Wasatch

Tony Saad, James C. Sutherland, Amir Biglari

May 23, 2012

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 supports the following transport equations:

- 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. In addition, a variety of source terms.

1.1 Momentum Transport

Wasatch solves the variable density momentum equations

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

in conjunction with the continuity equation

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

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. 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

$$\nabla^2 p = \frac{\partial^2 \rho}{\partial t^2} + \nabla \cdot \nabla \cdot \rho \mathbf{u}\mathbf{u} - \nabla \cdot \nabla \cdot \tau_{ij} - \nabla \cdot \rho \mathbf{g} \tag{1.3}$$

The solution of Eq. (1.3) 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

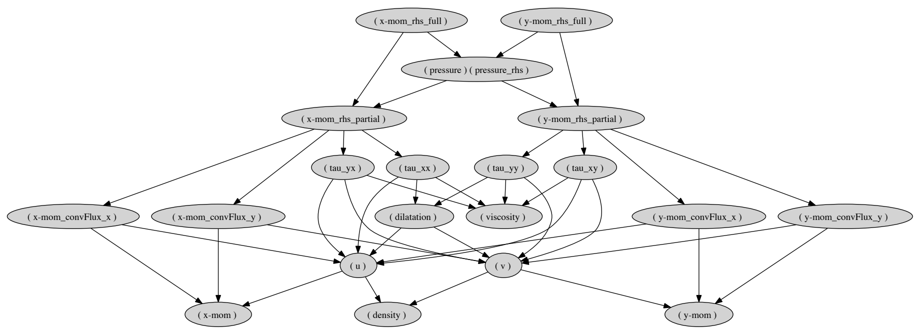

Figure 1.1: Expression graph for a two dimensional fluid flow problem using the Poisson equation for the pressure.

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 reduces to

$$\nabla^2 p = \nabla \cdot \nabla \cdot \rho \mathbf{uu} - \nabla \cdot \nabla \cdot \tau_{ij} - \nabla \cdot \rho \mathbf{g} \tag{1.4}$$

1.2 LES modeling

This part is currently in progress in Wasatch.

Momentum equation The filtered momentum equation is

$$\frac{\partial \bar{\rho} \tilde{v}_k}{\partial t} + \frac{\partial}{\partial x_l}(\bar{\rho} \tilde{v}_k \tilde{v}_l) = -\frac{\partial \bar{p}}{\partial x_k} + \frac{\partial}{\partial x_l}\tilde{\sigma}_{kl} - \frac{\partial}{\partial x_l}\tau_{kl} \tag{1.5}$$

where

$$\sigma_{kl} = -\frac{2}{3}\mu \frac{\partial v_j}{\partial x_j}\delta_{kl} + \mu \left(\frac{\partial v_k}{\partial x_l} + \frac{\partial v_l}{\partial x_k} \right) \tag{1.6}$$

And ~ shows the Favre averaged variables.

The most important difference between filtered and non-filtered momentum equation is in the diffusive term, τ_{kl}. According to the Smagorinsky eddy viscosity model it can be modeled as

$$\tau_{kl} - \frac{1}{3}q^2\delta_{kl} = -2C\bar{\rho}\Delta^2|\tilde{S}|(\tilde{S}_{kl} - \frac{1}{3}\tilde{S}_{mm}\delta_{kl}) \tag{1.7}$$

where $\tilde{S}_{kl} = \frac{1}{2}(\partial \tilde{u}_k/\partial x_l + \partial \tilde{u}_l/\partial x_k)$, $|\tilde{S}| = (2\tilde{S}_{kl}\tilde{S}_{kl})^{1/2}$ and $q^2 = \tau_{ii}$. q^2 is also parameterized using Yoshizawa's model to obtain

$$q^2 = 2C_I \bar{\rho} \Delta^2 |\tilde{S}|^2 \tag{1.8}$$

Where Δ is the grid size.

C and C_I are Smagorinsky constants which can be assumed to be a fixed number and also can be calculated using dynamic models. By using dynamic models and applying test-filters these constants are parameterized as

$$C_I = \frac{< \widehat{\bar{\rho}\tilde{v}_k\tilde{v}_k} - (1/\hat{\bar{\rho}})(\widehat{\bar{\rho}\tilde{v}_k})(\widehat{\bar{\rho}\tilde{v}_k}) >}{< 2\hat{\bar{\rho}}\hat{\Delta}^2|\hat{\tilde{S}}|^2 - 2\Delta^2\widehat{\bar{\rho}|\tilde{S}|^2} >} \tag{1.9}$$

$$C = \frac{\left\langle [\widehat{\bar{\rho}\tilde{v}_k\tilde{v}_l} - (1/\hat{\bar{\rho}})(\widehat{\bar{\rho}\tilde{v}_k}\widehat{\bar{\rho}\tilde{v}_l})]\tilde{S}_{kl} - \frac{1}{3}\tilde{S}_{mm}(\mathscr{T}_{jj} - \hat{\tau}_{jj}) \right\rangle}{\left\langle -2\hat{\bar{\rho}}\hat{\Delta}^2|\hat{\tilde{S}}|(\hat{\tilde{S}}_{kl}\tilde{S}_{kl} - \frac{1}{3}\hat{\tilde{S}}_{mm}\tilde{S}_{jj}) + 2\Delta^2(\widehat{\bar{\rho}|\tilde{S}|\tilde{S}_{kl}}\tilde{S}_{kl} - \frac{1}{3}\widehat{\bar{\rho}|\tilde{S}|\tilde{S}_{mm}}\tilde{S}_{jj}) \right\rangle} \tag{1.10}$$

Where $<>$ indicates volume averaging and $\mathscr{T}_{jj} - \hat{\tau}_{jj} = \mathscr{L}_{jj} = \widehat{\bar{\rho}\tilde{v}_j\tilde{v}_j} - (1/\hat{\bar{\rho}})(\widehat{\bar{\rho}\tilde{v}_j})(\widehat{\bar{\rho}\tilde{v}_j})$ and over hat means the test filter is applied.

Energy equation The filtered energy equation is

$$C_v \frac{\partial \bar{\rho}\tilde{T}}{\partial t} + C_v \frac{\partial}{\partial x_k}(\bar{\rho}\tilde{v}_k\tilde{T}) = -\bar{p}\frac{\partial \tilde{v}_k}{\partial x_k} + \tilde{\sigma}_{ik}\frac{\partial \tilde{v}_k}{\partial x_l} + \frac{\partial}{\partial x_k}\left(\tilde{\kappa}\frac{\partial \tilde{T}}{\partial x_l}\right) - C_v\frac{\partial q_k}{\partial x_k} \tag{1.11}$$

Similarly for the filtered energy equation, the diffusive term needs to be modeled somehow. Again, by considering eddy viscosity SGS model we can obtain

$$q_k = -\frac{\bar{\rho}\nu_T}{Pr_t}\frac{\partial \tilde{T}}{\partial x_k} \tag{1.12}$$

Where $\nu_T = C\Delta^2|\tilde{S}|$ and Pr number is again parametrized using dynamic models to obtain

$$Pr_t = C\left\langle \hat{\bar{\rho}}\hat{\Delta}^2|\hat{\tilde{S}}|\frac{\partial \hat{\tilde{T}}}{\partial x_k}\frac{\partial \tilde{T}}{\partial x_k} - \Delta^2\widehat{\bar{\rho}|\tilde{S}|\frac{\partial \tilde{T}}{\partial x_k}}\frac{\partial \tilde{T}}{\partial x_k} \right\rangle \times \left\langle \left(\frac{1}{\hat{\bar{\rho}}}\widehat{\bar{\rho}\tilde{v}_k}\widehat{\bar{\rho}\tilde{T}} - \widehat{\bar{\rho}\tilde{v}_k\tilde{T}}\right)\frac{\partial \tilde{T}}{\partial x_k} \right\rangle^{-1} \tag{1.13}$$

1.3 Generic Scalar Transport

Wasatch can solve any scalar transport equation of the general form

$$\frac{\partial \rho\phi}{\partial t} = -\nabla\cdot\rho\phi\mathbf{u} + \nabla\cdot\rho\Gamma_\phi\nabla\phi + S_\phi \tag{1.14}$$

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 1.14. Those forms are:

Strong form with constant density

$$\frac{\partial \phi}{\partial t} = -\nabla\cdot\phi\mathbf{u} + \nabla\cdot\Gamma_\phi\nabla\phi + \frac{S_\phi}{\rho} \tag{1.15}$$

Weak form with variable density

$$\frac{\partial \phi}{\partial t} = -\mathbf{u}\cdot\nabla\phi + \frac{1}{\rho}\nabla\cdot\rho\Gamma_{\phi}\nabla\phi + \frac{S_{\phi}}{\rho} \tag{1.16}$$

Weak from with constant density

$$\frac{\partial \phi}{\partial t} = -\mathbf{u}\cdot\nabla\phi + \nabla\cdot\Gamma_{\phi}\nabla\phi + \frac{S_{\phi}}{\rho} \tag{1.17}$$

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

$$\mathbf{u}\cdot\nabla\phi = \nabla\cdot\phi\mathbf{u} - \phi\nabla\cdot\mathbf{u} \tag{1.18}$$

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.

1.4 Population Balances

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

$$\frac{\partial N}{\partial t} = -\nabla\cdot\mathbf{u}N + \frac{\partial gN}{\partial r} + b; \quad N \equiv N(\mathbf{x}, r, t) \tag{1.19}$$

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

$$m_k \equiv \int_{-\infty}^{+\infty} r^k N(\mathbf{x}, r, t)\,\mathrm{d}r \tag{1.20}$$

By applying the moment transformation to 1.19, we recover the moment transport equation

$$\frac{\partial m_k}{\partial t} = -\nabla\cdot\mathbf{u}m_k - k\int_{-\infty}^{+\infty} r^{k-1} gN\,\mathrm{d}r + \int_{-\infty}^{+\infty} r^k b\,\mathrm{d}r \equiv F(m_{k+i}); \quad i \in \mathbb{Q} \tag{1.21}$$

The right-hand-side (RHS) in Eq. (1.21) 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.

1.5 Basic CFD Algorithm using Expression Graphs

Wasatch is currently able to solve the momentum equations in three dimensional space with periodic boundary conditions using the graph-based algorithm. An example problem that uses two dimensional

1.5.1 Improved Boundary Conditions Support

We have recently revisited the boundary conditions processing in Wasatch. The user/developer has the flexibility of automatically specifying Dirichlet or Neumann boundary conditions on any field type (Scalar, Staggered) and any expression used in the solution process. Furthermore, the boundary condition functionality is now fully templated on the type of field. This allows for partial and full specialization for fields that require specific treatment such as normal stresses.

1.6 Runge-Kutta Time Integrator

There is currently a working version of a third order, strong stability preserving Runge-Kutta integrator. There are two problems to be resolved as of this writing, one pertaining to the pressure projection on multi-patch problems while the other has to do with field memory management at various integrator stages. An abstraction for multistage integrators is in order to address these problems as well as provide support for higher order time integrators in the future.

Chapter 2

Order of Accuracy Verification

Order-of-accuracy analysis is based on the concept that the error must be reduced as spatial and temporal steps are reduced. By comparing the errors at different grid levels, one can estimate the theoretical order of accuracy for a code. First, we review the general concept.

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. 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

$$E = |\phi_{\text{exact}} - \phi_{\text{numerical}}| \tag{2.1}$$

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

$$\phi_{i+1} = \phi_i + \frac{\mathrm{d}\phi_i}{\mathrm{d}x}\Delta x + \frac{\mathrm{d}^2\phi_i}{\mathrm{d}x^2}\frac{\Delta x^2}{2!} + \frac{\mathrm{d}^3\phi_i}{\mathrm{d}x^3}\frac{\Delta x^3}{3!} + \dots \tag{2.2}$$

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

$$\frac{\mathrm{d}\phi_i}{\mathrm{d}x} = \frac{\phi_{i+1} - \phi_i}{\Delta x} + \underbrace{\frac{\mathrm{d}^2\phi_i}{\mathrm{d}x^2}\frac{\Delta x}{2!} + \frac{\mathrm{d}^3\phi_i}{\mathrm{d}x^3}\frac{\Delta x^2}{3!} + \dots}_{\text{truncation error}} \tag{2.3}$$

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 $\mathscr{O}$ terminology

$$T.E. = \underset{\Delta x \to 0}{\mathscr{O}}(\Delta x^m), \quad m \in \mathbb{R}^+ \tag{2.4}$$

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

$$\lim_{\Delta x \to 0} \frac{T.E.}{\Delta x^m} = M \tag{2.5}$$

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

$$T.E. = M\Delta x^m \tag{2.6}$$

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

$$E = \alpha \Delta x^p \tag{2.7}$$

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.

2.1 Order-of-Accuracy in a Single Dimension

To solve for p in Eq. (2.7), one needs two equations. 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

$$\begin{cases} E_1 = & \alpha \Delta x_1^p \\ E_2 = & \alpha \Delta x_2^p \end{cases} \tag{2.8}$$

These equations can be easily solved by taking the difference of their logarithms. One recovers

$$\ln E_1 - \ln E_2 = p \ln \Delta x_1 - p \ln \Delta x_2 \tag{2.9}$$

or

$$\ln \frac{E_1}{E_2} = p \ln \frac{\Delta x_1}{\Delta x_2} = p \ln r_x \tag{2.10}$$

finally, the observed order of accuracy is at hand

$$p = \frac{\ln \frac{E_1}{E_2}}{\ln r_x} \tag{2.11}$$

A word of caution must be said about temporal order of accuracy analysis. Many people think that by taking

It is easiest to first determine the temporal order of accuracy of a transient CFD code. As discussed in the previous paragraph

2.2 Combined Spatio-Temporal Order-of-Accuracy Analysis

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

$$E = u_{\text{exact}} - u_{\text{numerical}} = \alpha \Delta t^p + \beta \Delta x^q + \gamma \Delta y^r + \delta \Delta z^s \tag{2.12}$$

This excludes all terms that have mixed orders in them.

Table 2.1: 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**

2.3 Verification of Wasatch

Temporal Order of Accuracy Wasatch currently entertains both forward Euler (FE) and 3rd 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. 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

$$\frac{d\phi}{dt} = L(\phi) \tag{2.13}$$

written in discretized form, this leads to

Time-Dependent Source Term Our choice falls upon

$$\frac{d\phi}{dt} = \sin t; \quad \phi(0) = -1 \tag{2.14}$$

The analytical solution for this case is

$$\phi_{\text{exact}} = -\cos t \tag{2.15}$$

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

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

$$\frac{d\phi}{dt} = \phi; \quad \phi(0) = 1 \tag{2.16}$$

Since this is an ODE, the spatial resolution is set to [2, 2, 2] in Wasatch. The code is run with both FE and RK3 time integrators at several time-steps ranging from

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.

Table 2.2: 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**

Table 2.3: 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**

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:

$$u(x,y,t) = 1 - A\cos(x-t)\sin(y-t)\exp(-2\nu t) \tag{2.17}$$

$$v(x,y,t) = 1 + A\sin(x-t)\cos(y-t)\exp(-2\nu t) \tag{2.18}$$

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

$$E = norm(u_{exact} - u_{numerical}) \tag{2.19}$$

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.

Chapter 3

Scalability

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

Chapter 4

Boundary Conditions

4.1 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. 4.1.

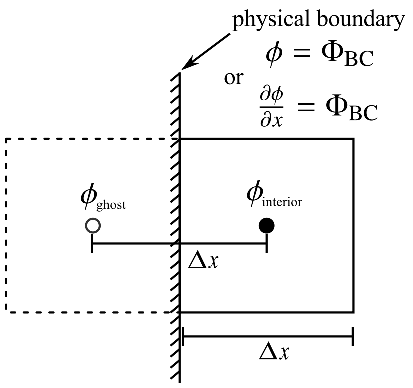

Figure 4.1: Boundary condition specification in Wasatch. Here, Φ_{BC} designates the desired value of the scalar or its flux at the boundary, $\phi_{interior}$ is the interior value of the scalar, and ϕ_{ghost} is the ghost value to be set.

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

$$\phi_{ghost} = \alpha \phi_{interior} + \beta \Phi_{BC} \tag{4.1}$$

where α and β are real coefficients that depend on the type of boundary condition specified. These are summarized in Table 4.1 .

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

4.2 Scalar Transport Boundary Conditions

4.2.1 Dirichlet Boundary Conditions

4.2.2 Neumann Boundary Conditions

4.3 Fluid Flow Boundary Conditions

4.3.1 Inlets, Walls, and Moving Walls

4.3.2 Pressure Outlet

Chapter 5

Modeling General Scalar Transport

Chapter 6

Modeling Species Transport

Chapter 7

Modeling Dispersed-Phase Flows

Chapter 8

Summary of Current Capabilities and Work in Progress

Table 8.1: 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