Wasatch Documentation

Tony Saad, James C. Sutherland, Amir Biglari, Alex W. Abboud

April 14, 2014

Wasatch is a finite volume computational fluid dynamics (CFD) code that employs the spatial operator (SpatialOps) and expression (ExprLib) libraries at its foundation. It is developed as a software component within the Uintah computational parallel framework. Wasatch provides the interface between the Uintah API and the various classes and tools supplied by the SpatialOps and ExprLib libraries making its development entirely based on the expressions graph abstraction.

Contents

Chapter 1

Capabilities

Wasatch currently supports basic physical models of transport phenomena in fluids and gases. These may be reactive with variable density or non-reactive. The working model is based on writing expressions that reflect physical processes, akin to mathematical terms in a partial differential equation (PDE). These expressions are subsequently used in a variety of transport equations as needed. Transport equations are derived from a base class and each specializes in solving a particular class of problems. Currently, Wasatch currently supports the following transport equations with constant density:

- Generic scalar transport
- Momentum transport
- Population balances via moment transport

A wide range of expressions are currently implemented in Wasatch. These include the standard convective and diffusive fluxes as well as stresses as well as a variety of source terms that are inherent from specific physical models such as precipitation among others.

1.1 Momentum Transport

Wasatch solves the variable density momentum equations

$$\frac{\partial \rho \mathbf{u}}{\partial t} = -\nabla \cdot \rho \mathbf{u}\mathbf{u} - \nabla \cdot \boldsymbol{\tau}_{ij} - \nabla p + \rho \mathbf{g} \tag{1.1}$$

in conjunction with the continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0. \tag{1.2}$$

Here, ρ stands for the density, $\mathbf{u} = (u, v, w)$ is the velocity field, p is the pressure, $\mathbf{g}$ is the gravity vector, and $\boldsymbol{\tau}_{ij}$ is the stress tensor given by

$$\boldsymbol{\tau}_{ij} = -2\mu S_{ij}; \quad S_{ij} \equiv \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{1}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}. \tag{1.3}$$

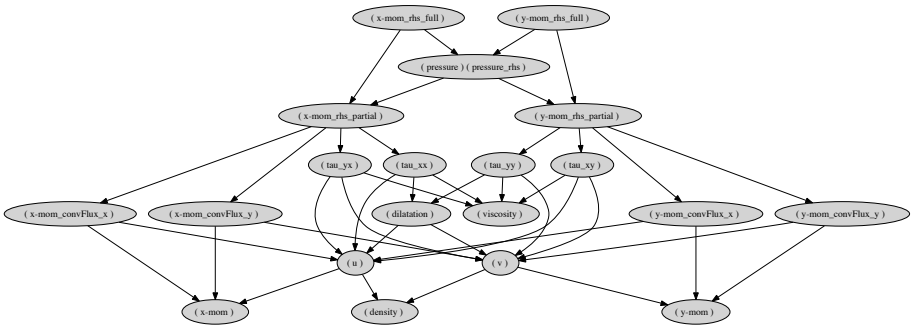


Figure 1.1: Expression graph for a two dimensional fluid flow problem using the Poisson equation for the pressure.

The rate of strain tensor S_{ij} is a fundamental quantity when using gradient diffusion turbulence models and will be used extensively in subsequent sections.

The code supports central, upwind, and over ten flux limiters for the discretization of the convective flux. The pressure is resolved by solving the pressure Poisson equation (PPE). This is constructed by first taking the divergence of the momentum equations. Then the continuity equation is used to substitute for the temporal term in the resulting equation. At the outset, one recovers a Poisson equation for the pressure

$$\nabla^2 p = \frac{\partial^2 \rho}{\partial t^2} + \nabla \cdot \nabla \cdot \rho \mathbf{u}\mathbf{u} + \nabla \cdot \nabla \cdot \boldsymbol{\tau}_{ij} - \nabla \cdot \rho \mathbf{g}. \tag{1.4}$$

The solution of Eq. (1.4) requires the inversion of a linear system of equations. At the time of writing, Wasatch can solve the constant density momentum equations in three dimensions with periodic boundary conditions. The solution is based on the expressions graph showing that a basic CFD algorithm may be achieved using the graph based algorithm approach. A sample graph for a two dimensional fluid flow is shown in Fig. 1.1.

Pressure Projection Wasatch uses the Pressure-Poisson-Equation (PPE) formulation to handle the pressure-velocity couple. The PPE formulation consists of first taking the divergence of the momentum equations and substituting the continuity equation to yield a Poisson equation for the pressure. This version of the pressure projection provides an elegant and straight-forward equation for the pressure. Part of the difficulty of using the PPE formulation resides in the specification of appropriate boundary and initial conditions. While boundary conditions correspond to "natural" conditions imposed on physical quantities such as mass flowrate, pressure gradient, and velocity; specification of the initial conditions requires further attention on the Pressure poisson equation.

Constant Density For a constant density flow, the Pressure-Poisson-Equation (1.4) reduces to

$$\nabla^2 p = \nabla \cdot \nabla \cdot \rho \mathbf{u}\mathbf{u} + \nabla \cdot \nabla \cdot \boldsymbol{\tau}_{ij} - \nabla \cdot \rho \mathbf{g}. \tag{1.5}$$

1.1.1 LES Modeling

Wasatch uses Favre filtering as the basis for its LES approach. Favre filtering is defined as a density-weighted filtering operation to accomodate variable density. It is given by the following operation

$$\overline{\rho\phi} = \overline{\rho}\widetilde{\phi}, \tag{1.6}$$

where the overbar denotes a traditional filtering operation. A traditional filtering operation consists of a convolution of a transported variable (i.e. a variable that one wishes to filter) with a filter. This is written as

$$\overline{\phi} = \int_{\Omega} G(\mathbf{r})\phi(\mathbf{x}-\mathbf{r})\,\mathrm{d}\mathbf{r}. \tag{1.7}$$

Filtered Momentum Equations To make further headway, we apply the traditional filter to the governing fluid transport equations. Starting with continuity, we have

$$\frac{\partial\overline{\rho}}{\partial t} + \nabla\cdot\overline{\rho\mathbf{u}} = 0. \tag{1.8}$$

Then, one makes use of the Favre filtering operation by setting $\overline{\rho\mathbf{u}} = \overline{\rho}\widetilde{\mathbf{u}}$. At the outset, the Favre-filtered continuity equation is given by

$$\frac{\partial\overline{\rho}}{\partial t} + \nabla\cdot\overline{\rho}\widetilde{\mathbf{u}} = 0. \tag{1.9}$$

Moving on to the momentum equation, the filtering operation yields

$$\frac{\partial\overline{\rho\mathbf{u}}}{\partial t} = -\nabla\cdot\overline{\rho\mathbf{u}\mathbf{u}} - \nabla\cdot\overline{\boldsymbol{\tau}_{ij}} - \nabla\overline{p} + \overline{\rho\mathbf{g}}. \tag{1.10}$$

It is best to transform to Favre-filtered terms one at a time. This leads to the following

- $\frac{\partial\overline{\rho\mathbf{u}}}{\partial t} = \frac{\partial\overline{\rho}\widetilde{\mathbf{u}}}{\partial t}$
- $\nabla\cdot\overline{\rho\mathbf{u}\mathbf{u}} = \nabla\cdot\overline{\rho}\widetilde{\mathbf{u}\mathbf{u}} = \nabla\cdot\overline{\rho}\widetilde{\mathbf{u}}\widetilde{\mathbf{u}} + (\nabla\cdot\overline{\rho}\widetilde{\mathbf{u}\mathbf{u}} - \nabla\cdot\overline{\rho}\widetilde{\mathbf{u}}\widetilde{\mathbf{u}}) \equiv \nabla\cdot\overline{\rho}\widetilde{\mathbf{u}}\widetilde{\mathbf{u}} + \nabla\cdot\boldsymbol{\tau}^R_{ij}; \quad \boldsymbol{\tau}^R_{ij} \equiv \overline{\rho}\widetilde{\mathbf{u}\mathbf{u}} - \overline{\rho}\widetilde{\mathbf{u}}\widetilde{\mathbf{u}}$
- $\nabla\cdot\overline{\boldsymbol{\tau}_{ij}} = \nabla\cdot\widetilde{\boldsymbol{\tau}_{ij}} + \left(\nabla\cdot\overline{\boldsymbol{\tau}_{ij}} - \nabla\cdot\widetilde{\boldsymbol{\tau}_{ij}}\right) \equiv \nabla\cdot\widetilde{\boldsymbol{\tau}_{ij}} + \nabla\cdot\boldsymbol{\tau}^L_{ij}$
- $\overline{\rho\mathbf{g}} = \overline{\rho}\mathbf{g}$

where $\boldsymbol{\tau}^R_{ij}$ and $\boldsymbol{\tau}^L_{ij}$ are subgrid-scale stress tensors (SGS) that result from applying filtering to the convective and diffusive terms in the momentum equations, respectively. At the outset, the filtered momentum equations may be written as

$$\frac{\partial\overline{\rho}\widetilde{\mathbf{u}}}{\partial t} = -\nabla\cdot\overline{\rho}\widetilde{\mathbf{u}}\widetilde{\mathbf{u}} - \nabla\cdot\widetilde{\boldsymbol{\tau}_{ij}} - \nabla\overline{p} - \nabla\cdot\boldsymbol{\tau}^R_{ij} - \nabla\cdot\underbrace{\boldsymbol{\tau}^L_{ij}}_{\text{neglect}} + \overline{\rho\mathbf{g}}. \tag{1.11}$$

It is evident from Eq. (1.11) that both subgrid tensors $\boldsymbol{\tau}^R_{ij}$ and $\boldsymbol{\tau}^L_{ij}$ require closure. However, it may be argued that the subgrid tensor $\boldsymbol{\tau}^L_{ij}$ resulting from the viscous terms is negligible compared

to $\boldsymbol{\tau}_{ij}^R$ that results from filtering of the convective terms (Geurts et al., 1993; Vreman et al., 1995). Hence, the resulting filtered momentum equations take the form

$$\frac{\partial \overline{\rho}\widetilde{\mathbf{u}}}{\partial t} = -\nabla\cdot\overline{\rho}\widetilde{\mathbf{u}}\widetilde{\mathbf{u}} - \nabla\cdot\widetilde{\boldsymbol{\tau}}_{ij} - \nabla\overline{p} - \nabla\cdot\boldsymbol{\tau}_{ij}^R + \overline{\rho}\mathbf{g}. \tag{1.12}$$

The subgrid stress tensor τ_{ij}^R requires closure as it introduces unknown quantities. The subject of closure is at the heart of the LES method and will be discussed in the next subsection.

1.1.1.1 Subgrid Closure

The most common approach to modeling the subgrid-scale tensors is based on eddy-viscosity models. First propsed by Smagorinsky (1963), these have the general form

$$\boldsymbol{\tau}_{ij}^R = \overbrace{\underbrace{\boldsymbol{\tau}_{ij}^R - \tfrac{1}{3}\delta_{ij}\boldsymbol{\tau}_{kk}^R}_{\text{deviatoric}}}^{\boldsymbol{\tau}_{ij}^{\mathrm{d}}} + \overbrace{\underbrace{\tfrac{1}{3}\delta_{ij}\boldsymbol{\tau}_{kk}^R}_{\text{isotropic}}}^{\boldsymbol{\tau}_{ij}^{\mathrm{s}}}. \tag{1.13}$$

The deviatoric subgrid-stress tensor $\boldsymbol{\tau}_{ij}^{\mathrm{d}}$ is modeled as

$$\boldsymbol{\tau}_{ij}^{\mathrm{d}} = -2\mu_{\mathrm{t}}\widetilde{S}_{ij}, \tag{1.14}$$

where ν_{t} is the eddy- or turbulent-viscosity and $\widetilde{S}_{ij}$ is the Favre-filtered rate of strain - a computable quantity.

The isotropic subgrid-stress tensor $\boldsymbol{\tau}_{ij}^{\mathrm{s}}$ may be absorbed into the filtered pressure or entirely neglected (Erlebacher et al., 1992; Garnier et al., 2009). This is based on the observation that small scales are generally incompressible (Garnier et al., 2009). In fact, $\boldsymbol{\tau}_{kk}$ may be written in terms of the subgrid Mach number as

$$\boldsymbol{\tau}_{kk} = \gamma M_{\mathrm{sgs}}^2 \overline{p}. \tag{1.15}$$

where γ is the ratio of specific heats and M_{sgs} is the subgrid Mach number whose values is expected to be small when the turbulent Mach number of the large scales is low.

Upon substitution of Eq. (1.13) into Eq. (1.11) one recovers

$$\frac{\partial \overline{\rho}\widetilde{\mathbf{u}}}{\partial t} = -\nabla\cdot\overline{\rho}\widetilde{\mathbf{u}}\widetilde{\mathbf{u}} - \nabla\cdot\widetilde{\boldsymbol{\tau}}_{ij} - \nabla\overline{p} - \nabla\cdot\boldsymbol{\tau}_{ij}^{\mathrm{d}} + \overline{\rho}\mathbf{g}. \tag{1.16}$$

Given that all quantities – except ν_{t} – are computable, the essence of all eddy-viscosity models is then to model the turbulent viscosity ν_{t}.

Constant Smagorinsky-Lilly Model In the constant Smagorinsky-Lilly model, a simple algebraic relation is proposed for the eddy-viscosity (Smagorinsky, 1963). This form may be written as

$$\mu_{\mathrm{t}} = \overline{\rho}(C_{\mathrm{s}}\Delta)^2\widetilde{\mathcal{S}} = \overline{\rho}L_s^2\widetilde{\mathcal{S}}, \tag{1.17}$$

where C_{s} is a constant, Δ is an average measure of the cell size, usually taken as the average cell volume $V^{1/3}$, and $\widetilde{\mathcal{S}} = \sqrt{2\widetilde{S}_{ij}\widetilde{S}_{ij}}$. Also, L_s is commonly referred to as the mixing length and is a measure of the characteristic mixing length of turbulent eddies.

Dynamic Smagorinsky Model While the constant Smagorinsky-Lilly model provides a well-rounded simple LES model, it fails to capture appropriate flow behaviour near walls or under low-Reynolds-number conditions where the model coefficient C_s needs to be attenuated or set to zero. The dynamic Smagorinsky models provides a means to allow for saptio-temporal variation of C_s. The idea behind the dynamic model is simple: apply a test filter to the grid-filtered velocity field and then use the difference between the test-filtered and grid-filtered stresses as a measure of how resolved the simulation is and calibrate the Smagorinsky model coefficient appropriately. In what follows, test-filtering is denoted by an over-hat $\widehat{\cdot}$ while grid filtered fields are denoted by an overbar.

Constant Density For a constant density flow, the test-filtered momentum equations are

$$\frac{\partial \widehat{\overline{\mathbf{u}}}}{\partial t} = -\nabla \cdot \widehat{\overline{\mathbf{u}}\overline{\mathbf{u}}} - \nabla \cdot \widehat{\overline{\boldsymbol{\tau}}}_{ij} - \nabla \widehat{\overline{p}}. \tag{1.18}$$

where the overbar denotes grid-filtered quantities. The test-filtered convective fluxes are unknown and require modeling. We follow standard expansion and set

$$\nabla \cdot \widehat{\overline{\mathbf{u}}\overline{\mathbf{u}}} = \nabla \cdot \widehat{\overline{\mathbf{u}}}\widehat{\overline{\mathbf{u}}} + \left(\nabla \cdot \widehat{\overline{\mathbf{u}}\overline{\mathbf{u}}} - \nabla \cdot \widehat{\overline{\mathbf{u}}}\widehat{\overline{\mathbf{u}}}\right) \equiv \nabla \cdot \widehat{\overline{\mathbf{u}}}\widehat{\overline{\mathbf{u}}} + \nabla \cdot \mathbf{T}^{\mathrm{R}}_{ij}; \quad \mathbf{T}^{\mathrm{R}}_{ij} \equiv \widehat{\overline{\mathbf{u}}\overline{\mathbf{u}}} - \widehat{\overline{\mathbf{u}}}\widehat{\overline{\mathbf{u}}} \tag{1.19}$$

where $\mathbf{T}^{\mathrm{R}}_{ij}$ is the test-filtered subgrid-scale stress tensors that results from applying the test-filter to the grid-filtered convective terms. In a manner similar to the grid-filtered residual stresses, the deviatoric part of $\mathbf{T}^{\mathrm{R}}_{ij}$ may be modeled using an eddy-viscosity model

$$\mathbf{T}^{\mathrm{R}}_{ij} = \overbrace{\underbrace{\mathbf{T}^{\mathrm{R}}_{ij} - \tfrac{1}{3}\delta_{ij}\mathbf{T}^{\mathrm{R}}_{ij}}_{\text{deviatoric}}}^{\mathbf{T}^{\mathrm{d}}_{ij}} + \overbrace{\underbrace{\tfrac{1}{3}\delta_{ij}\mathbf{T}^{\mathrm{R}}_{ij}}_{\text{isotropic}}}^{\mathbf{T}^{\mathrm{s}}_{ij}}. \tag{1.20}$$

Assuming that the test-filtered residual stresses have the same model coefficient as the grid-filtered streses, we set

$$\mathbf{T}^{\mathrm{d}}_{ij} = -2\mu_{\mathrm{t}}\widehat{\overline{S}}_{ij}; \quad \mu_{\mathrm{t}} \equiv C_{\mathrm{d}}\widehat{\overline{\Delta}}^2\widehat{\overline{\mathcal{S}}}, \tag{1.21}$$

where $\widehat{\overline{S}}_{ij} \equiv \frac{1}{2}\left(\frac{\partial \widehat{\overline{u}}_i}{\partial x_j} + \frac{\partial \widehat{\overline{u}}_j}{\partial x_i}\right)$, $\widehat{\overline{\mathcal{S}}} = \sqrt{2\widehat{\overline{S}}_{ij}\widehat{\overline{S}}_{ij}}$, and C_{d} is the dynamic Smagorinsky coefficient, akin to the one shown in 1.17. Recall that the grid-filtered deviatoric residual stress tensor is modeled as

$$\boldsymbol{\tau}^{\mathrm{d}}_{ij} = -2\mu_{\mathrm{t}}\overline{S}_{ij}; \quad \mu_{\mathrm{t}} \equiv C_{\mathrm{d}}\overline{\Delta}^2\overline{\mathcal{S}}. \tag{1.22}$$

The purpose of the dynamic procedure to determine C_{d} based on local flow conditions. If one could relate 1.21 and 1.22, then an equation for C_{d} could be derived. Germano constructed an identity that does that. By taking the test-filter of $\boldsymbol{\tau}^{\mathrm{R}}_{ij}$, and subtracting it from $\mathbf{T}^{\mathrm{R}}_{ij}$, one recovers

$$\mathcal{L}_{ij} \equiv \mathbf{T}^{\mathrm{R}}_{ij} - \widehat{\boldsymbol{\tau}^{\mathrm{R}}_{ij}} = \left(\widehat{\overline{\mathbf{u}}\overline{\mathbf{u}}} - \widehat{\overline{\mathbf{u}}}\widehat{\overline{\mathbf{u}}}\right) - \left(\widehat{\overline{\mathbf{u}}\overline{\mathbf{u}}} - \widehat{\overline{\mathbf{u}}}\widehat{\overline{\mathbf{u}}}\right) = \widehat{\overline{\mathbf{u}}}\widehat{\overline{\mathbf{u}}} - \widehat{\overline{\mathbf{u}}}\widehat{\overline{\mathbf{u}}} = \widehat{\overline{u}_i\overline{u}_j} - \widehat{\overline{u}_i}\widehat{\overline{u}_j} \tag{1.23}$$

$\mathcal{L}_{ij}$ is a known quantity and will be used to calibrate the model's coefficient. A model for $\mathcal{L}_{ij}$ is

$$\mathcal{L}^{\mathrm{d}}_{ij} \equiv \mathbf{T}^{\mathrm{d}}_{ij} - \widehat{\boldsymbol{\tau}^{\mathrm{d}}_{ij}} \tag{1.24}$$

or

$$\mathcal{L}_{ij}^{\mathrm{d}} = -2C_{\mathrm{d}}\widehat{\overline{\Delta}^2}\widehat{\mathcal{S}}\widehat{\overline{S}}_{ij} + 2C_{\mathrm{d}}\overline{\Delta}^2\widehat{\mathcal{S}\overline{S}}_{ij} \equiv 2C_{\mathrm{d}}\mathbf{M}_{ij}; \quad \mathbf{M}_{ij} \equiv \overline{\Delta}^2\widehat{\mathcal{S}\overline{S}}_{ij} - \widehat{\overline{\Delta}^2}\widehat{\mathcal{S}}\widehat{\overline{S}}_{ij} \tag{1.25}$$

Given that $\mathcal{L}_{ij} \approx \mathcal{L}_{ij}^{\mathrm{d}}$, then one can write (remember that $\mathcal{L}_{ij}$ is a known quantity)

$$2C_{\mathrm{d}}\mathbf{M}_{ij} \approx \mathcal{L}_{ij} \tag{1.26}$$

Of course this is an overdetermined system of equations and C_{d} cannot be calculated from the above equation. Instead, Germano suggested contracting the above equation with $\mathbf{M}_{ij}$ to recover

$$C_{\mathrm{d}} = \frac{1}{2}\frac{\mathcal{L}_{ij}\mathbf{M}_{ij}}{\mathbf{M}_{ij}\mathbf{M}_{ij}} \tag{1.27}$$

In fact, this apparently arbitrary contraction may be rigorously derived by minimizing the mean-square error. The error is defined as the difference between the subtest stress $\mathcal{L}_{ij}$ and its model $\mathcal{L}_{ij}^{\mathrm{d}}$ as

$$\varepsilon_{ij} = \mathcal{L}_{ij}^{\mathrm{d}} - \mathcal{L}_{ij} \tag{1.28}$$

The mean-square error is

$$E \equiv \varepsilon_{ij}\varepsilon_{ij} = \left(\mathcal{L}_{ij}^{\mathrm{d}} - \mathcal{L}_{ij}\right)\left(\mathcal{L}_{ij}^{\mathrm{d}} - \mathcal{L}_{ij}\right) \tag{1.29}$$

upon expansion, this leads to

$$E \equiv \left(2C_{\mathrm{d}}\mathbf{M}_{ij} - \mathcal{L}_{ij}\right)^2 \tag{1.30}$$

To find the extremum of the error with respect to C_{d}, we take its derivative and set it to zero. This returns

$$\frac{\partial E}{\partial C_{\mathrm{d}}} = 2 \times 2\mathbf{M}_{ij} \times \left(2C_{\mathrm{d}}\mathbf{M}_{ij} - \mathcal{L}_{ij}\right) = 8C_{\mathrm{d}}\mathbf{M}_{ij}\mathbf{M}_{ij} - 4\mathcal{L}_{ij}\mathbf{M}_{ij} = 0 \tag{1.31}$$

or

$$C_{\mathrm{d}} = \frac{1}{2}\frac{\mathcal{L}_{ij}\mathbf{M}_{ij}}{\mathbf{M}_{ij}\mathbf{M}_{ij}} \propto \frac{\mathcal{L}_{ij}\mathbf{M}_{ij}}{\mathbf{M}_{ij}\mathbf{M}_{ij}} \tag{1.32}$$

This tells us that, when 1.32 is true, then the mean-square error is at an extremum. To identify the type of extremum, we look at the second derivative of E, namely

$$\frac{\partial^2 E}{\partial C_{\mathrm{d}}^2} = 8\mathbf{M}_{ij}\mathbf{M}_{ij} \geq 0 \tag{1.33}$$

This result indicates that the curvature of E is concave up and that the extremum is a minimum.

In practice, one calculates the product $C_{\mathrm{d}}\overline{\Delta}^2$ so that

$$C_{\mathrm{d}}\overline{\Delta}^2 = \frac{1}{2}\frac{\mathcal{L}_{ij}\mathbf{M}_{ij}}{\mathbf{M}_{ij}\mathbf{M}_{ij}}; \quad \mathbf{M}_{ij} \equiv \widehat{\overline{\mathcal{S}}\overline{S}}_{ij} - \delta^2\widehat{\overline{\mathcal{S}}}\widehat{\overline{S}}_{ij}, \quad \delta \equiv \frac{\widehat{\overline{\Delta}}}{\overline{\Delta}} \tag{1.34}$$

where δ is the ratio of filter widths. In Wasatch, we use an isotropic box filter with a width $\widehat{\overline{\Delta}} = 3\overline{\Delta}$ (i.e. $\delta = 3$).

Variable Density For a variable density flow, things are slightly more complicated due to Favre filtering. Recall that Favre filtering is defined as $\overline{\rho\phi} \equiv \overline{\rho}\widetilde{\phi}$. When test filtered, we make the following modification

$$\widehat{\overline{\rho\phi}} = \widehat{\overline{\rho}}\widehat{\widetilde{\phi}} = \widehat{\overline{\rho}}\check{\widetilde{\phi}} \tag{1.35}$$

where the over-check, "ˇ", operation denotes a test-filtered quantity in the Favre sense. We will show below that there is no need to define the over-check filtering and thus view it as a mere mathematical device that allows us to arrive at the final dynamic formulation.

We start with the test-filtered variable density equations, given by

$$\frac{\partial\widehat{\overline{\rho\mathbf{u}}}}{\partial t} = -\nabla\cdot\widehat{\overline{\rho\mathbf{u}\mathbf{u}}} - \nabla\cdot\widehat{\overline{\boldsymbol{\tau}}}_{ij} - \nabla\widehat{\overline{p}}. \tag{1.36}$$

Because we are only interested in the residual stresses arising from test-filtering, we only need to focus on the convective fluxes, which are responsible for generating the residual dissipation. We therefore set

$$\nabla\cdot\widehat{\overline{\rho\mathbf{u}\mathbf{u}}} = \nabla\cdot\widehat{\overline{\rho}}\check{\widetilde{\mathbf{u}\mathbf{u}}} = \nabla\cdot\widehat{\overline{\rho}}\check{\widetilde{\mathbf{u}}}\check{\widetilde{\mathbf{u}}} + \left(\nabla\cdot\widehat{\overline{\rho}}\check{\widetilde{\mathbf{u}\mathbf{u}}} - \nabla\cdot\widehat{\overline{\rho}}\check{\widetilde{\mathbf{u}}}\check{\widetilde{\mathbf{u}}}\right) \equiv \nabla\cdot\widehat{\overline{\rho}}\check{\widetilde{\mathbf{u}}}\check{\widetilde{\mathbf{u}}} + \nabla\cdot\mathbf{T}^{\mathrm{R}}_{ij}; \quad \mathbf{T}^{\mathrm{R}}_{ij} \equiv \widehat{\overline{\rho}}\check{\widetilde{\mathbf{u}\mathbf{u}}} - \widehat{\overline{\rho}}\check{\widetilde{\mathbf{u}}}\check{\widetilde{\mathbf{u}}} \tag{1.37}$$

As usual, $\mathbf{T}^{\mathrm{R}}_{ij}$ is decomposed into a deviatoric and isotropic components as shown in 1.20. We neglect the isotropic part and model the deviatoric stress using the standard Smagorinsky model as

$$\mathbf{T}^{\mathrm{d}}_{ij} = -2C_{\mathrm{d}}\widehat{\overline{\rho}}\widehat{\Delta}^2\widehat{\widetilde{\mathcal{S}}}\widehat{\widetilde{S}}_{ij}, \tag{1.38}$$

To make further headway, we now take the difference between the grid-filtered and test-filtered residual stresses

$$\mathcal{L}_{ij} \equiv \mathbf{T}^{\mathrm{R}}_{ij} - \widehat{\boldsymbol{\tau}^{\mathrm{R}}_{ij}} = \widehat{\overline{\rho}\widetilde{u}_i\widetilde{u}_j} - \widehat{\overline{\rho}}\check{\widetilde{u}}_i\check{\widetilde{u}}_j. \tag{1.39}$$

At this point, the over-parenthese operation is still undefined which makes $\mathcal{L}_{ij}$ useless unless we can rewrite the undefined quantities in terms of known ones. This is easily accomplished by using 1.35. This allows us to write

$$\widehat{\overline{\rho}}\check{\widetilde{u}}_i\check{\widetilde{u}}_j = \frac{1}{\widehat{\overline{\rho}}}\widehat{\overline{\rho}}\check{\widetilde{u}}_i\widehat{\overline{\rho}}\check{\widetilde{u}}_j = \frac{1}{\widehat{\overline{\rho}}}\widehat{\overline{\rho}\widetilde{u}_i}\widehat{\overline{\rho}\widetilde{u}_j} \tag{1.40}$$

which is an easily calculated quantity. Then

$$\mathcal{L}_{ij} = \widehat{\overline{\rho}\widetilde{u}_i\widetilde{u}_j} - \frac{1}{\widehat{\overline{\rho}}}\widehat{\overline{\rho}\widetilde{u}_i}\widehat{\overline{\rho}\widetilde{u}_j} \tag{1.41}$$

making $\mathcal{L}_{ij}$ a computable tensor. The final step in the dynamic model is to select a model for $\mathcal{L}_{ij}$, more specifically, for its deviatoric part. For this we use

$$\mathcal{L}^{\mathrm{d}}_{ij} = \mathbf{T}^{\mathrm{d}}_{ij} - \widehat{\boldsymbol{\tau}^{\mathrm{d}}_{ij}} = 2C_{\mathrm{d}} \underbrace{\left(\overline{\Delta}^2 \widehat{\overline{\rho}\widetilde{\mathcal{S}}\widetilde{S}}_{ij} - \widehat{\Delta}^2 \widehat{\overline{\rho}}\widehat{\widetilde{\mathcal{S}}}\widehat{\widetilde{S}}_{ij} \right)}_{\mathbf{M}_{ij}} \equiv 2C_{\mathrm{d}}\mathbf{M}_{ij} \tag{1.42}$$

finally, we set $\mathcal{L}_{ij} = \mathcal{L}^{\mathrm{d}}_{ij}$ and contract with $\mathbf{M}_{ij}$ to recover

$$C_{\mathrm{d}} = \frac{1}{2}\frac{\mathcal{L}_{ij}\mathbf{M}_{ij}}{\mathbf{M}_{ij}\mathbf{M}_{ij}} \tag{1.43}$$

Again, it is more customary to calculate the product $C_{\mathrm{d}}\overline{\Delta}^2$ so that

$$C_{\mathrm{d}}\overline{\Delta}^2 = \frac{1}{2}\frac{\mathcal{L}_{ij}\mathbf{M}_{ij}}{\mathbf{M}_{ij}\mathbf{M}_{ij}}; \quad \mathbf{M}_{ij} \equiv \widehat{\overline{\rho}\widetilde{\mathcal{S}}\widetilde{S}}_{ij} - \delta^2 \widehat{\overline{\rho}}\widehat{\widetilde{\mathcal{S}}}\widehat{\widetilde{S}}_{ij}, \quad \delta \equiv \frac{\widehat{\Delta}}{\overline{\Delta}} \tag{1.44}$$

An important feature of our formulation is that one recovers the constant density version of the dynaimc model, a desirable feature in our code to avoid unecessary branch and promote polymorphism.

Wall-Adapting Local Eddy-Viscosity (WALE) In the WALE model (Geurts et al., 1993), the eddy-viscosity is modeled as

$$\mu_t = \rho(C_{\mathrm{w}}\Delta)^2 \frac{(S^d_{ij}S^d_{ij})^{3/2}}{(\widetilde{S}_{ij}\widetilde{S}_{ij})^{5/2} + (S^d_{ij}S^d_{ij})^{5/4}}. \tag{1.45}$$

Here, C_{w} is the WALE model constant, Δ is the average cell size, and S^d_{ij} is the traceless symmetric part f the square of the velocity gradient tensor, given by (Nicoud and Ducros, 1999)

$$S^d_{ij} = \tfrac{1}{2}\left(g^2_{ij} + g^2_{ji}\right) - \tfrac{1}{3}\delta_{ij}g^2_{kk}; \quad g^2_{ij} = g_{ik}g_{kj}; \quad g_{ij} = \frac{\partial u_i}{\partial x_j}. \tag{1.46}$$

Care must be taken in evaluating the components of S^d_{ij}.

Vreman Model The Vreman model Vreman (2004) is supposed to compete with the dynamic model by performing analytical filtering thus avoiding the computationally-intensive dynamic filtering of the dynamic model. The model is relatively simple to implement and is given by

$$\mu_t = \rho C_{\mathrm{v}}\Delta^2 \sqrt{\frac{B_\beta}{\alpha_{ij}\alpha_{ij}}}, \tag{1.47}$$

where

$$C_{\mathrm{v}} \approx 2.5C^2_{\mathrm{s}}, \tag{1.48}$$

$$\alpha_{ij} = \frac{\partial \widetilde{u_j}}{\partial x_i}, \tag{1.49}$$

and

$$B_\beta = \beta_{11}\beta_{22} - \beta_{12}^2 + \beta_{11}\beta_{33} - \beta_{13}^2 + \beta_{22}\beta_{33} - \beta_{23}^2; \quad \beta_{ij} = \alpha_{mi}\alpha_{mj}. \tag{1.50}$$

1.2 Generic Scalar Transport

Wasatch can solve any scalar transport equation of the general form

$$\frac{\partial \rho\phi}{\partial t} = -\nabla \cdot \rho\phi\mathbf{u} + \nabla \cdot \rho\Gamma_\phi \nabla\phi + S_\phi. \tag{1.51}$$

This general form is basically the strong form of the scalar transport equation when density is variable. Wasatch has the capability of choosing between the strong form or the weak form of this transport equation. Density type can also be specified in the input files of wasatch to determine if it is constant or variable. Regarding these capabilities we can have three more possible forms of transport equations rather than 1.51. Those forms are:

- Strong form with constant density,

$$\frac{\partial \phi}{\partial t} = -\nabla \cdot \phi\mathbf{u} + \nabla \cdot \Gamma_\phi \nabla\phi + \frac{S_\phi}{\rho}. \tag{1.52}$$

- Weak form with variable density,

$$\frac{\partial \phi}{\partial t} = -\mathbf{u} \cdot \nabla\phi + \frac{1}{\rho}\nabla \cdot \rho\Gamma_\phi \nabla\phi + \frac{S_\phi}{\rho}. \tag{1.53}$$

- Weak from with constant density

$$\frac{\partial \phi}{\partial t} = -\mathbf{u} \cdot \nabla\phi + \nabla \cdot \Gamma_\phi \nabla\phi + \frac{S_\phi}{\rho}. \tag{1.54}$$

Wasatch can now solve transport equations in strong form with either constant or variable density. Solving the weak form of the equation has not been implemented yet and is in progress.

The main reason that wasatch keeps both strong and weak form is to compare their result in future to see which one is more beneficial. In strong form with variable density wasatch solves for, $\rho\phi$, and in order to update the for scalar variable we need to solve a nonlinear equation where, $\rho\phi$ and $\rho = F(\phi)$ are given, in order to find the scalar variable in the current time step. On the other hand, weak form has its own problems. For example to solve weak form equations' convective terms in order to apply limiters on this term this equation is used

$$\mathbf{u} \cdot \nabla\phi = \nabla \cdot \phi\mathbf{u} - \phi\nabla \cdot \mathbf{u}. \tag{1.55}$$

And limiters can be used on the terms on the RHS easily. The scalar variable in this case should be assumed to be approximately constant in each cell on the grid.

1.2.1 LES Modeling

The process of filtering the generic scalar transport equation follows from that used for the momentum equations.

Filtered Scalar Transport Equation By applying the traditional filter on the strong form of the scalar transport equation with variable density for capturing the most general form of the equation, the filtered scalar equation will be:

$$\frac{\partial \overline{\rho\phi}}{\partial t} = -\nabla \cdot \overline{\rho\phi\mathbf{u}} + \nabla \cdot \overline{\rho\Gamma_\phi \nabla\phi} + \overline{S_\phi} \tag{1.56}$$

Application of the Favre filtering to the scalar transport equation term-by-term yields:

- $\frac{\partial \overline{\rho\phi}}{\partial t} = \frac{\partial \overline{\rho}\widetilde{\phi}}{\partial t}$
- $\nabla \cdot \overline{\rho\phi\mathbf{u}} = \nabla \cdot \overline{\rho}\widetilde{\phi\mathbf{u}} = \nabla \cdot \overline{\rho}\widetilde{\phi}\widetilde{\mathbf{u}} + \left(\nabla \cdot \overline{\rho}\widetilde{\phi\mathbf{u}} - \nabla \cdot \overline{\rho}\widetilde{\phi}\widetilde{\mathbf{u}}\right) \equiv \nabla \cdot \overline{\rho}\widetilde{\phi}\widetilde{\mathbf{u}} + \nabla \cdot J_t^R; \quad J_t^R \equiv \overline{\rho}\widetilde{\phi\mathbf{u}} - \overline{\rho}\widetilde{\phi}\widetilde{\mathbf{u}}$
- $\nabla \cdot \overline{\rho\Gamma_\phi \nabla\phi} = \nabla.(\overline{\rho}\widetilde{\Gamma_\phi \nabla\phi}) = \nabla.(\overline{\rho}\overline{\Gamma_\phi}\nabla\widetilde{\phi}) + \left(\nabla.(\overline{\rho}\widetilde{\Gamma_\phi \nabla\phi}) - \nabla.(\overline{\rho}\overline{\Gamma_\phi}\nabla\widetilde{\phi})\right) = \nabla.(\overline{\rho}\overline{\Gamma_\phi}\nabla\widetilde{\phi}) + \nabla.J_t^L; \quad J_t^L = \nabla.(\overline{\rho}\widetilde{\Gamma_\phi \nabla\phi}) - \nabla.(\overline{\rho}\overline{\Gamma_\phi}\nabla\widetilde{\phi})$
- $\overline{S_\phi} = \overline{S_\phi}$

where J_t^R and J_t^L are subgrid-scale diffusive fluxes, which are part of the filtered convective and diffusive terms in the scalar transport equations after applying Favre averaging, respectively. After substituting all of these terms back into the filtered scalar equation, the following equation will be obtained:

$$\frac{\partial \overline{\rho}\widetilde{\phi}}{\partial t} = \nabla \cdot \overline{\rho}\widetilde{\phi}\widetilde{\mathbf{u}} + \nabla.(\overline{\rho}\overline{\Gamma_\phi}\nabla\widetilde{\phi}) + \nabla \cdot J_t^R + \nabla.J_t^L + \overline{S_\phi} \tag{1.57}$$

Now, J_t^R and J_t^L that require closure models, can be merged into one single subgrid-scale diffusive flux, $J_t = J_t^L + J_t^R$. Then the filtered scalar equation will become

$$\frac{\partial \overline{\rho}\widetilde{\phi}}{\partial t} = \nabla \cdot \overline{\rho}\widetilde{\phi}\widetilde{\mathbf{u}} + \nabla.(\overline{\rho}\overline{\Gamma_\phi}\nabla\widetilde{\phi}) + \nabla \cdot J_t + \overline{S_\phi} \tag{1.58}$$

And now this new turbulent diffusive flux needs a closure model which will be discussed in the next section.

1.2.1.1 Subgrid Scalar Closure

By using gradient diffusion hypothesis, the SGS diffusive flux can be rewritten as:

$$J_t = \overline{\rho}\Gamma_{\phi_t}\nabla\widetilde{\phi} \tag{1.59}$$

Where Γ_{ϕ_t} is turbulent diffusivity constant that can be calculated from the turbulent Schmidt number which is defined as:

$$Sc_t = \frac{\mu_t}{\rho\Gamma_{\phi_t}} = \frac{\nu_t}{\Gamma_{\phi_t}} \tag{1.60}$$

With this definition, after specifying the turbulent Schmidt number of the problem and getting the turbulent viscosity, μ_t, from one of our closure models for momentum equation, the turbulent diffusivity constant will be

$$\Gamma_{\phi_t} = \frac{\rho S c_t}{\mu_t} \tag{1.61}$$

Finally, the filtered scalar transport equation will be reformed to the following equation:

$$\frac{\partial \overline{\rho}\widetilde{\phi}}{\partial t} = \nabla \cdot \overline{\rho}\widetilde{\phi}\widetilde{\mathbf{u}} + \nabla.(\overline{\rho}(\overline{\Gamma_\phi} + \Gamma_{\phi_t})\nabla\widetilde{\phi}) + \overline{S_\phi} \tag{1.62}$$

1.3 Population Balances

Wasatch currently supports a basic population balance equation with a single internal coordinate. The targeted equation is written as

$$\frac{\partial N}{\partial t} = -\nabla \cdot \mathbf{u}N + \frac{\partial gN}{\partial r} + b; \quad N \equiv N(\mathbf{x}, r, t), \tag{1.63}$$

where r is an internal coordinate, $g \equiv \frac{\mathrm{d}r}{\mathrm{d}t}$ denotes the growth rate of r, and b represents the death/birth rates of particles. To make further headway, solution of the population balance equation is usually achieved by solving for the moments of the number density function, N. The k^{th} moment is defined as

$$m_k \equiv \int_{-\infty}^{+\infty} r^k N(\mathbf{x}, r, t)\,\mathrm{d}r. \tag{1.64}$$

By applying the moment transformation to 1.63, we recover the moment transport equation

$$\frac{\partial m_k}{\partial t} = -\nabla \cdot \mathbf{u}m_k - k\int_{-\infty}^{+\infty} r^{k-1} gN\,\mathrm{d}r + \int_{-\infty}^{+\infty} r^k b\,\mathrm{d}r \equiv F(m_{k+i}); \quad i \in \mathbb{Q}. \tag{1.65}$$

The right-hand-side (RHS) in Eq. (1.65) may be a function of any moment order. As such, we can write additional transport equations for these moments. However, one usually ends up with a several moments that require closure. This may be achieve by the quadrature method of moments.

The quadrature method of moments (QMOM) is a closure technique used to solve the moment transport equations for population balance problems with a single internal coordinate. The technique is based on Gaussian quadrature to achieve closure for the unknown moments. Subsequently, the quadrature nodes and abscissae are obtained using the lower order moments by using the product-difference algorithm. The product-difference algorithm incurs a series of recursion formulas that result in an eigenvalue problem whose solution yields the abscissae (eigenvalues) and weights (eigenvectors).

We recently implemented the moment transport equation in Wasatch with the flexibility to solve for any number of moments. We also added three growth rate models; namely, constant growth, mono-surface nucleation, and bulk diffusion to illustrate the ease with which new growth models can be implemented.

Chapter 2

Numerics

Wasatch uses finite volume discretization on a staggered-grid arrangement. All scalars are stored at cell centers including the pressure, while the x, y, and z momentum and velocity components are stored at staggered locations. Fluxes are stored at the faces of their corresponding fields.

2.1 Spatial Discretization

2.1.1 Flux Limiters

2.2 Time Integration

Wasatch currently entertains both forward Euler (FE) and third-order Rungue-Kutta strong stability preserving (RK3SSP) time integrators.

2.2.1 Stable Timestep Selection

We use a simple stability constraint for timestep selection based on Von-Neumann stability analysis of convective and diffusive equations. Our stability constraint boils down to

$$\frac{|u|\,\Delta t}{\Delta x} + \frac{|v|\,\Delta t}{\Delta y} + \frac{|w|\,\Delta t}{\Delta z} + \frac{\nu \Delta t}{\Delta x^2} + \frac{\nu \Delta t}{\Delta y^2} + \frac{\nu \Delta t}{\Delta z^2} \leq C_{\max} \tag{2.1}$$

or

$$\Delta t \leq \frac{C_{\max}}{S}; \quad S \equiv \frac{|u|}{\Delta x} + \frac{|v|}{\Delta y} + \frac{|w|}{\Delta z} + \frac{\nu}{\Delta x^2} + \frac{\nu}{\Delta y^2} + \frac{\nu}{\Delta z^2} \tag{2.2}$$

where u, v, and w are the velocity components, ν is the kinematic viscosity, and Δx, Δy, and Δz are the mesh spacings in the x, y, and z directions, respectively. The value of $C_{\max}$ is specified by the user through the input file (see: timestep_Multiplier) and is typically chosen to be 0.5.

Chapter 3

Order of Accuracy Verification

3.1 Introduction

Order-of-accuracy analysis is based on the concept that the error must be reduced as spatial and temporal resolutions are increased. By comparing the errors at different grid and time resolutions, one can estimate the theoretical order of accuracy for a code. We first review the general concept of analyzing the order of accuracy of a code and follow that with the accuracy verification in Wasatch.

The order of accuracy of a discretization method is defined as *the exponent of the leading order term in the total error* incurred by the discretization process (Oberkampf and Roy, 2010). The total error refers to the accumulation of truncation and round off errors throughout the spatial and temporal domains. It can be generally defined as

$$E = |\phi_{\text{exact}} - \phi_{\text{numerical}}|, \tag{3.1}$$

where ϕ_{exact} and $\phi_{\text{numerical}}$ refer to the exact and numerical solutions, respectively.

Round-off errors are those due to finite precision calculation on a computer while the truncation error represents the error induced by neglecting terms in a finite numerical approximation. For most practical purposes, one can neglect round-off errors and focus on how the truncation error is propagated throughout the solution domain. This is the strategy that we will adopt in this document.

To illustrate how the truncation error is estimated, we make use of Taylor series expansions to approximate the various differential terms in an equation. For example, for a transported scalar ϕ on a grid with uniform spacing Δx, the Taylor series at an arbitrary point i is written as

$$\phi_{i+1} = \phi_i + \frac{\mathrm{d}\phi_i}{\mathrm{d}x}\Delta x + \frac{\mathrm{d}^2\phi_i}{\mathrm{d}x^2}\frac{\Delta x^2}{2!} + \frac{\mathrm{d}^3\phi_i}{\mathrm{d}x^3}\frac{\Delta x^3}{3!} + \ldots. \tag{3.2}$$

One can then approximate the first derivative at grid point i via

$$\frac{\mathrm{d}\phi_i}{\mathrm{d}x} = \frac{\phi_{i+1} - \phi_i}{\Delta x} + \underbrace{\frac{\mathrm{d}^2\phi_i}{\mathrm{d}x^2}\frac{\Delta x}{2!} + \frac{\mathrm{d}^3\phi_i}{\mathrm{d}x^3}\frac{\Delta x^2}{3!} + \ldots.}_{\text{truncation error}} \tag{3.3}$$

Then, one can choose to approximate the first derivative by using the first term on the right-hand-side of the previous equation. The remaining terms are then known as the truncation error.

The truncation error has to satisfy certain properties for a discretization scheme to be consistent. For example, all terms in the truncation error must decrease monotonically. Furthermore, the coefficients in the truncation error must be bounded.

The truncation error is usually lumped into a single term and assigned the O terminology

$$T.E. = \underset{\Delta x \to 0}{O}(\Delta x^m), \quad m \in \mathbb{R}^+. \tag{3.4}$$

This definition has a precise meaning: as $\Delta x \to 0$, the truncation error is proportional to Δx^p with a proportionality constant that is independent of Δx. Alternatively, one can write

$$\lim_{\Delta x \to 0} \frac{T.E.}{\Delta x^m} = M, \tag{3.5}$$

where M is a constant. Then, the truncation error can be written as

$$T.E. = M\Delta x^m. \tag{3.6}$$

Subsequently, and by neglecting round-off errors, the total error consists of the accumulated truncation error when discretization is carried out throughout the spatial or temporal domain. In general, this is not the sum of the truncation error incurred by applying a discretization method at a single point. We will discuss how the truncation error accumulates shortly, but for the moment, the total error can be written as

$$E = \alpha \Delta x^p, \tag{3.7}$$

where α is a real constant that is independent of the mesh size while p is a positive real number that designates the order-of-accuracy of a discretization method. The purpose of an order-of-accuracy verification is to determine the exponent p in the total error induced by the discretization process.

3.2 Common Approaches for Order-of-Accuracy Calculation

There are two key approaches to estimating the order of accuracy of a given implementation. The first requires an exact solution and requires at least two mesh refinements while the second is based on Richardson extrapolation and requires at least three mesh refinements. The latter, however, does not require an analytical solution to estimate the truncation error.

3.2.1 Using an Exact Solution

When an exact solution is available, the error is readily available as shown in 3.1. Using 3.7, there are two unknowns (α and p) and therefore one needs two equations to solve for p (and α). With knowledge of an exact solution, a test problem can be setup and solved on two different grid spacings, Δx_1 and $\Delta x_2 = \frac{\Delta x_1}{r_x}$. Then, one calculates the errors associated with these two grids. These are given by

$$\begin{cases} E_1 = & \alpha \Delta x_1^p \\ E_2 = & \alpha \Delta x_2^p \end{cases}. \tag{3.8}$$

The order p is readily calculated by taking the ratio of these two equations

$$\frac{E_1}{E_2} = \left(\frac{\Delta x_1}{\Delta x_2}\right)^p = r_x^p. \tag{3.9}$$

Finally, by taking the logarithm of the former equation, the observed order of accuracy is at hand

$$p = \frac{\ln \frac{E_1}{E_2}}{\ln r_x}. \tag{3.10}$$

This method is usually well suited for ODEs or for PDEs when one is after the dominant error source. The method can be extended for combined order analysis but requires the solution of a nonlinear system of equations and at least seven mesh refinement levels (see Oberkampf and Roy, 2010). A more convenient method uses Richardson's extrapolation and does not require an analytical solution.

3.2.2 Using Richardson's Extrapolation

When an analytical solution is not available, or when one is dealing with PDEs where the effects of spatial and temporal errors compound, Richardson's extrapolation provides a convenient way of calculating the order of accuracy. Recall that the order of accuracy of a given discretization method determines the rate of convergence of the method. Hence, in principle, one does not need an analytical solution to analyze the rate of convergence. The key disadvantage of this approach is that one cannot assess the validity of the numerical solution. However, such verification may be carried out by comparing to experimental data or nuermical solutions obtained via other codes.

Given three numerical solutions that were obtained with systematic mesh refinement, the order of a method can be determined via

$$p = \ln\left(\frac{f_{i+2} - f_{i+1}}{f_{i+1} - f_i}\right)\frac{1}{\ln \frac{1}{r}}. \tag{3.11}$$

3.2.3 Combined Spatio-Temporal Order-of-Accuracy Analysis

Without loss of generality, the truncation error may be represented as

$$E = u_{\text{exact}} - u_{\text{numerical}} = \alpha \Delta t^p + \beta \Delta x^q + \gamma \Delta y^r + \delta \Delta z^s. \tag{3.12}$$

This excludes all terms that have mixed orders in them.

3.3 Order-of-Accuracy Verification in Wasatch

3.3.1 Temporal Order of Accuracy

Wasatch currently entertains both forward Euler (FE) and third-order Rungue-Kutta strong stability preserving (RK3SSP) time integrators. To verify the temporal order of accuracy of these integrators, an ODE test was designed. In the first, we solve an ODE with a time dependent source term.

Table 3.1: Observed temporal order of accuracy using $\frac{d\phi}{dt} = \sin t$. Results are taken at $t = 1$s with systematic timestep refinement.

	Forward Euler		RK3SSP	
Δt	$E = \lvert\Phi - \phi\rvert$	Observed Order	$E = \lvert\Phi - \phi\rvert$	Observed Order
0.1	4.245×10^{-1}		1.596×10^{-8}	
0.05	2.113×10^{-1}	**1.006**	9.976×10^{-10}	**4.000**
0.025	1.054×10^{-1}	**1.003**	6.623×10^{-11}	**4.000**
0.0125	5.265×10^{-2}	**1.001**	3.900×10^{-12}	**3.998**
0.00625	2.631×10^{-2}	**1.000**	2.508×10^{-13}	**3.959**
0.003125	1.315×10^{-2}	**1.000**	3.175×10^{-14}	**2.984**

Care must be taken when calculating the order of accuracy at the first time-step. In this scenario, the observed order of accuracy will be one order higher than the theoretical one. This can be shown as follows. Consider a time-stepping discretization of order p. This solves the following ODE

$$\frac{d\phi}{dt} = L(\phi). \tag{3.13}$$

ODE with Time-Dependent Source Term Our choice falls upon

$$\frac{d\phi}{dt} = \sin t; \quad \phi(0) = -1. \tag{3.14}$$

The analytical solution for this case is

$$\phi_{\text{exact}} = -\cos t. \tag{3.15}$$

The tests varied over a time interval of 1 second with systematic mesh refinement. The results for both the FE and the RK3SSP are shown in the table below.

ODE with Implicit Source Term In this verification test, we solve the following ODE

$$\frac{d\phi}{dt} = \phi; \quad \phi(0) = 1. \tag{3.16}$$

The exact solution is given by

$$\Phi = e^t \tag{3.17}$$

Being a time-dependent ODE with no spatial variation, the mesh resolution is set to [2, 2, 2] in Wasatch to reduce computational time. The code is run with both FE and RK3 time integrators with 6 refinement levels (or a total of 7 tests) starting with $\Delta t_0 = 0.1$s with a refinement ratio of $r_t = 2$ (recall that $r_t \equiv \frac{\Delta t_i}{\Delta t_{i+1}}$ where Δt_i and Δt_{i+1} correspond to the coarse and fine meshes, respectively).

Pressure Projection with Taylor-Green Vortex

Table 3.2: Observed temporal order of accuracy using $\frac{d\phi}{dt} = \phi$. Results are taken at $t = 2$s with systematic timestep refinement.

	Forward Euler		RK3SSP	
Δt	$E = \lvert\Phi - \phi\rvert$	Observed Order	$E = \lvert\Phi - \phi\rvert$	Observed Order
0.1	6.615×10^{-1}		5.684×10^{-4}	
0.05	3.491×10^{-1}	**0.922**	7.395×10^{-5}	**2.942**
0.025	1.795×10^{-1}	**0.959**	9.431×10^{-6}	**2.971**
0.0125	9.103×10^{-2}	**0.979**	1.191×10^{-6}	**2.985**
0.00625	4.584×10^{-2}	**0.989**	1.496×10^{-7}	**2.992**
0.003125	2.301×10^{-3}	**0.994**	1.874×10^{-8}	**2.996**
0.0015625	1.152×10^{-3}	**0.977**	2.346×10^{-9}	**2.997**

3.3.2 Spatial Order of Accuracy

A method of manufactured solution has been done on wasatch using forward Euler method on a Taylor-Vortex problem to check the spatial and temporal order of accuracy. We state the spatial order of accuracy here as long as the temporal order of accuracy has been already shown in this document in previous subsection.

In order to check the spatial order of accuracy we looked at the four different grid resolutions for our domain, and we chose 0.001, 0.0005, 0.00025 and 0.000125 respectively for their time step. However, we just looked at the results after the first time step. This is in order to neglecting the temporal accuracy in the explicit solvers. Because, those terms will be calculated in the previous time step, therefore, in all of our cases they will be calculated at time zero. So, because of using the same time for these terms we don't let the time accuracy to influence our results for grid convergence study

Taylor-Vortex problem is a 2D problem where the exact solutions are:

$$u(x,y,t) = 1 - A\cos(x-t)\sin(y-t)\exp(-2\nu t), \tag{3.18}$$

$$v(x,y,t) = 1 + A\sin(x-t)\cos(y-t)\exp(-2\nu t). \tag{3.19}$$

Where in this study, $A = 4$ and $\nu = 0.1$.

In order to calculate the error the norm over the whole area is taken to obtain

$$E = \|u_{exact} - u_{numerical}\|. \tag{3.20}$$

So we can consider all of the point on the grid together.

Results for Spatial order of accuracy The results for both u and v using the FE are shown in the table below

As it is clear from the table the spatial order of accuracy is 2 as it was expected for the FE method with central 3 point discretization in space.

Table 3.3: Observed spatial order of accuracy using Taylor-Vortex. Results are taken at $t = 2$s with systematic time-step refinement.

	u		v	
Δx	$E = norm(U-u)$	Observed Order	$E = norm(U-u)$	Observed Order
0.1257	1.856×10^{-3}		1.856×10^{-3}	
0.0628	4.650×10^{-4}	**1.997**	4.650×10^{-4}	**1.997**
0.0314	1.163×10^{-4}	**1.999**	1.163×10^{-4}	**1.999**
0.0157	2.908×10^{-5}	**2.000**	2.908×10^{-5}	**2.000**

Chapter 4

Scalability

To test some scalability aspects of Wasatch, we implemented the model discussed in Eq. (**??**). [DEVIN, can you add some results here?]

Chapter 5

Boundary Conditions

5.1 Overview of Defining Boundary Conditions

Wasatch provides the machinery to apply Dirichlet and Neumann boundary conditions. Our approach is based on setting boundary conditions on ghost cells at the boundaries of the computational domain to reproduce the desired boundary at the face. This is illustrated in Fig. 5.1.

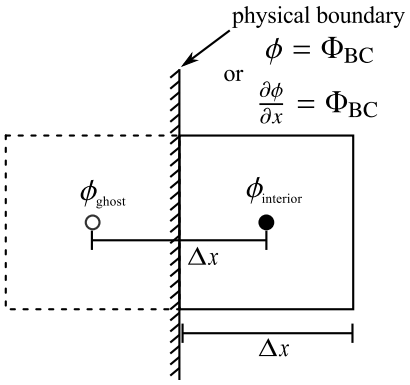


Figure 5.1: Boundary condition specification in Wasatch. Here, Φ_{BC} designates the desired value of the scalar or its flux at the boundary, $\phi_{interior}$ is the interior value of the scalar, and ϕ_{ghost} is the ghost value to be set.

Mathematically, this may be written as follows. Consider a scalar ϕ for which a boundary condition is to be specified. The ghost value of ϕ may then be written as

$$\phi_{ghost} = \alpha \phi_{interior} + \beta \Phi_{BC}, \tag{5.1}$$

where α and β are real coefficients that depend on the type of boundary condition specified. These are summarized in Table 5.1 .

For Staggered fields on uniform structured grids, a simplification follows for setting Dirichlet conditions. In this case, certain boundary faces coincide with staggered control volumes and the desired Dirichlet condition may be set directly on those cells.

	α	β
Dirichlet	0.5	0.5
Neumann	1.0	$-\Delta x$

Table 5.1: Summary of coefficients for setting values at ghost cells to reproduce a desired boundary condition at a physical boundary.

5.2 Scalar Transport Boundary Conditions

The treatment of scalar boundary conditions follows from the previous discussion. Specification of either fixed values or fluxes of transported scalars is easily achieved using the machinery discussed in §5.1.

5.3 Fluid Flow Boundary Conditions

5.3.1 Inlets, Walls, and Moving Walls

Inlets and walls, whether moving or stationary, share the same boundary condition specification. The user only specifies the momentum at those boundaries,

$$\rho \mathbf{u} = \rho(U, V, W). \tag{5.2}$$

Treatment of the pressure is derived from the pressure Poisson equation as follows. For simplicity, we illustrate this in one-dimension at a boundary cell on the x-minus side of the computational domain as shown in 5.2. In this case, the pressure poisson equation is written as

$$\frac{\partial^2 p}{\partial x^2} = \frac{\partial F}{\partial x}. \tag{5.3}$$

Using second order finite differencing, Eq. (5.3) may be written as

$$\frac{p_{-1} - 2p_0 + p_1}{\Delta x^2} = \frac{F_{\frac{1}{2}} - F_{-\frac{1}{2}}}{\Delta x}. \tag{5.4}$$

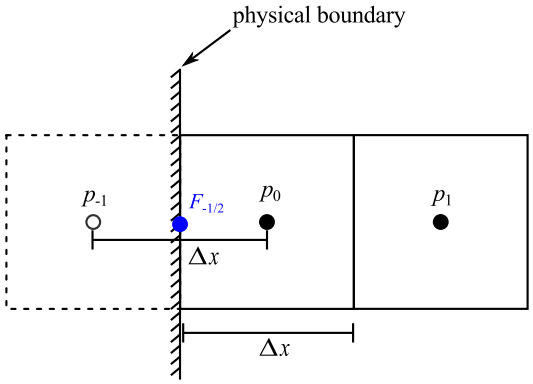


Figure 5.2: Derivation of pressure boundary conditions.

Here, both p_{-1} and $F_{i-\frac{1}{2}}$ are unknown and one may be inclined to specify these quantities at the boundary. However, by using the x-momentum equation, one can relate these quantities by projecting and evaluating the momentum equations in the direction normal to the boundary. In our example, on an x-minus face, we evaluate the x-momentum equation at the x-minus face. This yields

$$\left(\frac{\partial \rho u}{\partial t}\right)_{-\frac{1}{2}} = -\left(\frac{\partial p}{\partial x}\right)_{-\frac{1}{2}} - F_{-\frac{1}{2}}. \tag{5.5}$$

Again, using central finite differencing, one recovers

$$\left(\frac{\partial \rho u}{\partial t}\right)_{-\frac{1}{2}} = -\frac{p_0 - p_{-1}}{\Delta x} - F_{i-\frac{1}{2}} \tag{5.6}$$

or

$$\frac{p_{-1} - p_0}{\Delta x} = \left(\frac{\partial \rho u}{\partial t}\right)_{-\frac{1}{2}} + F_{-\frac{1}{2}}. \tag{5.7}$$

The force balance in 5.7 states that the pressure gradient is balanced by accumulation of momentum and convective and diffusive fluxes - a restatement of Newton's second law. One then substitutes 5.7 into 5.4. This operation returns the following Poisson equation at the boundary

$$\frac{\left(\frac{\partial \rho u}{\partial t}\right)_{-\frac{1}{2}}}{\Delta x} + \frac{F_{-\frac{1}{2}}}{\Delta x} + \frac{-p_0 + p_1}{\Delta x^2} = \frac{F_{\frac{1}{2}} - F_{-\frac{1}{2}}}{\Delta x}. \tag{5.8}$$

Notice how the diffusive and convective fluxes cancel out and one is left with

$$\frac{-p_0 + p_1}{\Delta x^2} = \frac{F_{\frac{1}{2}}}{\Delta x} - \frac{\left(\frac{\partial \rho u}{\partial t}\right)_{-\frac{1}{2}}}{\Delta x}. \tag{5.9}$$

According to 5.9, both the pressure-coefficient-matrix and the pressure right-hand-side must be modified. Because $F_{-\frac{1}{2}}$ never shows up in 5.9, we simply set it to zero in the Wasatch input file while also setting the coefficient for p_{-1} to 0 (the code automatically takes care of that).

5.3.2 Pressure Outlet

The designation of a pressure outlet has a different meaning for different CFD schools of thought. In Wasatch, a pressure outlet is simply an artificial location in space where one does not have information about the velocity or momentum profile. In practice, however, one may have knowledge of the pressure at such boundaries.

The user must specify a value for the outlet pressure. This value may be specified at the extra-cell or on the domain boundary face. Both options give stable results. In Wasatch, we implement the latter option.

Treatment of the pressure follows the same guidelines exposed in the previous section. In this case, however, the pressure p_{-1} may be related to $P_{-\frac{1}{2}}$ on the boundary face. Given that the pressure lives on scalar cell centers, using second order approximation, we have

$$P_{-\frac{1}{2}} = \frac{p_{-1} + p_0}{2} \tag{5.10}$$

or

$$p_{-1} = 2P_{-\frac{1}{2}} - p_0. \tag{5.11}$$

Then, the pressure poisson equation (evaluated at the boundary) returns

$$\frac{-3p_0 + p_1}{\Delta x^2} = -2P_{-\frac{1}{2}} + \frac{F_{\frac{1}{2}} - F_{-\frac{1}{2}}}{\Delta x}. \tag{5.12}$$

The momentum partial right-hand-side remains unknown in the previous equation. One can choose either of the following conditions

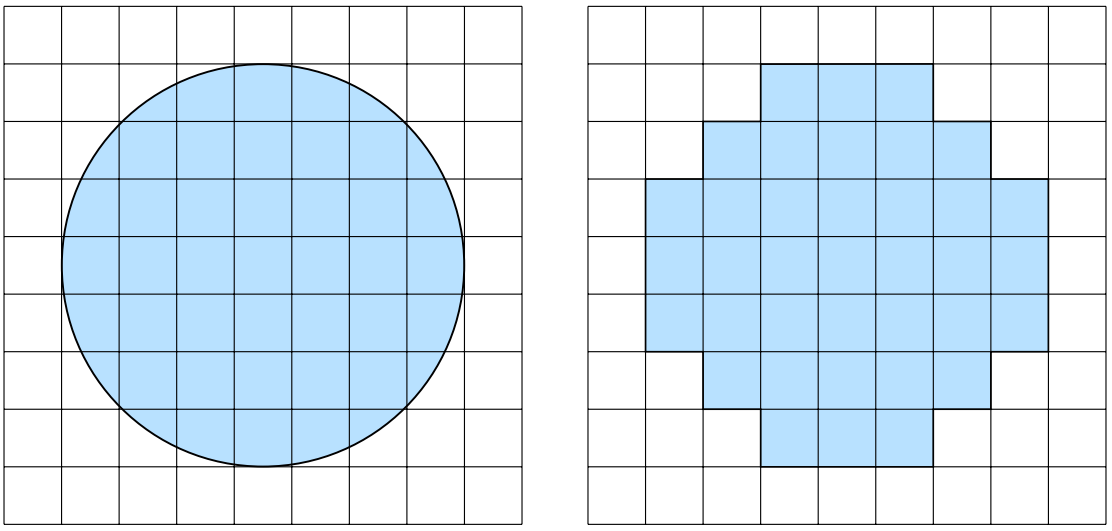
$$\begin{cases} F_{-\frac{1}{2}} = 0 \\ \frac{\partial F}{\partial x} = 0 \end{cases}. \tag{5.13}$$

The former may be interpreted to imply a balance between diffusive and convective fluxes at the pressure outlet boundary. It may also be interpreted as a zero flux condition for convective and diffusive fluxes. The latter is self evident where one is assuming a self-similar profile for convective and diffusive fluxes.

Chapter 6

Complex Geometry

Wasatch can, in principle, handle geometries of arbitrary complexity by using a discrete stairstepping procedure. This approach may be thought of as a simplified version of the cut-cell method where only integer volume (and area) fractions are allowed. A volume fraction field is defined as the amount of fluid that fills a given control volume while the area-fraction corresponds to the fraction of the cell-boundary at a fluid-solid interface.



(a) Embedded cylinder using cut-cells.

(b) Embedded cylinder using stairstepping.

Figure 6.1: Comparison of cut-cells and stairstepping for embedded geometries.

6.1 Rational Formulation

To handle embedded geometries, Wasatch first calculates four scalar fields, φ_v, φ_x, φ_y, and φ_z. These correspond to the cell-centered and x, y, z-staggered volume fractions, respectively. The staggered volume fractions are nothing more than the area-fractions. We call them staggered volume fractions, however, because of the nature of stairstpping and the fact that area fractions have integer values in this case and hence coincide with staggered volumes on structured, uniform grids. These are shown in figure 1 below where the various volume fractions are depicted along with their associated control volumes.

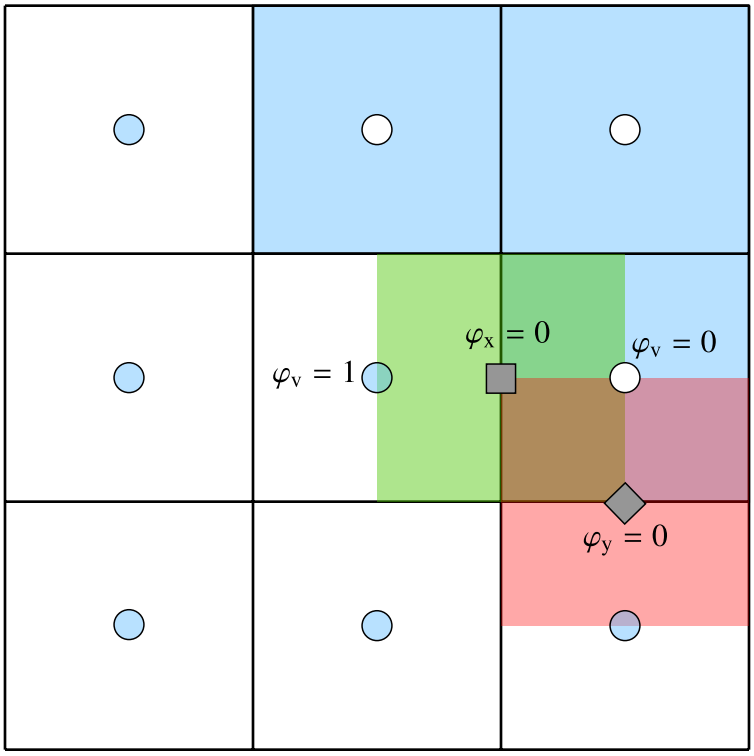


Figure 6.2: Cell-centered and staggered volume fractions associated with a solid intrusion.

6.1.1 Momentum Transport

In order for the solid intrusions to affect the fluid dynamics, the surface (convection, diffusion) and volume (pressure) forces must be set appropriately at cell boundaries. In general, one must multiply surface forces by the area fractions while volume forces get multiplied by the cell-centered volume fraction. In Wasatch, this may be easily accomplished by multiplying the momentum partial-right-hand-side expressions by their corresponding area fractions, as follows

$$\frac{\partial \rho \mathbf{u}}{\partial t} = \boldsymbol{\varphi} \cdot \mathbf{F} - \varphi_v \nabla p; \quad \mathbf{F} \equiv -\nabla \cdot \rho \mathbf{u}\mathbf{u} - \nabla \cdot \boldsymbol{\tau}_{ij}; \quad \boldsymbol{\varphi} \equiv (\varphi_x, \varphi_y, \varphi_z) \tag{6.1}$$

The pressure poisson equation is also modified to account for the presence of the solids. We assume that the pressure is uniformly known inside the solid (e.g. $p = 0$) and modify the matrix

accordingly. For a cell inside a solid, we set all matrix coefficients to zero except the center coefficient which is set to 1

$$a.w = a.e = a.n = a.s = a.t = a.b = 0, \quad a.p = 1 \tag{6.2}$$

For a fluid cell neighboring a solid, treatment is similar to that at a wall. For illustration purposes, we give the following example at a corner cell.

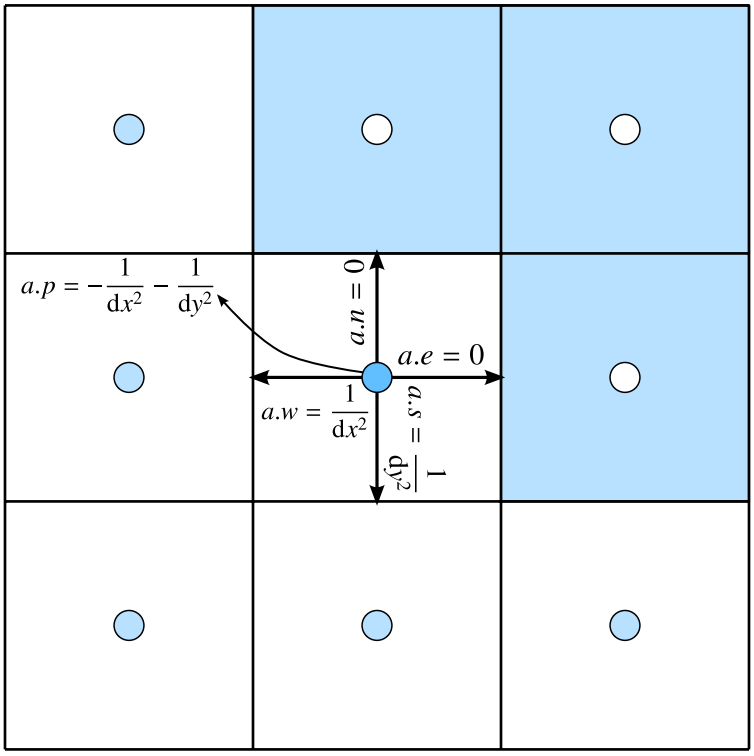


Figure 6.3: Pressure matrix coefficients near embedded boundaries.

6.2 Input Specification

Wasatch provides a simple interface for specifying intrusions as shown in Listing (6.1). Geometry, which is represented by volume and area fractions, can be specified via three mechanisms:

1. Uintah's geometry objects: These include all of Uintah's geometry pieces as well as geometry operations such as unions, intersections, differences, etc... Please refer to Uintah's UPS specification for a list of supported geometry objects.

2. Wasatch's geometry expressions: These are useful when designing special geometry objects not handled by Uintah including time-dependent geometry.

3. External volume and area fractions: This is useful when another Uintah component (e.g. Arches) is driving Wasatch and that component calculates area and volume fractions. All one needs in this case is to provide the names of the volume and area fractions calculated by the external component.

```
<EmbeddedGeometry  spec="OPTIONAL NO_DATA"
                   children1="ONE_OF(CHILD Intrusion,
                      GeometryExpression, External)" >
  <Inverted       spec="OPTIONAL NO_DATA"/>
  <MovingGeometry spec="OPTIONAL NO_DATA"/>
  <External       spec="OPTIONAL NO_DATA">
    <SVolFraction     spec="REQUIRED STRING"/>
    <XVolFraction     spec="REQUIRED STRING"/>
    <YVolFraction     spec="REQUIRED STRING"/>
    <ZVolFraction     spec="REQUIRED STRING"/>
  </External>
  <Intrusion      spec="MULTIPLE NO_DATA">
    <geom_object  spec="REQUIRED"/>
  </Intrusion>
  <GeometryExpression            spec="OPTIONAL" />
</EmbeddedGeometry>
```

Listing 6.1: Specification of embedded geometry in Wasatch.

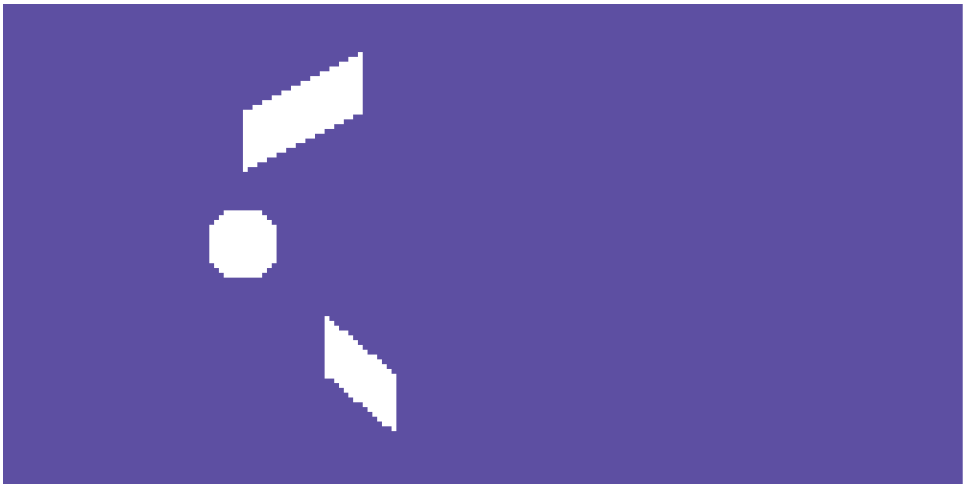
As an illustrative example of using some of Uintah's geometry objects, Listing (6.2) summarizes the embedded geometry specification for the intrusion of two parallelpepipeds and a cylinder in a cross-flow.

```
<EmbeddedGeometry>
 <Intrusion>
    <geom_object>
    <union>
       <cylinder>
          <bottom>[0,0,-1]</bottom>
          <top>[0,0,1]</top>
          <radius>0.15</radius>
       </cylinder>
       <parallelepiped>
         <p1>[0,0.3,1]</p1>
         <p2>[0,0.55,1]</p2>
         <p3>[0.5,0.55,1]</p3>
         <p4>[0,0.3,-1]</p4>
       </parallelepiped>
       <parallelepiped>
         <p1>[0.35, -0.3,  1]</p1>
         <p2>[0.35, -0.55, 1]</p2>
         <p3>[0.65, -0.55, 1]</p3>
         <p4>[0.35, -0.3, -1]</p4>
       </parallelepiped>
```

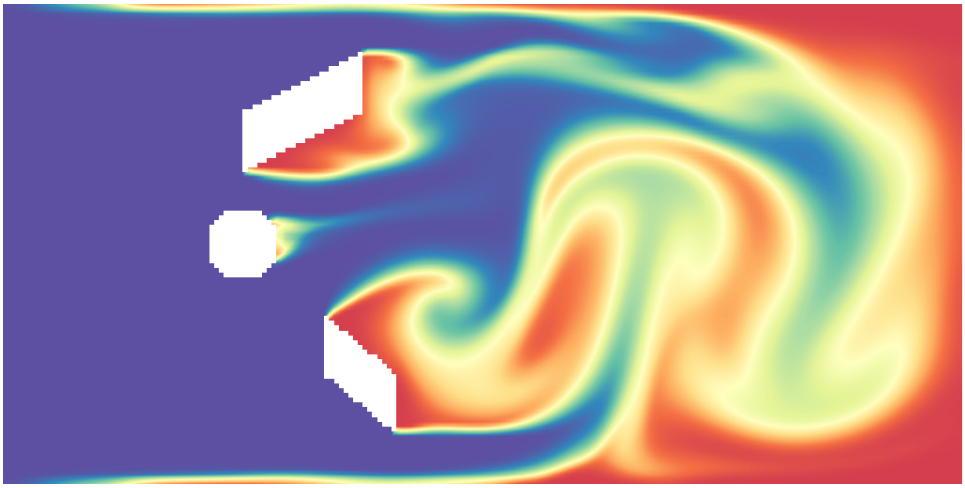
```
    </union>
  </geom_object>
 </Intrusion>
</EmbeddedGeometry>
```

Listing 6.2: Example of embedding multiple geometrical objects in Wasatch.

The specification shown in Listing (6.2) produces the snapshows shown in Fig. 6.4.



(a) Illustration of the embedded geometry shown in Listing (6.2).



(b) Flow over multiple intrusions showing scalar mixing in the wake region of the embedded geometry.

Figure 6.4: Embedded geometry example in Wasatch.

6.3 Limitations

At the time of writing (January 2013), Wasatch embedded boundaries have the following limitations:

- Simple wall boundary conditions: embedded objects are assumed to be simple/inert walls with no scalar or momentum fluxes at their boundaries.
- Embedded geometry is assumed to be stationary.

Future efforts in this arena will address the above limitations. There is currently no interest, however, in moving from stairstepping towards cut-cell methods.

Chapter 7

Modeling General Scalar Transport

Chapter 8

Modeling Species Transport

Chapter 9

Modeling Dispersed-Phase Flows

9.1 Precipitation

Wasatch supports precipitation-physics reactions utilizing the quadrature method of moments to solve the population balance equation. The internal coordinate in this case is the particle radius.

9.1.1 Governing Equations

The precipitation model utilized in Wasatch uses the implemented moment transport equations.

$$\frac{\partial m_k}{\partial t} = -\nabla \cdot \mathbf{u} m_k - k \int_{-\infty}^{+\infty} r^{k-1} g N \, \mathrm{d}r + \int_{-\infty}^{+\infty} r^k b \, \mathrm{d}r \equiv F(m_{k+i}); \quad i \in \mathbb{Q}, \tag{9.1}$$

with QMOM used as closure for the moments. The moments that require closure will change based on the growth rate model that is utilized, and the closure is taken care of automatically in the Wasatch algorithm. The relavent terms dictating the reaction rate are tabulated by the extent of reaction and the mixture fraction. The main term required is the supersaturation value of the liquid. The source term for the reaction progress is

$$S_{\xi} = \frac{1}{\iota} \sum_{\alpha=1}^{n} \nu_{\alpha} S_{3,\alpha}, \tag{9.2}$$

where $S_{3,\alpha}$ is the source term of the third moment, representing the volume, of species α, ν_{α} is the molecular volume, and ι is the maximum moles of precipitate.

9.1.2 Birth Models

Nucleation terms in Wasatch can be specifed as one of three distributions, point, uniform, and normal. Each of these distributions requires the specification of a nucleation radius which can either be set as a constant or as the critical radius $r^* = r_0 / \log(S)$. The uniform and normal distributions also require a specified standard deviation for the distribution. The coefficient used for the birth term can either be set as constant or using the classical nucleation definition

$$B_0 = \exp\left[\tfrac{16\pi}{3} (\tfrac{\gamma}{K_B T})^3 \frac{\nu}{N_{\mathrm{A}} \log(S)^2} \right], \tag{9.3}$$

where γ is the surface energy, K_B is the Boltzman constant, T is the temperature in Kelvin, ν is the molecular volume, and N_A is Avogadro's constant. The Wasatch formulation also includes a "pre-Birth coefficient" which is used as a converison factor from SI units into smaller lengths (*i.e.* μm or nm) used as precipitate measurements.

9.1.3 Growth Models

Currently Wasatch has three growth models implemented; constant growth, bulk diffusion and monosurface nucleation. Each model uses a different form of $g(r)$ with $g(r) = g_0$, $g(r) = g_0\frac{1}{r}$, and $g(r) = g_0 r^2$, respectively, as well as a different requirement for the closure of the moment transport equations as: none, m_{-2} & m_{-1}, and m_{k+1}. The bulk diffusion model uses the appropriate coefficient as

$$g_0 = \nu D C_{eq}(S - \bar{S}), \quad g(r) = 1/r \tag{9.4}$$

with D as the diffusion coefficient, C_{eq} as the equilibirum concentration and where $\bar{S}$ can either be set as one, or set as needed for Ostwald ripening as

$$\bar{S} = \exp(2\nu\gamma/RTr) \tag{9.5}$$

For Ostwald ripening, the r appearing in the exponential does not allow for this term to be converted into a moment of the problem, therefore the quadrature weights and abscissae are used to evaluate the Ostwald ripening term. A similar model is also available for bulk diffusion growth on cylindrical particles as

$$g_0 = \frac{7\nu D C_{\mathrm{eq}}(\bar{S} - S)}{6\log(1/2)}, \quad g(r) = 1/r \tag{9.6}$$

For monosurface nucleation the growth coefficient is

$$g_0 = \beta_A D d^3 exp(-\Delta G/K_B T), \quad g(r) = r^2 \tag{9.7}$$

with

$$\Delta G = \frac{\beta_L \gamma^2 d^2}{4\beta_A K_B T \log(S)}, \tag{9.8}$$

where β_L and β_A are shape factors, and d is the molecular diameter. As with the birth models, a "pre-Growth coeffcient" can be used for unit conversions. A kintically limted model can be used that has no size dependence of the particles, this is essentially the constant model with supersaturation dependence.

$$g_0 = K_A(S - \bar{S}), \quad g(r) = 1 \tag{9.9}$$

Here K_A is a kinetic growth coefficient to be determined from fitting experimental data.

9.1.4 Aggregation Models

To account for aggregation in particulate systems, two terms must be considered the death kernel from two particles combining alogn with the birth kernel from the new particle. The general form

for each of these is in volume formulation

$$\frac{1}{2}\int_0^v \beta(v-\epsilon,\epsilon)n'(v-\epsilon;t)n'(\epsilon;t)d\epsilon \tag{9.10}$$

$$-n'(v;t)\int_0^\infty \beta'(v,\epsilon)n'(\epsilon;t)d\epsilon \tag{9.11}$$

Converting the population balance into a radius formulation, and applying a moment transform, using quadrature on the integrals yields

$$\frac{1}{2}\sum_i w_i \sum_j w_j (r_i^3 + r_j^3)^{k/3}\beta_{ij} \tag{9.12}$$

$$-\sum_i r_i^k w_i \sum \beta_{ij} w_j \tag{9.13}$$

where β_{ij} is a specified frequency kernel, and r_i & w_i are the abscissae and weights. The Wasatch formulation spilts the frequency term into two, the size dependent frequncy and the fluid property frequency. The fluid property frequency can then be factored outside of the summation. For aggregation due to Brownian motion

$$\beta_{ij} = \frac{2k_B T}{3\rho} \times \frac{(r_i + r_j)^2}{r_i r_j}, \tag{9.14}$$

where k_B is the Boltzmann cosntant, T is the temperature, and ρ is the fluid density. For aggregation due to turbulent shear forces the frequency kernel is

$$\beta_{ij} = \frac{4}{3}\left(\frac{3\pi}{10}\right)^{1/2}\left(\frac{\epsilon}{\nu}\right)^{1/2} \times \left(r_i + r_j^3\right), \tag{9.15}$$

where ϵ is the energy dissipation and ν is the kinematic viscosity of the fluid. In addition a simple constant frequency is included, $B_{ij} = 1$.

9.1.5 Multi Environment Mixing Model

Wasatch has a subgrid scale model for mixing for the precipitation flows. Since the source terms of the moment transport equations are highly dependent on the current value of the moments, which are solved on the slow scale, a tabulation of data for use in a continuous PDF (i.e. such as a flamelet) is not a viable option. Instead a discrete PDF is used consisting of three delta functions, one at mixture fraction $z = 0$, one at mixture fraction $z = 1$, and the third at the average mixture fraction. In precipitation flows, no reaction occurs at either mixture fraction 0 or 1, so only the source terms from the central delta function need to be solved.

Wasatch requires the averaged mixture fraction and the scalar variance to be solved on the grid. Then the weights of the three delta functions are calculated as

$$w_1 = \frac{\left\langle z'^2 \right\rangle}{\langle z \rangle}, \tag{9.16}$$

$$w_2 = \frac{\left\langle z'^2 \right\rangle - \langle z \rangle + \langle z \rangle^2}{\langle z \rangle^2 - \langle z \rangle}, \tag{9.17}$$

$$w_3 = \frac{-\left\langle z'^2 \right\rangle}{\langle z \rangle - 1}, \tag{9.18}$$

where $\langle z \rangle$ is the average mixture fraction, and $\left\langle z'^2 \right\rangle$ is the scalar variance.

When the Multi Environment Model is enabled, the moment equations are only solved for the second environment with the moments from environments 1 and 3 held constant. The averaged moments are then post-processed based on the values of the environment weights. In addition the transport of the second environment requires a new source term to account for the change in the moments due to mixing

$$S_{\phi_{\alpha,2},\, mix} = -\frac{dw_1/dt}{w_2}\left(\phi_{\alpha,1} - \phi_{\alpha,2}\right) - \frac{dw_3/dt}{w_2}\left(\phi_{\alpha,3} - \phi_{\alpha,2}\right), \tag{9.19}$$

where $\phi_{\alpha,i}$ represents the αth precipitate moment located in the ith mixing environment. With the time derivatives of the weights of the mixing environments calculated as

$$\frac{dw_1}{dt} = \frac{-\left\langle \epsilon_\phi \right\rangle}{\langle z \rangle}, \tag{9.20}$$

$$\frac{dw_3}{dt} = \frac{-\left\langle \epsilon_\phi \right\rangle}{1 - \langle z \rangle}, \tag{9.21}$$

where $\left\langle \epsilon_\phi \right\rangle$ is the scalar dissipation rate.

9.1.6 Viscosity Model

Wasatch has one model to account for the viscosity change of the fluid as particles precipitate. The two parameters dictating the viscosity are the volume fraction of particles and the strain tensor magnitude. The volume fraction is calculated as

$$\phi = \sum_i^n \frac{4\pi}{3}\left(\frac{m_{1,i}}{m_{0,i}}\right)^3 m_{0,i}, \tag{9.22}$$

where $m_{1,i}$ and $m_{0,i}$ are the first and zeroth moments of the ith polymorph phase. In flows with low volume fractions of particles the relative viscosity can be solved for using kinetic theory with ideal assumptions as

$$\mu_r = \frac{\mu}{\mu_0} = 1 + 2.5\lambda\phi, \tag{9.23}$$

with λ as a correction factor for non-idealities. In some case particle flows will have a shear thinning phenomena in which case the viscosity can be expressed as a power law with relation to the strain rate. The model used by Wasatch combines this with the linear fit of the kinetic theory assumptions so that

$$\mu_r = (1 + 2.5\lambda\phi)\,|S|^n\,, \tag{9.24}$$

where $|S|$ is the strain tensor magnitude. The parameters n and λ should then be fit to experimental data. If no shear thinning is expected, n can be set as 0. This formulation should only be used for low volume fractions of particles.

Chapter 10

Verification and Benchmark Studies

10.1 Constant Density Lid-Driven-Cavity

This case aims at verifying the results obtained by the Wasatch pressure projection method with wall boundary conditions. Although experimental data is scarcely available for this test case, there are several numerical studies on the lid-driven-cavity that one may employ for comparison purposes. Our choice falls on the study by Ghia et al. (1982) which presents a comprehensive review of data in the literature as well as a reliable benchmark for comparison purposes. Here, we reproduce all the cases discussed in their study.

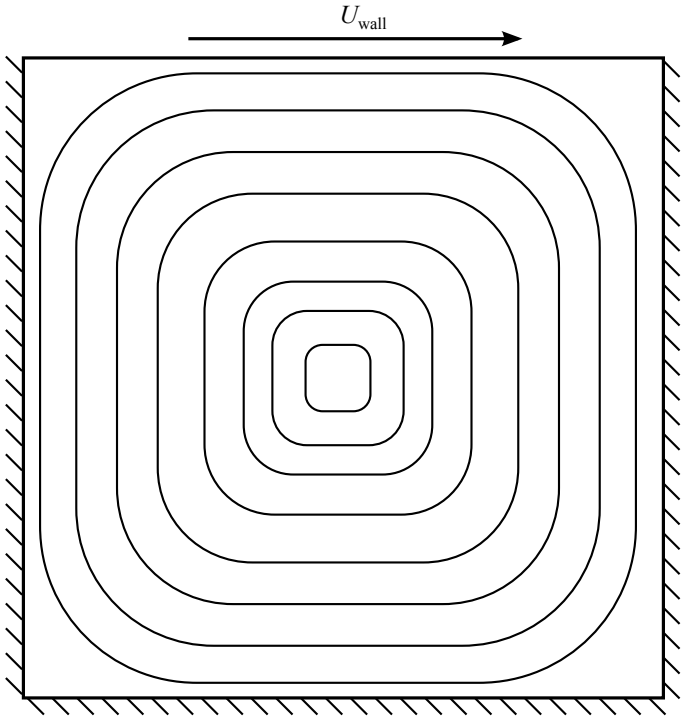


Figure 10.1: Lid-Driven-Cavity geometry.

Table 10.1: Summary of parameters used for the various cases.

Re	ρU_{wall}	Δt	CFL
100	0.1002	0.75	0.0096
400	0.4008	0.75	0.0385
1000	1.002	0.75	0.0963
3200	3.2064	0.75	0.3083
5000	5.01	0.5	0.3212
7500	7.515	0.25	0.2409
10000	10.02	0.125	0.1606

10.1.1 Problem Description

Consider a 1mx1m square cavity that is filled with a fluid with the top lid moving at a constant speed. We choose water at 293 K as our working fluid. The domain is discretized into $N_x = N_y = 128$ equally spaced control volumes (Note that in Uintah, we must specify a nonzero dimension in the orthogonal direction to the cavity's plane. here, we set $L_z = 1$m and $N_z = 2$ with periodic conditions in the z-direction. We also only solve the momentum equations in the plane of the driven cavity).

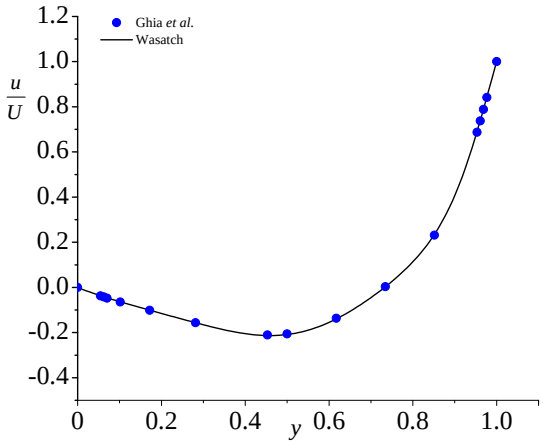
We test seven cases with different Reynolds number. The Reynolds number for the cavity is defined as $\text{Re} = \frac{\rho U_{\text{wall}} L}{\mu}$ where ρ is the density of the fluid (in this case, water @ 293 K), U_{wall} is the velocity of the lid, L is the width of the lid, and μ is the dynamic viscosity. The Reynolds numbers range is given by $\text{Re} = \{100, 400, 1000, 3200, 5000, 7500, 10000\}$ as dictated by the reference data. The boundary conditions on all walls are set to no-slip and no-permeability conditions ($\rho \mathbf{u} = \mathbf{0}$) except on the moving lid where the tangential momentum is set to reproduce U_{wall} that corresponds to the desired Reynolds number while the normal velocity is set to zero. A summary of these conditions is given in Table

10.2 Taylor Flow in a Channel

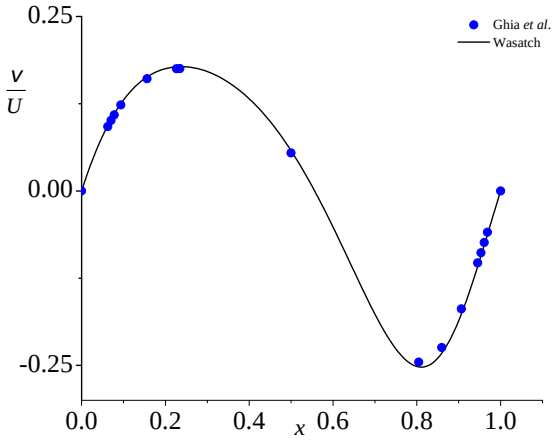
The Taylor flow in a channel describes the steady inviscid state of a fluid in a channel with permeable sidewalls. This is a great case for testing inlet and outflow boundary conditions.

10.3 Decay of Isotropic Turbulence

This classic verification case provides a somewhat reasonable benchmark to verify the sanity of our turbulence model implementations. It consists of an initial isotropic velocity field in a periodic box that is allowed to decay over time. The initial condition is made to fit an experimentally-measured energy spectrum. The code is then allowed to calculate the decay of this initial field, whose energy spectrum is measured at subsequent times and compared with experimental measurements.

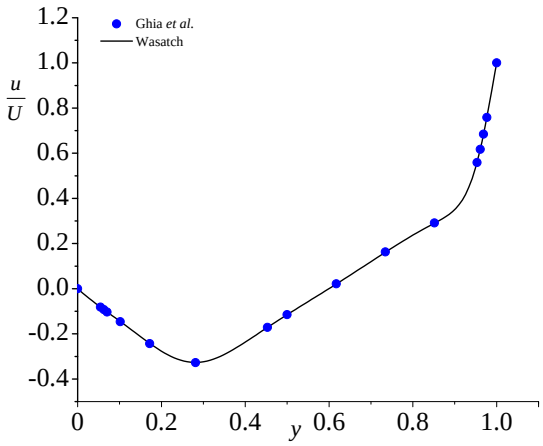


(a) Normalized u-velocity component.

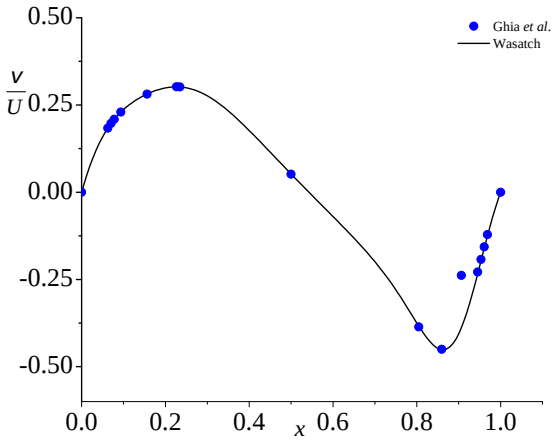


(b) Normalized y-velocity component.

Figure 10.2: Comparison of Wasatch results with those presented in Ghia et al. (1982) for Re = 100.

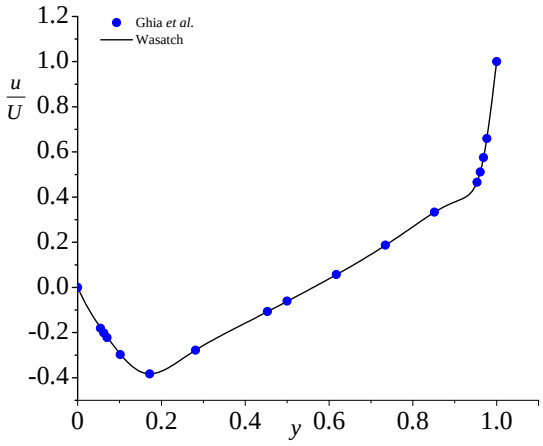


(a) Normalized u-velocity component.

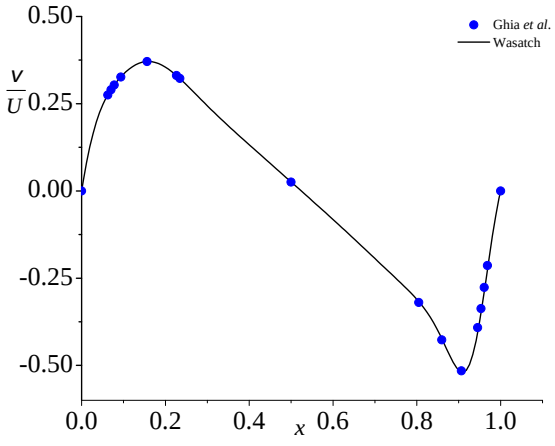


(b) Normalized y-velocity component.

Figure 10.3: Comparison of Wasatch results with those presented in Ghia et al. (1982) for Re = 400.

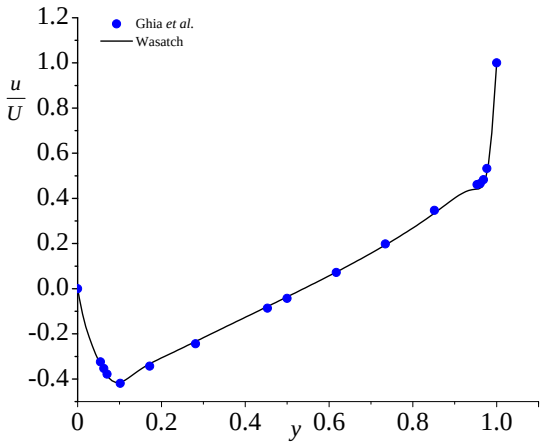


(a) Normalized u-velocity component.

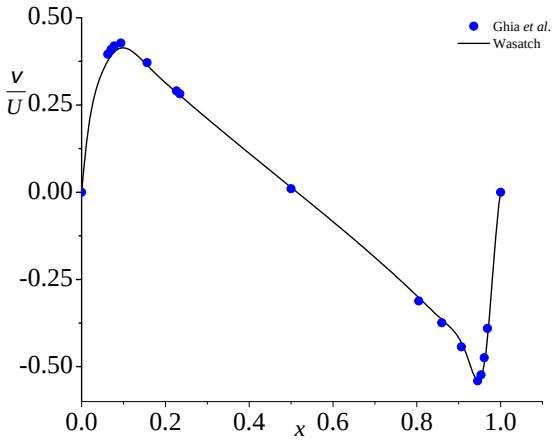


(b) Normalized y-velocity component.

Figure 10.4: Comparison of Wasatch results with those presented in Ghia et al. (1982) for Re = 1000.

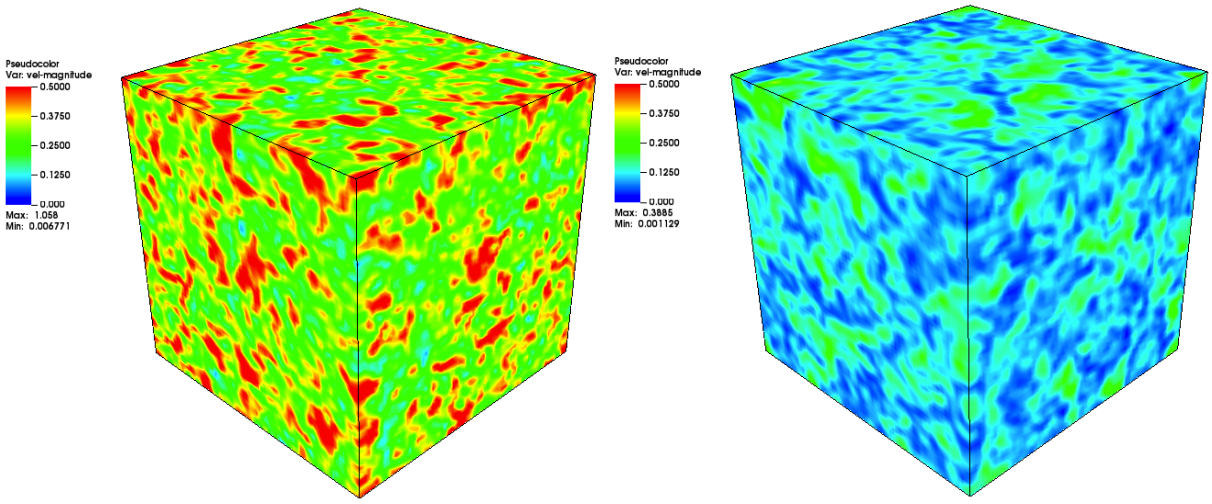


(a) Normalized u-velocity component.



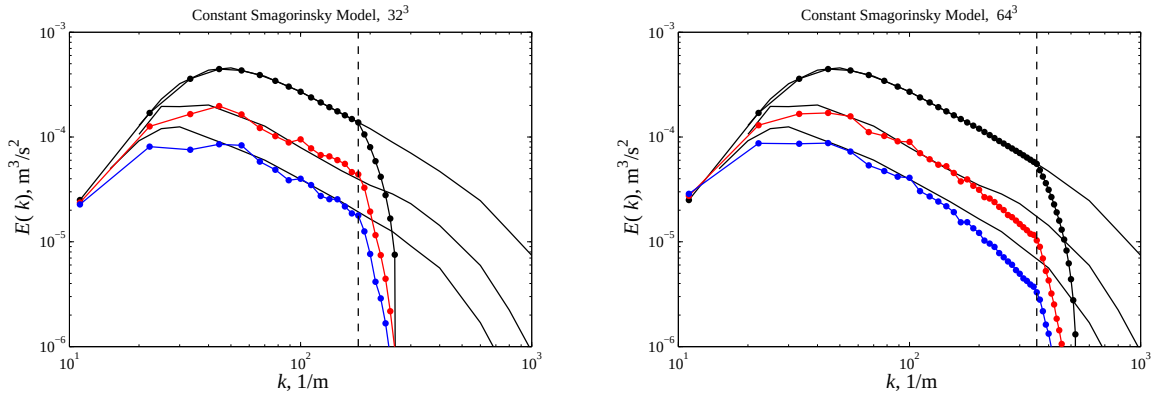
(b) Normalized y-velocity component.

Figure 10.5: Comparison of Wasatch results with those presented in Ghia et al. (1982) for Re = 3200.



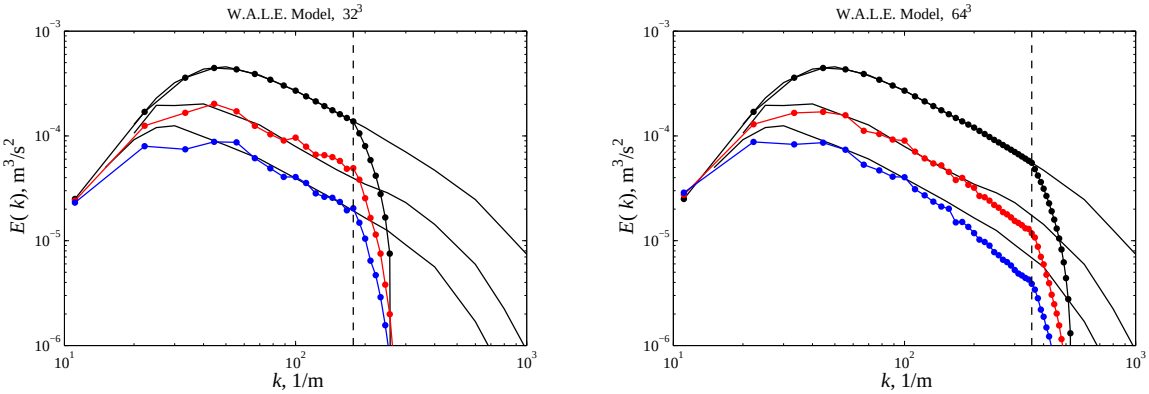
(a) $t = 0$ s **(b)** $t = 0.67$ s

Figure 10.6: Contours of velocity magnitude for the decay of isotropic turbulence in a periodic box. Results are shown using the Smagorinsky-Lilly model at (a) $t = 0$ s and (b) $t = 0.67$ s.



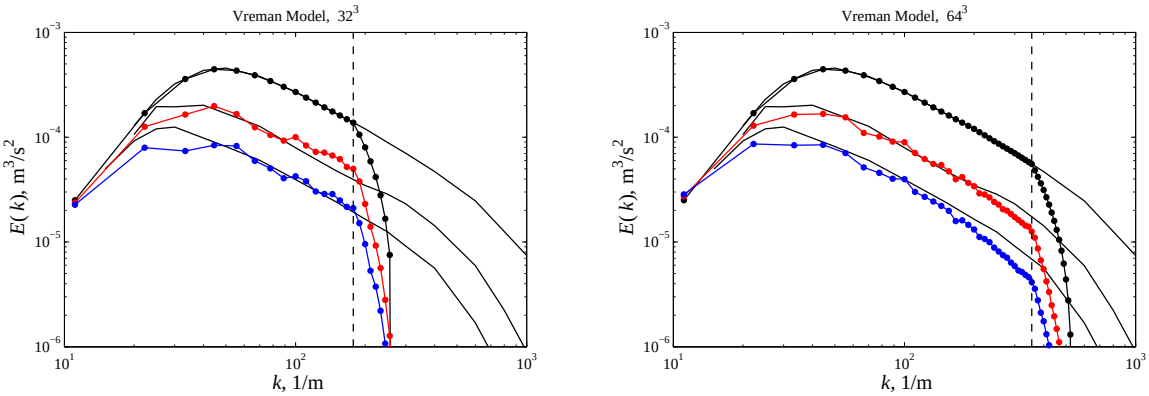
(a) **(b)**

Figure 10.7: Turbulent kinetic energy spectrum for the constant Smagorinsky model with $C_s = 0.2$ and for grid sizes (a) 32x32x32, and (b) 64x64x64.



(a) **(b)**

Figure 10.8: Turbulent kinetic energy spectrum for the WALE model with $C_{\mathrm{w}} = 0.7$ and for grid sizes (a) 32x32x32, and (b) 64x64x64.



(a) **(b)**

Figure 10.9: Turbulent kinetic energy spectrum for the Vreman model with $C_{\mathrm{s}} = 0.2$ (or $C_{\mathrm{v}} = 2.5C_{\mathrm{s}}^2 = 0.1$) and for grid sizes (a) 32x32x32, and (b) 64x64x64.

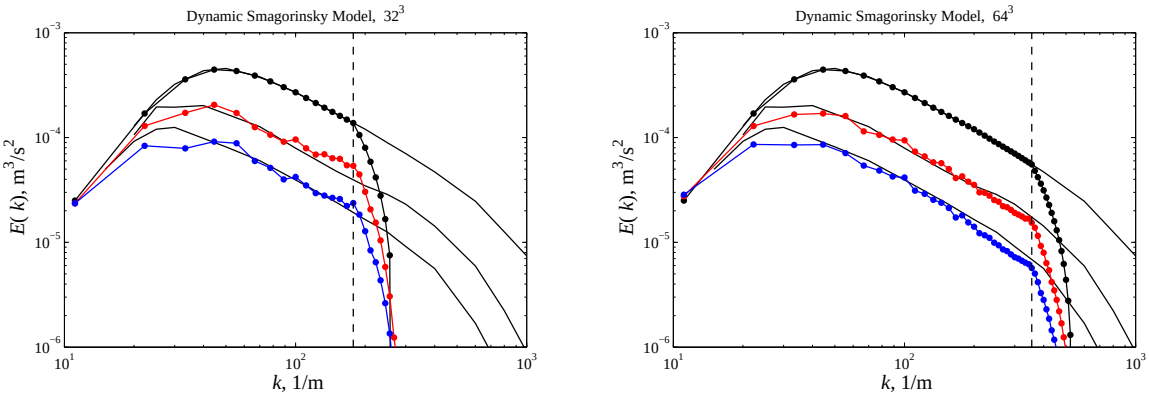


Figure 10.10: Turbulent kinetic energy spectrum for the dynamic Smagorinsky model for grid sizes (a) 32x32x32, and (b) 64x64x64.

Chapter 11

Summary of Supported Expressions

11.1 Field Expressions

Constant

The `Constant` expression is the most elemental type of expression used in Wasatch. As the name designates, this expression creates a constant field for any quantity. For example, Listing (11.1) shows the input specification for a constant expression that initializes a field ϕ to 1.0.

```
<BasicExpression type="SVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="phi" state="STATE_NONE" />
  <Constant>1.0</Constant>
</BasicExpression>
```

Listing 11.1: Specification of a Constant expression.

ExprAlgebra

The `ExprAlgebra` expression provides a simple and intuitive mechanism to add, subtract, or multiply an arbitrary number of source expressions. As an example, consider initializing a scalar variable ϕ as

$$\phi = x + y + z \tag{11.1}$$

This may be easily accomplished by The `ExprAlgebra` expression. The input specification is shown in Listing (11.2).

```
<BasicExpression type="SVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="phi" state="STATE_N" />
  <ExprAlgebra algebraicOperation="SUM">
      <!-- specify the list of tags of the source expressions -->
      <NameTag name="XSVOL" state="STATE_NONE"/>
```

```
      <NameTag name="YSVOL" state="STATE_NONE"/>
      <NameTag name="ZSVOL" state="STATE_NONE"/>
  </ExprAlgebra>
</BasicExpression>
```

Listing 11.2: Specification of an ExprAlgebra expression as the sum of three other expressions.

It is possible to construct highly complicated expressions with `ExprAlgebra` by combining different `ExprAlgebra` expressions. This, however, is highly costly from the point of view of graph granularity and scheduling. It is highly desirable that dedicated expressions be written that accomplish specific purposes. However, `ExprAlgebra` remains a very useful expression for constructing complex initial conditions where the cost invested at initialization is shadowed by the time advance solution.

LinearFunction

The `LinearFunction` expression applies a linear transformation to any field of your choice according to

$$\psi = a\phi + b, \tag{11.2}$$

where ψ is the destination field, ϕ is the source field, a is the slope, and b is the intercept. Below is an example of using the LinearFunction expression to initialize a scalar field ϕ as a linear function of the the coordinate x as $\phi = 0.1x + 1.25$.

```
<BasicExpression type="SVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="phi" state="STATE_N" />
  <LinearFunction slope="0.1" intercept="1.25">
      <!-- specify the tag of the independent variable -->
      <NameTag name="XSVOL" state="STATE_NONE"/>
  </LinearFunction>
</BasicExpression>
```

Listing 11.3: Specification of a LinearFunction as an initial condition for a scalar field ϕ.

ParabolicFunction

As the name implies, the `ParabolicFunction` expression generates a field of the form

$$\phi = ax^2 + bx + c. \tag{11.3}$$

The following listing shows how to initialize a field with a `ParabolicFunction` as $\phi = 0.1x^2 + 1.25x + 0.3$.

```
<BasicExpression type="SVOL">
  <TaskList>initialization</TaskList>
```

```
  <NameTag name="phi" state="STATE_N" />
  <LinearFunction slope="0.1" intercept="1.25">
      <!-- specify the tag of the independent variable -->
      <NameTag name="XSVOL" state="STATE_NONE"/>
  </LinearFunction>
</BasicExpression>
```

Listing 11.4: Specification of a ParabolicFunction as an initial condition for a scalar field ϕ.

SineFunction

As the name implies, the `SineFunction` expression generates a field of the form

$$\phi = A\sin(\omega x) + \psi. \tag{11.4}$$

where A is the amplitude, ω is the frequency, and ψ is the offset. The specification of a `SineFunction` takes three attributes and the nametag of an independent variable (e.g. x). The following listing shows how to initialize a field with a `SineFunction` as $f = 2\sin(10x) + 3$.

```
<BasicExpression type="SVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="f" state="STATE_NONE" />
  <SineFunction amplitude="2.0" frequency="10.0" offset="3.0">
     <NameTag name="XSVOL" state="STATE_NONE" />
   </SineFunction>
</BasicExpression>
```

Listing 11.5: Specification of a SineFunction as an initial condition for a scalar field f.

GaussianFunction

Generates a field of the form

$$f(x) = y_0 + a\exp\left[\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{11.5}$$

where a is the amplitude, μ is the mean, and σ is the deviation, and y_0 is the baseline. The specification of a `GaussianFunction` takes four attributes and the nametag of an independent variable (e.g. x). The following listing shows how to initialize a field with a `GaussianFunction` as $f = 1 + 20\exp\left[\frac{(x-5)^2}{8}\right]$.

```
<BasicExpression type="SVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="f" state="STATE_NONE" />
  <GaussianFunction amplitude="20.0" deviation="2.0" mean="5.0"
     baseline="1.0">
    <NameTag name="XSVOL" state="STATE_NONE" />
```

```
  </GaussianFunction>
</BasicExpression>
```

Listing 11.6: Specification of a GaussianFunction as an initial condition for a scalar field f.

DoubleTanhFunction

Generates a field of the form

$$f(x) = \frac{A}{2}\left(1+\tanh\left(\frac{x-L_1}{w}\right)\right)\left(1-\frac{1}{2}\tanh\left(\frac{x-L_2}{w}\right)\right) \tag{11.6}$$

where w is the width of the transition, A is the amplitude of the transition, L_1 is the midpoint for the upward transition, and L_2 is the midpoint for the downward transition. The following listing shows how to initialize a field with a `DoubleTanhFunction` as $f(x) = 5\left(1+\tanh\left(\frac{x-1}{4}\right)\right)\left(1-\frac{1}{2}\tanh\left(\frac{x-2}{4}\right)\right)$.

```
<BasicExpression type="SVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="phi3-scalar" state="STATE_N" />
  <DoubleTanhFunction amplitude="10.0" width="4.0" midpointUp="1.0"
     midpointDown="2.0">
      <NameTag name="XSVOL" state="STATE_NONE"/>
  </DoubleTanhFunction>
</BasicExpression>
```

Listing 11.7: Specification of a DoubleTanhFunction as an initial condition for a scalar field f.

StepFunction

Generates a Heaviside step function of the form

$$f(x) = \begin{cases} l & x < x_0 \\ g & x \geq x_0 \end{cases} \tag{11.7}$$

where x_0 is the transition point, l is referred to as the low value and denotes the value of f below the transition point, and g is referred to as the high value and denotes the value of f above the transition point. The following listing shows how to initialize a field with a `StepFunction` as $f(x) = \begin{cases} 2 & x < 0.1 \\ 1 & x \geq 0.1 \end{cases}$.

```
<BasicExpression type="SVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="phi3-scalar" state="STATE_N" />
  <StepFunction transitionPoint="0.5" lowValue="2.0" highValue="1.0">
```

```
      <NameTag name="XSVOL" state="STATE_NONE"/>
  </DoubleTanhFunction>
</BasicExpression>
```

Listing 11.8: Specification of a StepFunction as an initial condition for a scalar field f.

ReadFromFile

Creates an expression that reads data from a file. The file is formatted as "x y z value" ASCII and then zipped. A sample file is found in the Wasatch inputs directory. The specification is shown below.

```
<BasicExpression type="SVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="f" state="STATE_NONE"/>
  <ReadFromFile>
    <FileName>inputs/Wasatch/read-from-file-test-init.dat.gz</FileName>
  </ReadFromFile>
</BasicExpression>
```

Listing 11.9: ReadFromFile specification as an initial condition for a scalar field f.

ExponentialVortex

This expression implements a rotating vortex in a uniform stream. The corresponding streamfunction is given by the superposition of an exponential vortex and a uniform flow and is given by

$$\psi = \Gamma \exp\left(-\frac{r^2}{2R^2}\right) - Vx + Uy, \tag{11.8}$$

where Γ is the strength of the vortex, $r = \sqrt{(x-x_0)^2 + (y-y_0)^2}$ is the radial distance from the center of the vortex, R is the radius of the vortex, and V and U are the the transverse and axial free stream velocities, respectively. The resulting velocity field is then calculated using the Stokes' streamfunction

$$u = \frac{\partial \psi}{\partial y} = U - \frac{\Gamma}{R^2}(y-y_0)\exp\left(-\frac{r^2}{2R^2}\right) \tag{11.9}$$

and

$$v = -\frac{\partial \psi}{\partial x} = V + \frac{\Gamma}{R^2}(x-x_0)\exp\left(-\frac{r^2}{2R^2}\right). \tag{11.10}$$

This expression is best used as an initialization for the momentum or velocity fields.

What follows is an example for initializating the axial and transverse momentum components with an exponential vortex, centered at $[0.5, 0.5]$ with a strength of 10^{-3}, a radius of $R_0 = 0.1$, and a free stream velocity $U_0 = 1$. Note that you must specify both velocity or momentum components to recover physically relevant initial conditions. You can find the "ups" file in the Wasatch inputs directory under the name "exponential-vortex-example.ups".

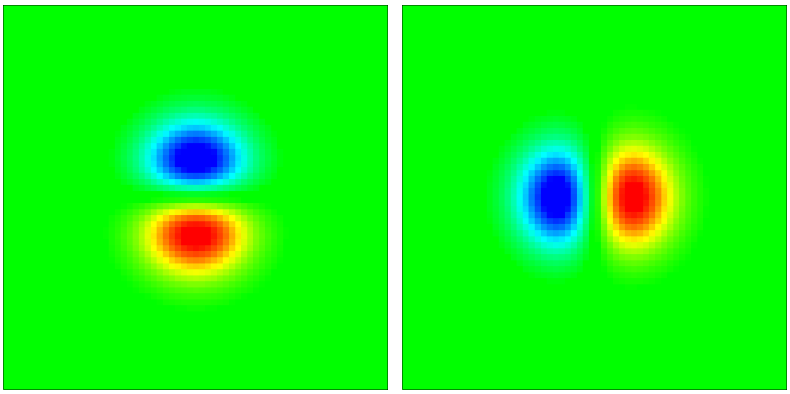
```
<BasicExpression type="XVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="x-mom" state="STATE_N" />
  <ExponentialVortex x0="0.5" y0="0.5" G="1e-3" R="0.1" U="1" V="0"
     velocityComponent="X1">
    <Coordinate1>
      <NameTag name="XXVOL" state="STATE_NONE"/>
    </Coordinate1>
    <Coordinate2>
      <NameTag name="YXVOL" state="STATE_NONE"/>
    </Coordinate2>
  </ExponentialVortex>
</BasicExpression>
```

Listing 11.10: Specification of an ExponentialVortex as an initial condition for axial momentum.

```
<BasicExpression type="YVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="y-mom" state="STATE_N" />
  <ExponentialVortex x0="0.5" y0="0.5" G="1e-3" R="0.1" U="1" V="0"
     velocityComponent="X2">
    <Coordinate1>
      <NameTag name="XYVOL" state="STATE_NONE"/>
    </Coordinate1>
    <Coordinate2>
      <NameTag name="YYVOL" state="STATE_NONE"/>
    </Coordinate2>
  </ExponentialVortex>
</BasicExpression>
```

Listing 11.11: Specification of an ExponentialVortex as an initial condition for transverse momentum.

The resulting flow field generated by this initialization is shown below.



(a) x-momentum **(b)** y-momentum

Figure 11.1: Momentum field resulting from ExponentialVortex initialization.

LambsDipole

Lamb's dipole is a steady-state solution to the Euler equations using a vorticity-streamfunction approach. In this approach, one specifies an explicit relation between the vorticity and the streamfunction. Lamb's dipole solution assumes that the azimuthal vorticity has a linear dependence on the streamfunction such that

$$\omega_z = k^2 \psi \tag{11.11}$$

The solution to the Euler equations in this case, in an infinite mass of fluid, results in the following

$$\psi_{\text{dipole}}(r,\theta) = \begin{cases} \frac{2U_0}{kJ_0(kR)} J_1(kr)\sin\theta & r \le R \\ U_0(r - \frac{R^2}{r})\sin\theta & r > R \end{cases} \tag{11.12}$$

where U_0 is the propagation velocity of the dipole, k is the proportionality constant relating the vorticity to the streamfunction and may be determined from $J_1(kR) = 0$ or $kR \approx 3.8317$, R is the radius of the dipole, and J_0 and J_1 are the zeroth- and first-order Bessel functions of the first kind, respectively. The dipole is finall be superimposed on a uniform stream.

To make this solution accessible to code implementation, it must be transformed to cartesian coordinates. We also wish to place the dipole at any location in the flow field. We first calculate the radial and tangenial velocities of ψ_{dipole}. These are given by the polar streamfunction relations

$$u_r = \frac{1}{r}\frac{\partial \psi}{\partial \theta}; \quad u_\theta = -\frac{\partial \psi}{\partial r} \tag{11.13}$$

These calculations return

$$u_{r,\text{dipole}} = \begin{cases} \frac{2U_0}{kJ_0(kR)} J_1(kr)\cos\theta & r \le R \\ U_0(1 - \frac{R^2}{r^2})\cos\theta & r > R \end{cases} \tag{11.14}$$

$$u_{\theta,\text{dipole}} = \begin{cases} -U_0 \sin\theta + \frac{U_0}{J_0(kR)} J_2(kr)\sin\theta & r \le R \\ -U_0(1+\frac{R^2}{r^2})\cos\theta & r > R \end{cases} \tag{11.15}$$

Finally, the axial and transverse velocities are constructed by setting

$$u_{x,\text{dipole}} = u_{r,\text{dipole}}\cos\theta - u_{\theta,\text{dipole}}\sin\theta \tag{11.16}$$

and

$$u_{y,\text{dipole}} = u_{r,\text{dipole}}\sin\theta + u_{\theta,\text{dipole}}\cos\theta \tag{11.17}$$

By subsituting $\theta = \tan^{-1}(\frac{y}{x})$ or using $\cos\theta = \frac{x}{r}$ and $\sin\theta = \frac{y}{r}$, we recover

$$u_{x,\text{dipole}} = \begin{cases} \frac{2U_0}{kJ_0(kR)r^3}[kry^2 J_0(kr) + (x^2-y^2)J_1(kr)] & r \le R \\ U_0 + \frac{U_0 R^2}{r^4}(r^2 - 2x^2) & r > R \end{cases}; \quad r = \sqrt{x^2+y^2} \tag{11.18}$$

and

$$u_{y,\text{dipole}} = \begin{cases} \frac{2U_0}{J_0(kR)r^2} J_2(kr) & r \le R \\ -\frac{2U_0 R^2}{r^4} xy & r > R \end{cases}; \quad r = \sqrt{x^2+y^2} \tag{11.19}$$

Finally, the total flowfield with a uniform flow given by U in the x-direction is computed as

$$u_x = U + u_{x,\text{dipole}} \tag{11.20}$$

and

$$u_y = u_{y,\text{dipole}} \tag{11.21}$$

To place the dipole at a random point (x_0, y_0), we simply perform the following substitutions in the above equations

$$x \to x - x_0; \quad y \to y - y_0; \quad r \to \sqrt{(x-x_0)^2 + (y-y_0)^2} \tag{11.22}$$

What follows is an example for initializating the axial and transverse momentum components with a Lamb-dipole vortex, centered at $[0.0, 0.0]$ with a propagation velocity of $U_0 = 0.9$, a radius of $R_0 = 0.1$, and a free stream velocity $U = 0.1$. Note that you must specify both velocity or momentum components to recover physically relevant initial conditions. You can find the "ups" file in the Wasatch inputs directory under the name "dipole-vortex-example.ups".

```
<BasicExpression type="XVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="x-mom" state="STATE_N" />
  <LambsDipole x0="0.0" y0="0.0" G="0.9" R="0.25" U="0.1"
     velocityComponent="X1">
    <Coordinate1>
      <NameTag name="XXVOL" state="STATE_NONE"/>
```

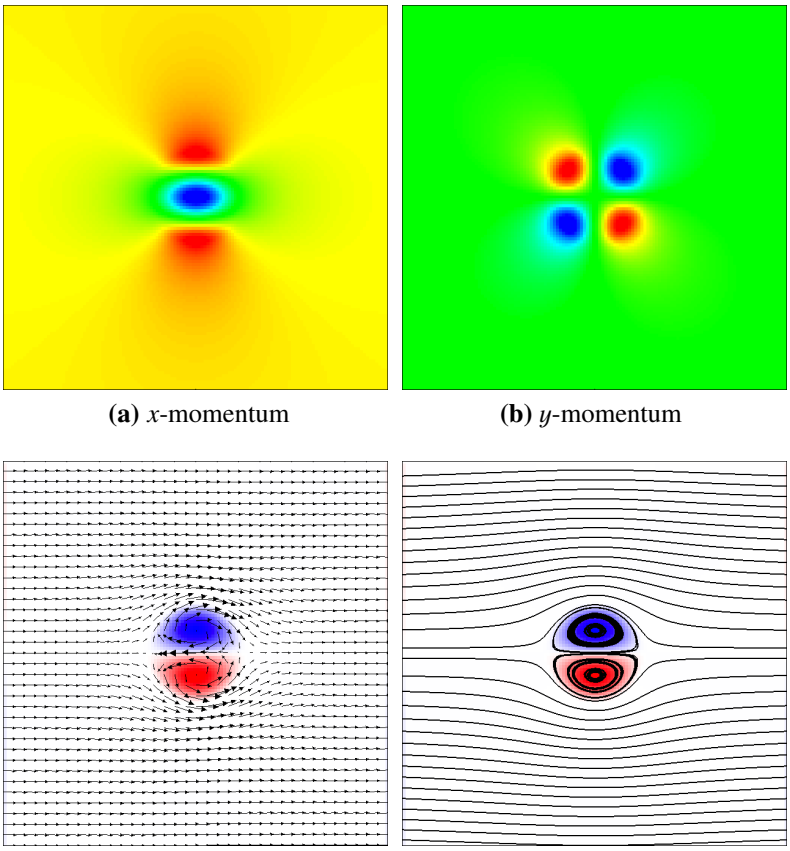
```
    </Coordinate1>
    <Coordinate2>
      <NameTag name="YXVOL" state="STATE_NONE"/>
    </Coordinate2>
  </LambsDipole>
</BasicExpression>
```

Listing 11.12: Specification of a LambsDipole expression as an initial condition for axial momentum.

```
<BasicExpression type="YVOL">
  <TaskList>initialization</TaskList>
  <NameTag name="y-mom" state="STATE_N" />
  <LambsDipole x0="0.0" y0="0.0" G="0.9" R="0.25" U="0.1"
     velocityComponent="X2">
    <Coordinate1>
      <NameTag name="XYVOL" state="STATE_NONE"/>
    </Coordinate1>
    <Coordinate2>
      <NameTag name="YYVOL" state="STATE_NONE"/>
    </Coordinate2>
  </ExponentialVortex>
</BasicExpression>
```

Listing 11.13: Specification of a LambsDipole expression as an initial condition for transverse momentum.

The resulting flow field generated by this initialization is shown below.



(a) x-momentum **(b)** y-momentum

(c) vorticity field wih vectors **(d)** vorticity field with streamlines

Figure 11.2: Momentum and vorticity fields resulting from LambsDipole initialization.

Reduction

The Reduction expression provides a means for specifying MPI-Reduction operations on Wasatch expressions. The Reduction expression consumes a field by applying a certain reduction operation (i.e. min) and returns a single value that is the same across all patches. Reduction variables can be driven through the input file with the following spec

```
<Reduction spec="MULTIPLE NO_DATA"
  <!-- tasklist: task that computes this expression. currently support
     advance_solution only -->
  attribute1="tasklist REQUIRED STRING 'advance_solution'"
  <!-- op: reduction operation: min, max, or sum -->
  attribute2="op REQUIRED STRING 'min,max,sum'"
  <!-- output: specify if you wish to output the reduced value to the
     terminal -->
  attribute3="output OPTIONAL BOOLEAN">
  <!-- NameTag: nametag of the reduced variable -->
  <NameTag          spec="REQUIRED" />
```

```
  <!-- Source: information about the sourcefield -->
  <Source spec="REQUIRED NO_DATA"
    <!-- type: fieldtype of the source field -->
    attribute1="type REQUIRED STRING 'PERPATCH,SVOL,XVOL,YVOL,ZVOL'" >\
    <!-- NameTag: nametag of the source field -->
    <NameTag spec="REQUIRED"/>
  </Source>
</Reduction>
```

Listing 11.14: XML specification for a Reduction expression

The comments in Listing (11.14) describe the available options that can be used when specifying reduction variables via input. As an example of defining a reduction variable through the input file, consider calculating the maximum velocity (i.e. u) and outputing it to the terminal. This may achived via the following specification

```
<Reduction tasklist="advance_solution" op="max" output="true">
  <NameTag name="u_max" state="STATE_NONE" />
  <Source type="XVOL">
    <NameTag name="u" state="STATE_NONE" />
  </Source>
</Reduction>
```

Listing 11.15: Specification of a reduction variable that calculates the maximum velocity and outputs its value to the terminal.

Listing (11.15) creates an expression called u_max that stores the maximum value of the velocity field.

Warning: Reduction expressions cannot be saved by design (e.g. PerPatch variable). However, the reduction counterpart can be saved. Therefore, to save reduction variables, simply append their name with "_uintah" and use that in the DataArchiver section of the ups input file. For example, to the save "u_max" shown in Listing (11.15), use the following in the DataArchiver <save label="u_max_uintah"/>.

RayleighTaylor

Generates an initial condition for a Rayleigh-Taylor instability simulation with a perturbed interface. The formula is

$$f(y) = \begin{cases} l & y < y_0 + A\sin(2\pi x_1)\sin(2\pi x_2) \\ g & \text{otherwise} \end{cases} \tag{11.23}$$

where l and g are constants, y, x_1, and x_2 are coordinates. An example of the RayleighTaylor expression is shown below

```
<BasicExpression type="SVOL">
   <TaskList>initialization</TaskList>
   <NameTag name="f" state="STATE_NONE" />
```

```
    <RayleighTaylor transitionPoint="0.5" lowValue="0.0" highValue="1.0
       " frequency="10" amplitude="0.008" x1="XSVOL" x2="XSVOL">
      <NameTag name="YSVOL" state="STATE_NONE"/>
    </RayleighTaylor>
</BasicExpression>
```

Listing 11.16: Specification of a RayleighTaylor expression.

GeometryBased

The GeometryBased expression is a very powerful tool for setting values on fields using complex gometrical shapes. This expression uses geometry primitives to build complicated shapes with different values inside those primitives. The specification is shown in Listing (11.17). An example is shown in Listing (11.18).

```
<!-- provides a way to initialize a field using geometry primitives -->
<GeometryBased      spec="OPTIONAL NO_DATA"
                    attribute1="value REQUIRED DOUBLE"> <!-- this is
                       the outside value -->
  <Intrusion      spec       = "MULTIPLE NO_DATA"
                  attribute1 = "name OPTIONAL STRING"
                  attribute2 = "value REQUIRED DOUBLE"> <!-- value of
                     field inside this intrusion -->
    <geom_object  spec="REQUIRED"/>
  </Intrusion>
</GeometryBased>
```

Listing 11.17: Specification of a GeometryBased expression.

```
<BasicExpression type="SVOL" >
  <TaskList>initialization</TaskList>
  <NameTag name="f" state="STATE_NONE" />
  <GeometryBased value="0.0">
    <Intrusion value="10" name="mickey mouse">
      <geom_object>
        <union>
         <cylinder>
          <bottom>[0.5,0.5,-1]</bottom>
          <top>[0.5,0.5,1]</top>
          <radius>0.2</radius>
         </cylinder>

         <cylinder>
          <bottom>[0.25,0.65,-1]</bottom>
          <top>[0.25,0.65,1]</top>
          <radius>0.1</radius>
         </cylinder>
```

```
        <cylinder>
         <bottom>[0.75,0.65,-1]</bottom>
         <top>[0.75,0.65,1]</top>
         <radius>0.1</radius>
        </cylinder>

       </union>
     </geom_object>
   </Intrusion>

 </GeometryBased>
</BasicExpression>
```

Listing 11.18: Example of a GometryBased expression.

11.2 Boundary-Condition Expressions

TurbulentInlet

The `TurbulentInlet` expression is a boundary expression that provides a means for applying time-dependent data that is correlated based on turbulence arguments. This expression requires that the random data be first generated by the DigitalFilterGenerator (DFG) tool. The DFG tool can be found in the StandAlone directory (e.g. build/StandAlone). Details about using the DFG can be found in the Uintah user guide, in the Arches chapter.

```
<TurbulentInlet  spec="OPTIONAL NO_DATA"
                 attribute1="period OPTIONAL INTEGER"
                 attribute2="timeperiod OPTIONAL DOUBLE">
  <BaseName      spec="REQUIRED STRING" />  <!-- this will create 3
     expressions: x-basename, y-basename, z-basename !-->
  <InputFile     spec="REQUIRED STRING"/>
</TurbulentInlet>
```

Listing 11.19: TurbulentInlet UPS specification

An example of using `TurbulentInlet` in Wasatch is shown here. In this example, the `TurbulentInlet` expression will parse the data provided through the specified inputfile. The Wasatch parser will then create three expressions corresponding to the u, v, and w velocities, respectively. The names of these expressions are `x-basename`, `y-basename`, and `z-basename`. These can be set into the `<functor_name>` specification for the boundary conditions. The `period` specifies the number of timesteps at which the next turbulent data is loaded into the turbulent inlet while the `timeperiod` specifies after how many seconds the next turbulent data is loaded. Both of these are optional and if omitted, the data is loaded at every timestep. When the turbulent data file is parsed entirely, the data is repeated from the begining.

Chapter 12

Summary of Current Capabilities and Work in Progress

Table 12.1: Synopsis of Wasatch development.

Task	Status
Advective terms including flux limiters	completed
Diffusive terms for basic diffusive flux expressions	completed
Tabular property evaluation for implementation of common combustion models	completed
Pressure projection	completed
Basic CFD algorithm using expressions graph	completed
Boundary conditions for scalar transport	completed
Quadrature method of moments	completed
Basic population balance equation	completed
Filter operator for dynamic turbulent models	completed
Boundary conditions for momentum equations	in progress
Weak form of transport equations	in progress
Variable density algorithm (including appropriate terms in the pressure projection)	in progress
Higher-order time integrator (Runge-Kutta based)	in progress
LES models for turbulent diffusive fluxes (Smagorinsky and dynamic models)	in progress

Bibliography

Erlebacher, G., Hussaini, M. Y., Speziale, C. G., and Zang, T. A. (1992). Toward the Large-Eddy simulation of compressible turbulent flows. *Journal of Fluid Mechanics*, 238:155–185. 1.1.1.1

Garnier, E., Sagaut, P., and Adams, N. (2009). *Large eddy simulation for compressible flows*. Springer Verlag. 1.1.1.1

Geurts, B., Vreman, B., Kuerten, H., and Theofilis, V. (1993). LES modelling errors in free and wall-bounded compressible shear layers. *Engineering Turbulence Modelling and Experiments*, 2:325–334. 1.1.1, 1.1.1.1

Ghia, U., Ghia, K., and Shin, C. (1982). High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of computational physics*, 48(3):387–411. 10.1, 10.2, 10.3, 10.4, 10.5

Nicoud, F. and Ducros, F. (1999). Subgrid-Scale stress modelling based on the square of the velocity gradient tensor. *Flow, Turbulence and Combustion*, 62(3):183–200. 1.1.1.1

Oberkampf, W. L. and Roy, C. J. (2010). *Verification and validation in scientific computing*. Cambridge University Press. 3.1, 3.2.1

Smagorinsky, J. (1963). General circulation experiments with the primitive equations. *Monthly Weather Review*, 91:99–164. 1.1.1.1, 1.1.1.1

Vreman, A. (2004). An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications. *Physics of fluids*, 16:3670. 1.1.1.1

Vreman, B., Geurts, B., and Kuerten, H. (1995). Subgrid-modelling in LES of compressible flow. *Applied Scientific Research*, 54(3):191–203. 1.1.1