2.1 with the porous media models discussed in Sections 8.2.2.2 and 8.2.2.3 to give the form of the momentum equation used in PRONGHORN:

$$\frac{\partial(\epsilon\rho_f\vec{V})}{\partial t} + \nabla\cdot(\epsilon\rho_f\vec{V}\vec{V}) + \nabla(\epsilon P) - \nabla\cdot(\epsilon\bar{\bar{\tau}}) - \epsilon\rho_f\vec{g} + \mathcal{D}\mu\vec{v} + \mathcal{F}\rho_f|\vec{v}|\vec{v} = 0 \tag{8.35}$$

Transforming all $\vec{v}$ to $\vec{V}$ using Eq. (8.2):

$$\frac{\partial(\epsilon\rho_f\vec{V})}{\partial t} + \nabla\cdot(\epsilon\rho_f\vec{V}\vec{V}) + \nabla(\epsilon P) - \nabla\cdot(\epsilon\bar{\bar{\tau}}) - \epsilon\rho_f\vec{g} + \mathcal{D}\mu\epsilon\vec{V} + \mathcal{F}\epsilon^2\rho_f|\vec{V}|\vec{V} = 0 \tag{8.36}$$

Finally, PRONGHORN neglects viscous shear forces, where Section **??** provides justification. Then, the momentum equation in PRONGHORN becomes:

$$\frac{\partial(\epsilon\rho_f\vec{V})}{\partial t} + \nabla\cdot(\epsilon\rho_f\vec{V}\vec{V}) + \nabla(\epsilon P) - \epsilon\rho_f\vec{g} + \mathcal{D}\mu\epsilon\vec{V} + \mathcal{F}\epsilon^2\rho_f|\vec{V}|\vec{V} = 0 \tag{8.37}$$

8.2.3 The Solid and Fluid Energy Equations

Finally, an energy equation is needed because the equations of state for pressure are in terms of two variables - density and total energy. The first law of thermodynamics relates the rate of change of energy in a volume to the rate of energy and heat addition. The total energy is the sum of internal energy and kinetic energy. e, the internal energy per unit mass, represents the potential energy due to bond stretching plus the mean kinetic energy of molecules when moving in a frame of reference moving with the center of mass velocity (vibratory motion). The total energy E per unit mass is the sum of this internal energy and kinetic energy of the body, per unit mass.

$$E = e + \frac{1}{2}|\vec{V}|^2 \tag{8.38}$$

Conservation of this total energy requires a balance between the power supplied by the contact and body forces and the heat addition. This balance is obtained by recognizing that all the forces present in the general momentum balance in Eq. (8.9) should be multiplied by velocity, since power equals force times velocity.

$$\underbrace{\frac{d}{dt}\int_{V(t)}\rho E\,dV}_{\text{rate of change of total energy}} = \underbrace{\int_{V(t)}\rho g_i V_i\,dV}_{\text{power from gravity forces}} + \underbrace{\oint_{S(t)} V_i\sigma_{ij}n_j dS}_{\text{power from viscous forces}} - \underbrace{\oint_{S(t)} q_i n_i dS}_{\text{heat addition}} \tag{8.39}$$

where $\vec{q}$ is the heat flux vector representing heat flow out of the fluid particle. The divergence theorem is applied to convert surface integrals to volume integrals.

$$\frac{d}{dt}\int_{V(t)}\rho E\,dV = \int_{V(t)}\rho g_i V_i\,dV + \int_{V(t)}\frac{\partial}{\partial x_j}\left(V_i\sigma_{ij}\right)dV - \int_{V(t)}\frac{\partial}{\partial x_j}\left(q_j\right)dV \tag{8.40}$$

Assuming conservation of mass, and by recognizing that the choice of the volume is arbitrary, Eq. (8.40) can be written as:

$$\rho\frac{dE}{dt} = \rho\vec{g}\cdot\vec{V} + \frac{\partial}{\partial x_j}\left(V_i\sigma_{ij}\right) - \nabla\cdot\vec{q} = \rho\vec{g}\cdot\vec{V} + \underbrace{V_i\frac{\partial\sigma_{ij}}{\partial x_j}}_{\text{contact power}} + \underbrace{\sigma_{ij}\frac{\partial V_i}{\partial x_j}}_{\text{deformation power}} - \nabla\cdot\vec{q} \tag{8.41}$$

where suffix notation has been replaced by vector notation in all but one term. By inserting the constitutive relation for σ_{ij} for a Newtonian fluid from Eq. (8.26) into the total energy equation in Eq. (8.41), after some manipulation, a form useful for implementation is obtained:

$$\rho\frac{\partial E}{\partial t} + \rho\vec{V}\cdot\nabla E = \rho\vec{g}\cdot\vec{V} - \nabla\cdot(P\vec{V}) + \nabla\cdot(\bar{\bar{\tau}}\vec{V}) + \nabla\cdot(\vec{V}\kappa\nabla\cdot\vec{V}) - \nabla\cdot\vec{q} \tag{8.42}$$

Eq. (8.42) can be cast into conservative form by reversing the chain rule:

$$\frac{\partial(\rho E)}{\partial t} + \nabla\cdot((\rho E + P)\vec{V}) = \rho\vec{g}\cdot\vec{V} + \nabla\cdot(\bar{\bar{\tau}}\vec{V}) + \nabla\cdot(\vec{V}\kappa\nabla\cdot\vec{V}) - \nabla\cdot\vec{q} \tag{8.43}$$

The enthalpy per unit mass h is defined as:

$$\rho h = \rho E + P \tag{8.44}$$

Inserting this definition into Eq. (8.43), and neglecting viscous heating effects ($\nabla\cdot(\bar{\bar{\tau}}\vec{V})$) and setting $\kappa = 0$ according to the discussion surrounding Eq. (8.30), the conservative, non-porous form of the energy equation used in PRONGHORN becomes:

$$\frac{\partial(\rho E)}{\partial t} + \nabla\cdot(\rho h\vec{V}) = \rho\vec{g}\cdot\vec{V} - \nabla\cdot\vec{q} \tag{8.45}$$

Assuming that $\vec{q} = -k\nabla T$ represents the conduction heating, where k is the thermal conductivity, and including body heating q and convection heating $\alpha(T - T_\infty)$ with a heat transfer coefficient α, the non-porous form of the energy equation becomes:

$$\frac{\partial(\rho E)}{\partial t} + \nabla\cdot(\rho h\vec{V}) - \rho\vec{g}\cdot\vec{V} - \nabla\cdot(k\nabla T) + \alpha(T - T_\infty) - q = 0 \tag{8.46}$$

To obtain the correct units in Eq. (8.46), α represents the traditional heat transfer coefficient h_{HTC} multiplied by a_w:

$$\alpha = a_w h_{HTC} \tag{8.47}$$

where a_w represents the ratio of the wetted area through which heat transfer can occur in a length of δx per unit volume of length δx, in the limit of δx going to zero.

$$a_w = \lim_{\delta x\to 0}\frac{\text{wetted heat transfer area of length } \delta x}{\text{volume of length } \delta x} \tag{8.48}$$

The heat transfer coefficient can then be calculated directly from correlations in terms of Nu.

$$\alpha = a_w\frac{k_f Nu}{d_{pebble}} \tag{8.49}$$

With these definitions, the porous media form of Eq. (8.46) for the fluid is obtained by replacing each occurrence of ρ_f by $\epsilon\rho_f$ to reflect the portion of the volume that is fluid. Assuming that the porosity, which is technically defined based on the fraction of volume that is fluid , can be extended to represent the fraction of cross-sectional area that is fluid, the conduction term is modified by ϵ to reflect the heat transfer in the fluid phase. This assumes that the core is isotropic and all voids are connected, a condition expected in reactor applications. With these modifications, the fluid energy equation used in PRONGHORN becomes:

$$\frac{\partial(\epsilon\rho_f E)}{\partial t} + \nabla\cdot(\epsilon h_f \rho_f \vec{V}) - \nabla\cdot(\epsilon k_f \nabla T) - \epsilon\rho_f \vec{g}\cdot\vec{V} + \alpha(T_f - T_s) - q_f = 0 \tag{8.50}$$

Note that the convection heat flux is simply added on as a term, but not in the form $\nabla\cdot\vec{q}$, which would translate to $\nabla\cdot(\alpha(T_s - T_f))$. In non-porous media, the convective heat transfer appears in the boundary conditions, but here due to the homogeneous nature of the porous media assumption, this term appears in the governing equation itself.

To obtain the solid energy equation, one could simply set $\vec{V} = 0$ in Eq. (8.46) and perform the appropriate porous media modifications. However, equations of state for solid material properties as a function of energy are much more difficult to encounter in the literature than correlations in terms of temperature. Hence, to derive the solid energy equation, equilibrium thermodynamics is used to relate total energy to entropy, and then entropy to temperature to transform Eq. (8.41) to an equation entirely for temperature. Then, $\vec{V} = 0$ will be used to reduce the equation to an equation for the solid temperature.

In order to obtain the entropy equation, the mechanical energy component of Eq. (8.41) is subtracted out by first forming the mechanical energy equation by taking the scalar product of the Cauchy momentum equation in Eq. (8.10) with velocity (since power equals force multiplied by velocity).

$$V_i\rho\frac{dV_i}{dt} \rightarrow \rho\frac{d}{dt}\left(\frac{1}{2}V_i^2\right) = \rho g_i V_i + V_i\frac{\partial\sigma_{ij}}{\partial x_j} \tag{8.51}$$

Note that $\vec{q}$ does not appear, since the heat flux increases internal energy, not kinetic energy. Recognizing that $V_i\partial\sigma_{ij}/\partial x_j$ appears in the total energy equation in Eq. (8.41), a balance of internal energy equation is obtained by subtracting Eq. (8.51) from Eq. (8.41):

$$\rho\frac{de}{dt} = \sigma_{ij}\frac{\partial V_i}{\partial x_j} - \nabla\cdot\vec{q} \tag{8.52}$$

In order to obtain equations for temperature and entropy, relations from equilibrium thermodynamics are needed. But, from the form of Eq. (8.41), the dissipative terms on the RHS do not represent equilibrium conditions at all - hence, *local* thermodynamic equilibrium must be assumed. While in a global sense the system is not in thermodynamic equilibrium, each fluid particle is assumed to be in equilibrium. This assumption is valid if a fluid particle reaches equilibrium with its surroundings quickly. The time scale on which a fluid particle reaches equilibrium can usually be cast as some multiple of the collision frequency, and provided that this time scale is much smaller than the time scale characterizing the problem, local thermodynamic equilibrium can be assumed. Local equilibrium is attained when local changes in pressure and temperature are relatively small with respect to the far-field values of pressure and temperature. Assuming local thermodynamic equilibrium, changes in energy and entropy can be related to other state variables.

Any system always has at least two modes of energy transfer - work and heat. The classification of a system depends on the number of available energy transfer modes. Simple systems only experience pressure-volume work, and are not subject to work by electromagnetic fields or other means. Hence, there are two ways by which to change the energy of a simple system - by pressure-volume work and by heat transfer. One thermodynamic property is needed to fix the state of a system for each mode of energy transfer, so the state of a simple, pure system can be fixed with two properties. From the first law of thermodynamics, for a simple, pure system:

$$de = dq + dw \tag{8.53}$$

where q represents heat addition and w represents work done on the body. For an internally reversible process, $dq = Tds$, where s is the entropy of the fluid. For a pure fluid, the only mode of energy transfer besides heat addition is pressure-volume work, so $dw = -Pd\nu$, where ν is the specific volume. While no fluid is truly pure, a fluid can be approximated as pure as long as the components are well-mixed, there is no significant reaction between the components, and the components do not break apart in the flow. Inserting these definitions into Eq. (8.53), the Gibbs identity is obtained:

$$de = Tds - Pd\nu \tag{8.54}$$

Because local thermodynamic equilibrium is assumed, the Gibbs identity also applies for irreversible processes. The Gibbs identity is an exact differential, meaning that any form of a differential in e, s, and ν produce the form above. Using Eq. (8.54), the internal energy equation in Eq. (8.52) can be formulated into an equation for entropy conservation.

$$\rho\left(T\frac{ds}{dt} - P\frac{d\nu}{dt}\right) = \sigma_{ij}\frac{\partial V_i}{\partial x_j} - \nabla\cdot\vec{q} \tag{8.55}$$

The second term on the LHS can be related to the velocity using the continuity equation, Eq. (8.5).

$$\frac{d}{dt}\left(\rho^{-1}\right) \rightarrow \frac{-1}{\rho^2}\frac{d\rho}{dt} \rightarrow \frac{\nabla\cdot\vec{V}}{\rho} \tag{8.56}$$

Inserting this into Eq. (8.55), and using the constitutive relation for a Newtonian fluid in Eq. (8.25):

$$\begin{aligned} \rho T\frac{ds}{dt} - P\nabla\cdot\vec{V} &= \left(-P\delta_{ij} + 2\mu e_{ij} + \lambda\nabla\cdot\vec{V}\delta_{ij}\right)\frac{\partial V_i}{\partial x_j} - \nabla\cdot\vec{q} \\ \rho T\frac{ds}{dt} &= 2\mu e_{ij}\frac{\partial V_i}{\partial x_j} + \lambda(\nabla\cdot\vec{V})^2 - \nabla\cdot\vec{q} \end{aligned} \tag{8.57}$$

Using the definition of $\partial V_i/\partial x_j$ in Eq. (8.13):

$$\rho T\frac{ds}{dt} = 2\mu e_{ij}(e_{ij} + \xi_{ij}) + \lambda(\nabla\cdot\vec{V})^2 - \nabla\cdot\vec{q} \tag{8.58}$$

By the property of the multiplication of the symmetric tensor e_{ij} with the antisymmetric tensor ξ_{ij}, by Eq. (**??**), Eq. (8.58) simplifies to an equation representing conservation of entropy:

$$\rho T\frac{ds}{dt} = 2\mu e_{ij}^2 + \lambda(\nabla\cdot\vec{V})^2 - \nabla\cdot\vec{q} \tag{8.59}$$

The entropy equation can equivalently be written in terms of the derivatoric stress tensor $\bar{\bar{\tau}}$ defined in Eq. (8.29). Defining:

$$\tilde{e}_{ij} \equiv e_{ij} - \frac{1}{3}\nabla\cdot\vec{V}\delta_{ij} \equiv \frac{\bar{\bar{\tau}}}{2\mu} \tag{8.60}$$

and substituting this into Eq. (8.59) gives an equivalent form of the entropy equation.

$$\rho T\frac{ds}{dt} = 2\mu\tilde{e}_{ij}^2 + \kappa(\nabla\cdot\vec{V})^2 - \nabla\cdot\vec{q} \tag{8.61}$$

To obtain an equation for temperature that will be used for the solid energy equation, equilibrium thermodynamics requires a relationship between entropy, temperature, and one other state variable. Pressure is

selected as this state variable because it is directly related to the boundary conditions and forces. A choice such as ρ would complicate the equation, since it itself is a function of the other state variable, temperature. With these choices, the derivative of entropy is given by the chain rule as:

$$\frac{ds}{dt} = \left(\frac{\partial s}{\partial T}\right)_P \frac{dT}{dt} + \left(\frac{\partial s}{\partial P}\right)_T \frac{dP}{dt} \tag{8.62}$$

Specific heats are defined according to the amount of heat required to increase temperature when heat is added reversibly:

$$C_p \equiv T\left(\frac{\partial s}{\partial T}\right)_P \tag{8.63}$$

$$C_\nu \equiv T\left(\frac{\partial s}{\partial T}\right)_\nu \tag{8.64}$$

Hence, the coefficient on the first term on the RHS of Eq. (8.62) is C_p/T. The second coefficient can be determined by manipulating the Gibbs identity in Eq. (8.54). This manipulation gives a thermodynamic relationship known as the Maxwell identity:

$$-\left(\frac{\partial s}{\partial P}\right)_T = \left(\frac{\partial \nu}{\partial T}\right)_P \tag{8.65}$$

The expansivity, or volumetric coefficient of thermal expansion, is defined as:

$$\beta \equiv \frac{1}{\nu}\left(\frac{\partial \nu}{\partial T}\right)_P \tag{8.66}$$

Combining Eqs. (8.66), (8.65), (8.63), and (8.62) gives the following relationship between entropy and other state variables that will allow the entropy equation to be transformed to an equation entirely for temperature.

$$\frac{ds}{dt} = \frac{C_p}{T}\frac{dT}{dt} - \beta\nu\frac{dP}{dt} \tag{8.67}$$

Substituting Eq. (8.67) into Eq. (8.61) gives the temperature equation:

$$\underbrace{\rho C_p \frac{dT}{dt}}_{\text{stored energy}} - \underbrace{T\beta\frac{dP}{dt}}_{\text{compression work}} = \underbrace{2\mu\tilde{e}_{ij}^2}_{\text{viscous heat}} + \underbrace{\kappa(\nabla\cdot\vec{V})^2}_{\text{volume viscous heat}} - \underbrace{\nabla\cdot\vec{q}}_{\text{heat conduction}} \tag{8.68}$$

Like for the fluid, viscous heating and volumetric heating effects are assumed to be negligible (which is especially true for the solid because its velocity is assumed zero). In addition, compression work is also negligible for a solid. Therefore, the non-porous media form of the solid energy equation simplifies to:

$$\rho C_p \frac{\partial T}{\partial t} = -\nabla\cdot\vec{q} \tag{8.69}$$

To obtain the porous-media form, multiply each occurrence of ρ by $1-\epsilon$ to reflect the portion of the volume that is solid. Assuming that heat transfer occurs by conduction within the solid and convection at the surface, while allowing a body heat source, Eq. (8.69) becomes:

$$(1-\epsilon)\rho_s C_{p,s}\frac{\partial T_s}{\partial t} - \nabla\cdot(\kappa_s\nabla T_s) + \alpha(T_s - T_f) - q_s = 0 \tag{8.70}$$

The material thermal conductivity k_s that would appear in the conduction term is replaced by κ_s, the effective

thermal conductivity of the solid. Conduction depends not only on the temperature of the solid, but also on the size and shape of the particles, since heat can be conducted from one discrete particle to another, in addition to within a single particle itself. κ_s captures the combined effects of radiation, conduction in the fluid and solid, and convection in the fluid.

$$\kappa_s = \kappa_{\text{radiation}} + \kappa_{\text{fluid conduction}} + \kappa_{\text{solid conduction}} + \kappa_{\text{fluid convection}} \tag{8.71}$$

8.3 The Finite Element Method

This section provides a high-level overview of the FEM that is used to solve the discretized forms of the governing equations in PRONGHORN. The finite element method seeks a solution to a governing differential equation by assuming the solution u can be expressed as a summation of coefficients C and shape functions ϕ. The solution in a particular element, u^h, is the summation of the expansion coefficients in that element, multiplied by the shape functions over that element.

$$u^h = \sum_{j=1}^{n_{nodes}} C_j \phi_j \tag{8.72}$$

Any differential equation can in general be expressed as $A(u) = f$, where A is an operator that acts on the true solution u, and f is a function that does not contain u. For example, for the continuity equation solved in PRONGHORN, this operator is:

$$A() = \left(\frac{\partial()}{\partial t} + \vec{V} \cdot \nabla() + () \cdot \nabla \vec{V} \right) \tag{8.73}$$

and because there are no sources of mass due to phase change, or other possible means, $f = 0$. To keep the discussion of the numerical method general, all equations will simply be discussed in the form $A(u) = f$, where the particular forms of A and f can be determined from knowledge of the governing equations. The approximation in Eq. (8.72) in most cases will not be the true solution, and hence if u^h is substituted back into the governing equation, the equation will not be strictly satisfied. Rather, if $A(u) - f = 0$, then $A(u^h) - f \neq 0$. The "leftover" amount is the residual r^N:

$$r^N = A(u^h) - f \tag{8.74}$$

The objective of the FEM is to minimize the residual with respect to a particular norm, since a zero residual means that the exact solution has been determined, so long as there exists a unique solution. A weighted residual method is, in general, a requirement that the integral of the residual over the phase space, multiplied by some weighting function ψ, be zero:

$$\int_\Omega r^N(\Omega)\psi(\Omega)d\Omega = 0 \tag{8.75}$$

The FEM is a weighted residual method, with the most common form of the FEM being the Galerkin method. The Galerkin method chooses the optimal weight functions ψ in order to most closely minimize the error $e^N \equiv u - u^N$. The functions used to construct u^N in Eq. (8.72) "live" in a particular function space, and the true solution may or may not exist within that space. However, for the error to be minimized, e^N should be orthogonal to u^N, since that will minimize the "distance" between u^N and the true solution. This requires:

$$u^N \cdot e^N = 0 \tag{8.76}$$

However, the complication with this formulation is that the error is never known (unless solving a problem for which the analytical solution is known), and hence the above requirement does not provide a useful condition. The best approximation to the error can be given by the residual r^N. Hence, Galerkin's method seeks to minimize the residual by requiring that is be orthogonal to the approximate solution u^N:

$$u^N \cdot r^N = 0 \tag{8.77}$$

While only in special cases are the error and residual the same, Galerkin's method very effectively obtains an approximate solution, which is why the method is the foundation of the FEM. Galerkin's method technically only gives one equation, while we have N unknowns. However, an even stricter requirement that the above be true for every C_j in the expansion for u^N provides the needed N equations for N unknowns:

$$\int_\Omega C_j \phi_j \left(A(C_j \phi_j) - f \right) d\Omega = 0 \quad \text{for } j = 1, 2, 3, \cdots, N \tag{8.78}$$

What further differentiates the FEM from other weighted residual methods is the systematic selection of the shape functions ϕ and the formulation of the weak form of the differential equation to reduce differentiability requirements.

8.3.1 The Weak Form

Many solutions to physical problems are not continuously differentiable, and hence the higher-order derivatives in the operator A, when integrated, may give infinite or singular values, and hence make application of the Galerkin method impossible. However, by constructing a weak form, the differentiability requirements on the solution are lowered by transferring some of the differentiation to the weight function.

By transferring some differentiation to the weight function, then the differentiability requirements of the approximate solution are lowered, permitting the use of lower-order shape functions (a significant cost savings). In addition, the weak form is actually more accurate than the strong form because it doesn't assume anything about differentiability of material properties or the solution (more than what is found in the weak form itself).

The weak form is identical to the strong form of the governing equations provided the true solution is sufficiently smooth to satisfy the strong form, and hence the weak form is still an exact representation of the governing physics. The problem statement for a FEM solution is:

$$\int_\Omega r^N(\Omega) \psi(\Omega) d\Omega = 0 \quad \forall \psi \in \text{space of admissible functions} \tag{8.79}$$

If this weighted residual statement is required for any arbitrary weight function, then it must be true that the integrand, $r^N(\Omega)$ equals zero. While the "for all" symbol appears in Eq. (8.79), not any arbitrary shape function can be selected. The considerations that define the space of admissible weight functions are discussed next.

8.3.2 Shape Functions

In principle, any set of functions can be used as the shape functions ϕ, provided that the integrals in the weak form remain finite. One way to ensure that the integrals remain finite is to require the shape functions to "live" within a particular function space. The Hilbert Sobolev space is defined as $H^l(\Omega)$, where l refers to the highest derivative of the function that exists. For instance, a square wave lives in $H^0(\Omega)$, since its first derivative is undefined at locations where the wave height changes. A piecewise function lives in $H^1(\Omega)$, since first derivatives are everywhere defined, but jumps in the second derivatives lead to infinite integrals when measured in the H^2 norm. The Hilbert-space norm is defined in 1-D as:

$$\|u\|_{H^I(\Omega)} \equiv \left[\sum_{j=0}^{I} \int_\Omega \left(\frac{d^j u}{dx^j}\right)^2 d\Omega\right]^{1/I} \tag{8.80}$$

Hence, a function is in $H^I(\Omega)$ if $\|u\|_{H^I(\Omega)} < \infty$. The weight functions in the Galerkin method are equivalent to the shape functions, and hence both must be in the same (or higher) Hilbert space required as a minimum in the weak form. For other types of problems, such as in electricity and magnetism where curls appear, the space of admissible test functions may be different from the Hilbert space.

The shape functions are also *chosen* to be zero on the Dirichlet boundaries. In essence, the expansion in Eq. (8.72) in its most explicit form reads:

$$u^N(\Omega) = \phi_0 + \sum_{j=1}^{N} C_j \phi_j \tag{8.81}$$

where ϕ_0 is a constant whose only purpose is to satisfy the homogeneous form of the essential (Dirichlet) boundary conditions. This choice is very convenient for the weak form approach, since then the term with displacements evaluated on the boundary drops out of the weak form because the shape functions ϕ_j are always zero on the boundary.

With the restriction on the function space from which the shape functions can derive, there are still infinitely many choices for shape functions, but some classes of functions have special properties that makes their use in the finite element method nearly universal. The most common choice for shape functions are the Lagrange shape functions, which are polynomials that obtain values of 1 at the node to which they correspond and zero at all other nodes. The fact that these shape functions are zero at all nodes except 1 is extremely advantageous. If ϕ are defined such that they are nonzero over only a small portion of the entire domain, then the matrices that result are sparse. Sparse matrices are much easier to solve.

8.3.3 Quadrature

Quadrature rules are used for approximating the integrals that appear in the residual statements for the purpose of reducing the high computational cost associated with analytic integration and allowing evaluation of integrands that may not even have analytic integrals. The appropriate quadrature rule is automatically selected by libMesh.

8.3.4 Element Mappings

In order to utilize quadrature rules, all integrals must be transformed to the domain over which the quadrature rule is defined. For Gaussian quadrature, this domain is, in 1-D, from $-1 \leq \xi \leq 1$, where ξ is the coordinate

in the "master element," and x is the coordinate in the physical finite element. This mapping from a global element perspective to a master element allows the use of Gaussian quadrature. Any arbitrary integral in x is transformed to an integral in ξ by multiplying by the Jacobian of the transformation, $\mathcal{J}$:

$$\int_a^b f(x)dx \quad \rightarrow \quad \int_{-1}^{1} f(x(\xi))\mathcal{J}d\xi \tag{8.82}$$

where the Jacobian of the transformation is, in 1-D:

$$\mathcal{J} \equiv \frac{dx}{d\xi} \tag{8.83}$$

The mapping $x(\xi)$ can take a variety of forms, but is generally selected to be an isoparametric mapping defined by:

$$x(\xi) = \sum_{i=1}^{n_{en}} X_i \phi_i(\xi) \tag{8.84}$$

where X_i are the physical nodes and ϕ the shape functions in the master element. All calculations are performed element-by-element, and after completion, are assembled into a global matrix that is then solved using the JFNK method.

8.4 Numerical Method

The nonlinear system solved by MOOSE is stated such that each of the weighted residuals $R(\vec{x})$ constructed under the finite element formulation equal zero. This essentially requires that the governing equations are satisfied in an averaged sense. For a single residual statement, this is represented symbolically as:

$$R(\vec{x}) = 0 \tag{8.85}$$

Eq. (8.85) is nothing more than a short-hand statement of any single one of the governing equations:

$$\begin{aligned}
R_c(\vec{x}) &= \epsilon \frac{\partial \rho_f}{\partial t} + \nabla \cdot (\epsilon \rho_f \vec{V}) = 0 \\
R_m(\vec{x}) &= \epsilon \frac{\partial \vec{\eta}}{\partial t} + \nabla \cdot (\epsilon \vec{\eta} \vec{V}) + \nabla \cdot (\epsilon P \mathbf{I}) - \epsilon \rho_f \vec{g} + \mathcal{D} \mu \epsilon \vec{V} + \mathcal{F} \epsilon^2 \vec{\eta} |\vec{V}| = 0 \\
R_s(\vec{x}) &= (1-\epsilon)\rho_s C_{p,s} \frac{\partial T_s}{\partial t} - \nabla \cdot (\kappa_s \nabla T_s) + \alpha (T_s - T_f) - q_s = 0 \\
R_f(\vec{x}) &= \epsilon \frac{\partial \gamma_f}{\partial t} + \nabla \cdot (\epsilon h_f \vec{\eta}) - \epsilon \vec{g} \cdot \vec{\eta} + \alpha (T_f - T_s) - q_f = 0
\end{aligned} \tag{8.86}$$

where the subscripts c, m, s, and f refer to the continuity, momentum, solid energy, and fluid energy equations, respectively. When decomposed into the weak form, each of the residuals also includes the boundary integral terms. The solution vector $\vec{x}$ is a single vector containing all the problem unknowns. For instance, for structural problems, the solution desired may be the x, y, and z displacements. The solution vector therefore would be of the form:

$$\vec{x} = \begin{bmatrix} x_1 & y_1 & z_1 & x_2 & y_2 & z_2 & \cdots & x_n & y_n & z_n \end{bmatrix}^T \tag{8.87}$$

where the subscripts denote node numbers and the entries are actually the expansion coefficients C_j from Eq. (8.72) that are solved for. Hence, the information for each node in structural problems is usually held

adjacent within the solution vector, and the information for other nodes held in a neighboring manner. For the thermal-hydraulics equations solved in PRONGHORN, a similar method is utilized, and all problem unknowns are held in a single column vector, organized by node. Each residual is a function of these unknowns.

A residual, such as the generic residual in Eq. (8.85), can be expanded in a Taylor series for iteration $k+1$ about the previous iterate x^k. Neglecting higher-order terms, this provides the numerical algorithm for a basic Newton method.

$$\begin{aligned} R(\vec{x}^{k+1}) = R(\vec{x}^{k}) + \frac{\partial R(\vec{x}^{k})}{\partial x}(\vec{x}^{k+1} - \vec{x}^{k}) + \cdots = 0 \\ -R(\vec{x}^{k}) = \frac{\partial R(\vec{x}^{k})}{\partial x}\,\vec{\delta}^{k} \end{aligned} \tag{8.88}$$

where:

$$\vec{\delta}^{k} = \vec{x}^{k+1} - \vec{x}^{k} \tag{8.89}$$

While Eq. (8.88) is written for a single residual statement, oftentimes several governing equations are solved together, such as the continuity, momentum, fluid energy, and solid energy equations in PRONGHORN. MOOSE solves a tightly-coupled system by forming a "monolithic block" - a massive nonlinear system that is solved *together*, rather than in pieces. To be more specific, instead of solving the continuity equation alone, and passing the density to the momentum equation, at which point the momentum equation would solve for momentum, and then pass velocities back to the continuity equation in an iterative method, both the continuity and momentum equations are solved simultaneously in the same nonlinear solve. Hence, Eq. (8.88) in practice is more complex, since simultaneously we require a system of residuals to equal zero. For a multi-equation system, the Jacobian becomes more complex, and is replaced by a matrix instead of a single value. It should be noted that MOOSE has the capability to perform operator-split coupling methods.

$$\begin{aligned} \frac{\partial R(\vec{x}^{k})}{\partial x} \rightarrow \mathbf{J} \\ J_{ij} = \frac{\partial R_i}{\partial x_j} \end{aligned} \tag{8.90}$$

Then, the Newton method becomes a multi-equation solve, upgrading from that shown in Eq. (8.88) for a single residual to that in Eq. (8.91) for a system of equations (a system of residuals).

$$-\mathbf{R}(\vec{x}^{k}) = \mathbf{J}\,\vec{\delta}^{k} \tag{8.91}$$

Hence, the actual Jacobian is a matrix, where each row corresponds to a particular governing equation, represented by R_i, and each column refers to the partial derivative of that equation R_i with respect to one of the nonlinear variables x_j. For PRONGHORN, the nonlinear variables are ρ, $\vec{\eta}$, γ_s, and γ_f. The Jacobian is then built up by each of the residual statements given in Eq. (8.86) according to its definition in Eq. (8.90):

$$\mathbf{J} = \begin{bmatrix} \frac{\partial R_c}{\partial \rho} & \frac{\partial R_c}{\partial \vec{\eta}} & \frac{\partial R_c}{\partial \gamma_s} & \frac{\partial R_c}{\partial \gamma_f} \\ \frac{\partial R_m}{\partial \rho} & \frac{\partial R_m}{\partial \vec{\eta}} & \frac{\partial R_m}{\partial \gamma_s} & \frac{\partial R_m}{\partial \gamma_f} \\ \frac{\partial R_s}{\partial \rho} & \frac{\partial R_s}{\partial \vec{\eta}} & \frac{\partial R_s}{\partial \gamma_s} & \frac{\partial R_s}{\partial \gamma_f} \\ \frac{\partial R_f}{\partial \rho} & \frac{\partial R_f}{\partial \vec{\eta}} & \frac{\partial R_f}{\partial \gamma_s} & \frac{\partial R_f}{\partial \gamma_f} \end{bmatrix} \tag{8.92}$$

For physical situations where the material properties, such as thermal conductivity, are not very strong functions of any of the solution variables, then determining the Jacobian is as simple as taking derivatives of the residual, and entering these derivatives in the MOOSE App source code. However, if, for example, the fluid thermal conductivity k_f in R_f in Eq. (8.86) is a strong nonlinear function of ρ_f, $\vec{\eta}$, γ_s, or γ_f, then taking the derivative with respect to one of these nonlinear variables may be difficult and highly error-prone. If the material property is not even specified by an analytic function, such as nuclear cross sections, then it may be impossible to analytically specify the Jacobian, and some advanced preconditioning is needed to form an acceptable guess for the Jacobian. While the Jacobian does not influence the end simulation results, it does impact "how to get there," and a poor Jacobian means that the solve will not converge.

While the method utilized by MOOSE is termed "Jacobian-free," computation of the Jacobian (by the user) is not avoidable unless a very particular type of preconditioning is used. With finite difference preconditioning, the derivative in Eq. (8.90) is approximated by a finite difference of the residual statement, and is, in the limit of discretization error, exact. Because this type of preconditioning is relatively slow, the user must supply the on-diagonal Jacobian terms in the source code. The off-diagonal terms are by default zero, unless also supplied by source code. Even once the off-diagonal terms are included in the source code, they are not included in an actual solve unless specified in the `[Preconditioning]` block. These off-diagonal terms can in most situations be neglected, unless there is a very strong dependence of the equation in question (the row in question) to a different nonlinear variable. In most of the PRONGHORN kernels, the off-diagonal terms are neglected.

The numerical method used by MOOSE is termed "Jacobian-free" because, while the user must enter in the source code an exact or approximate Jacobian, the full Jacobian itself is not needed by the nonlinear solve. The JFNK method requires only the action of the Jacobian on a vector - a matrix-vector product. This matrix-vector product is the $\mathbf{J}\vec{\delta}^k$ term in Eq. (8.91). The Jacobian is a matrix, while the update vector, as described in Eq. (8.89), is the vector. This solution method is highly advantageous, since forming the Jacobian is expensive and would dominate the overall cost of a solve. The matrix-vector product is approximated using a finite difference:

$$\mathbf{J}\vec{\delta}^k \approx \frac{\mathbf{R}(\vec{x}^k + \epsilon\vec{\delta}^k) - \mathbf{R}(\vec{x}^k)}{\varepsilon} \tag{8.93}$$

where ε is a value chosen to optimize between discretization error (smaller ε yields more precise finite difference approximations) and machine precision (too small ε leads to machine precision difficulties). By approximating the product of the Jacobian on the solution vector

In MOOSE, Newton's method is implemented by:

1. Solve Eq. (8.88) for δ^k. As long as the Jacobian is nonzero, then this equation can be solved. This Jacobian could only be zero if 1) the user did not enter a Jacobian, in which case finite difference

preconditioning is needed, or 2) the governing equation has no terms that include the nonlinear variable that is to be solved for. This would be like solving $d^2T/dx^2 = 0$ for density, a variable that does not appear in any kernels.

2. Solve Eq. (8.89) to update the solution.

Newton's method sets the next solution guess $\vec{x}^{k+1}$ to be the zero of the line tangent to $R(\vec{x}^k)$. For simple roots (roots that do not also coincide with the root of a derivative of R), Newton's method converges quadratically. However, for multiple roots, Newton's method only converges linearly. These linear and quadratic convergence rates are local, however, and if the solve begins too far away from the solution, Newton's method might never converge. MOOSE capitalizes on the fact that the JFNK method only requires residual and Jacobian evaluations. This permits individual physics to be separated, and chained together in customizable manners for a high degree of specialization.

8.5 SUPG Weak Forms

The SUPG weak forms of the governing equations are:

$$\left(\epsilon\frac{\partial\rho_f}{\partial t},\psi\right) - \left(\epsilon\rho_f\vec{V},\nabla\psi\right) + <\epsilon\rho_f\vec{V},\psi> + \sum_{n_{el}}\left(\tau\vec{V}\left(\epsilon\frac{\partial\rho_f}{\partial t} + \nabla\cdot\epsilon\rho_f\vec{V}\right),\nabla\psi\right) = 0$$

$$\begin{aligned}\left(\epsilon\frac{\partial\eta_x}{\partial t},\psi\right) - \left(\epsilon\eta_x\vec{V},\nabla\psi\right) + <\epsilon\eta_x\vec{V} + \epsilon P\vec{I},\psi> - \left(\epsilon P\vec{I},\nabla\psi\right) + (W\eta_x,\psi) - \left(\epsilon\rho_f g_x,\psi\right) + \\ + \sum_{n_{el}}\left(\tau\vec{V}\left(\epsilon\frac{\partial\vec{\eta}}{\partial t} + \nabla\cdot\left(\epsilon\eta_x\vec{V}\right) + \nabla(\epsilon P) + \mathcal{D}\mu\epsilon V_x + \mathcal{F}\epsilon^2\rho_f|\vec{V}|\vec{V} - \epsilon\rho_f\vec{g}\right),\nabla\psi\right) = 0\end{aligned}$$

$$\begin{aligned}\left((1-\epsilon)\frac{\partial(\rho_s C_{p_s}T_s)}{\partial t},\psi\right) + (\kappa_s\nabla T_s,\nabla\psi) - <\kappa_s\nabla T_s,\psi> + \left(\alpha(T_s-T_f),\psi\right) - (q_s,\psi) + \\ \sum_{n_{el}}\left(\tau\vec{V}\left((1-\epsilon)\frac{\partial(\rho_s C_{p_s}T_s)}{\partial t} - \nabla\cdot(\kappa_s\nabla T_s) + \alpha(T_s-T_f) - q_s\right),\nabla\psi\right) = 0\end{aligned} \tag{8.94}$$

$$\begin{aligned}\left(\epsilon\frac{\partial\gamma_f}{\partial t},\psi\right) - \left(\epsilon h_f\vec{\eta},\nabla\psi\right) + <\epsilon h_f\vec{\eta}-,\psi> - (\epsilon\vec{g}\cdot\vec{\eta},\psi) + + \left(\alpha(T_f-T_s),\psi\right) - \left(q_f,\psi\right) + \\ + \sum_{n_{el}}\left(\tau\vec{V}\left(\epsilon\frac{\partial\gamma_f}{\partial t} + \nabla\cdot\left(\epsilon h_f\vec{\eta}\right) - \epsilon\vec{g}\cdot\vec{\eta} + \alpha(T_f-T_s) - q_f\right),\nabla\psi\right) = 0\end{aligned}$$

The SUPG stabilization is implemented in PRONGHORN by creating one SUPG kernel for each kernel to be stabilized, with the naming convention that the SUPG kernel is simply the kernel name followed by "SUPG." The SUPG stabilization is split up in this manner to permit a high level of generality.

8.6 SUPG Verification

Table 1. Summary of 2-D and 3-D MMS tests and convergence rates for PRONGHORN SUPG kernels. The spatial error is not assessed for the time kernels.

Kernel	Test Name		Linear	Quadratic
ContinuityEqnSUPG	continuity_eqn_SUPG	2-D	2.11 - 3.02	2.99 - 3.00
	continuity_eqn_SUPG_3D	3-D	2.26 - 3.05	2.94 - 3.03
MomConvectiveFluxSUPG	mom_conv_flux_SUPG	2-D	2.11 - 3.02	2.99 - 3.00
	mom_conv_flux_SUPG_3D	3-D	2.26 - 3.05	2.94 - 3.02
MomPressureGradientSUPG	mom_press_grad_SUPG_2D	2-D	2.11 - 3.02	2.99 - 3.00
	mom_press_grad_SUPG	3-D	2.28 - 3.05	2.94 - 3.02
MomFrictionForceSUPG	mom_friction_SUPG	2-D	2.99 - 3.00	2.82 - 2.93
	mom_friction_SUPG_3D	3-D	2.33 - 3.00	2.94 - 2.95
MomGravityForceSUPG	mom_gravity_SUPG	2-D	1.99 - 2.00	2.83 - 2.93
	mom_gravity_SUPG_3D	3-D		
FluidEnergyConvectiveFluxSUPG	fluid_energy_conv_flux_SUPG	2-D	2.05 - 2.08	2.95 - 2.97
	fluid_energy_conv_flux_SUPG_3D_v2	3-D	2.04 - 2.05	2.88 - 2.97
FluidEnergyGravityForceSUPG	fluid_energy_gravity_force_SUPG	2-D	2.06 - 2.10	2.93 - 2.97
	fluid_energy_gravity_force_SUPG_3D	3-D	2.05 - 2.06	2.86 - 2.99
FluidSolidConvectionSUPG	fluid_solid_convection_SUPG	2-D	2.00	2.99 - 3.57
		3-D		

8.7 PRONGHORN Input File

This section contains the PRONGHORN input file `1D_convection_diffusion.i` used to solve the 1-D convection diffusion equation to provide 1-D nodally-exact results demonstrating the successful implementation of SUPG stabilization.

```
# solves 1-D convection diffusion to produce figure showing
# how solution becomes unstable for Pe > 1

[GlobalParams]
  vel_x = vel_x
  porosity = 1.0
[]

[Mesh]
  type = GeneratedMesh
  dim = 1
  nx = 10
  xmin = 0.0
  xmax = 1.0
[]
```

```
[Variables]
  [./rho]
  [../]
[]

[AuxVariables]
  [./vel_x]
  [../]
[]

[Functions]
  [./vel_x]
    type = ParsedFunction
    value = 10
  [../]
[]

[AuxKernels]
  [./vel_x]
    type = FunctionAux
    function = 'vel_x'
    variable = vel_x
  [../]
[]

[Kernels]
  #active = 'continuity diffusion' # no stabilization
  [./continuity]
    type = ContinuityEqn
    variable = rho
  [../]
  [./continuitySUPG]
    type = ContinuityEqnSUPG
    variable = rho
  [../]
  [./diffusion]
    type = Diffusion
    variable = rho
  [../]
  [./diffusionSUPG] # zero contribution if rho is linear or lower order
    type = FluidEnergyDiffusionSUPG
    variable = rho
```

```
    k_fluid = 1.0
    T_fluid = rho
  [../]
[]

[BCs]
  [./left]
    type = DirichletBC
    variable = rho
    boundary = 'left'
    value = 0.0
  [../]
  [./right]
    type = DirichletBC
    variable = rho
    boundary = 'right'
    value = 1.0
  [../]
[]

[Materials]
  [./tau]
    type = Tau1DConvectionDiffusion
    k = 1
  [../]
[]

[Executioner]
  type = Steady
[]

[Outputs]
    exodus = true
  [./console]
    type = Console
    max_rows = 400
  [../]
[]
```